

UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA



SISTEMAS INFORMÁTICOS Y COMPUTACIÓN ARQUITECTURA DE APLICACIONES

Integrantes:

Jheyson Steven Gaona Pineda
Cristian Enrique Aguirre Minga

Docente:

Jorge Afranio López Vargas

Tema:

GraalVM

Loja – Ecuador

Arquitectura de Aplicaciones

Proyecto GraalVM

Objetivo general:

Conocer el comportamiento de GraalVM y ver cómo interpreta (ejecuta) los diferentes escenarios (lenguajes de programación), con la finalidad de aprender a usar una serie de diversos lenguajes al mismo tiempo, en la misma máquina y sin pérdida de rendimiento.

Objetivos Específicos:

- Elaborar una guía con procedimientos paso a paso, incluyendo la instalación hasta el uso del mismo.
- Mediante el uso de algunos lenguajes de programación se mostrará el funcionamiento de GraalVM.

Introducción:

Antes de iniciar es factible dejar en claro que es GraalVM; este un compilador desarrollado por Oracle, y tiene como funcionalidad eliminar la exclusión entre diversos tipos de lenguaje, ya sea si tenemos una parte del programa escrito en java y la otra escrito en Python con GraalVM podemos agruparlos y tener un solo producto final, se listara los tipos de lenguaje de programación que soporta el compilador así como las características que detallan su funcionalidad, se brindara una guía de instalación de GraalVM para el sistemas Linux y en si su funcionalidad.

La finalidad de este documento es enseñar y dar a conocer a los lectores, programadores, etc... como podemos construir un software hecho en diversos lenguajes de programación, debido a que todos los desarrolladores no optan a un solo tipo de lenguaje de programación.

Marco Teórico:

Antes de empezar conozcamos los siguientes significados para estar al día:

- **JVM** (Java Virtual Machine), Latorre (2010) expresa “máquina virtual java, es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el java bytecode), es cual es generado por el compilador del lenguaje java.”
- **LLVM** (Low Level Virtual Machine), es una infraestructura de compilación utilizada para optimizar el código para varios lenguajes de programación diferentes

Oracle en 2018 realiza el lanzamiento su máquina virtual políglota, el GraalVM que proporciona ventajas de rendimiento para eliminar el aislamiento entre lenguajes de programación, permitiendo funcionar varios lenguajes en un tiempo de ejecución compartido, es decir varios lenguajes de programación que trabajan juntos sin ningún tipo de conflicto, GraalVM se basa en Oracle Labs JDK 8 (Java Development Kit 8).

GraalVM, además, permite la ejecución de código nativo en la JVM a través de un front-end basado en LLVM (proyecto Sulong).

- **El Front-end:** como señala campusMVP (2018) “traduce código fuente en una representación intermedia (IR) de éste. LLVM IR es un lenguaje de bajo nivel parecido a ensamblador. Sin embargo, omite cualquier información específica del hardware.”

- **Proyecto Sulong:**

De acuerdo con Schatz (2018):

Sulong is a high-performance LLVM bitcode interpreter built on the GraalVM by Oracle Labs.

Sulong is written in Java and uses the Truffle language implementation framework and Graal as a dynamic compiler.

With Sulong you can execute C/C++, Fortran, and other programming languages that can be transformed to LLVM bitcode on Graal VM. To execute a program, you have to compile the program to LLVM bitcode by a LLVM front end such as clang.

- **Truffle:** Douglas (2018) define “es una biblioteca de código abierto para crear implementaciones de lenguajes de programación como intérpretes para los árboles de sintaxis abstracta auto-modificables.”
- **Clang:** Delgado (2015) nos dice “Clang es un compilador de código abierto para los lenguajes de la familia C: C, C++, Objective-C y Objective-C++”

GraalVM permite ejecutar programas java de una manera más rápido, ya sea creando imágenes nativas para poder compilarlas con anticipación. GraalVM está diseñado para la integralidad en base de datos ya sea con Oracle Database y MySQL. Permite reutilizar bibliotecas desde JAVA, R o Python.

Seaton (2018) argumenta “GraalVM es una máquina virtual políglota para ejecutar aplicaciones escritas en JavaScript, Python, Ruby, R, lenguajes basados en JVM como Java, Scala, Kotlin, Clojure y lenguajes basados en LLVM como C y C ++.”

La siguiente imagen da una mejor idea de las posibilidades de GraalVM

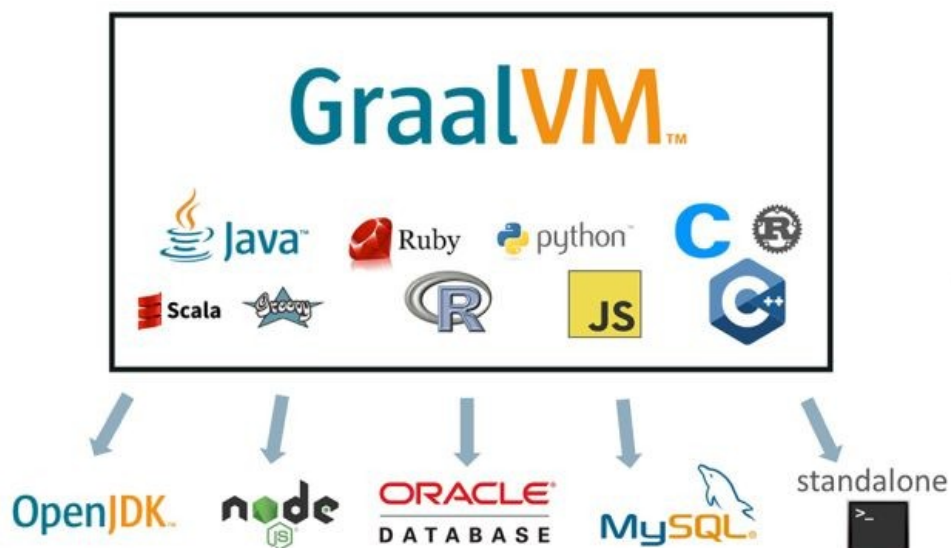


Figura 1. Algunos de muchos lenguajes de programación con los que GraalVM funciona disponible en:
<https://static.imasters.com.br/wp-content/uploads/2018/04/GRAA.jpg>

Entre las otras capacidades de GraalVM se encuentran:

- Un mecanismo para crear imágenes nativas pre-compiladas para lenguajes JVM con inicio "instantáneo" y con poco uso de la memoria.
- Depurador, perfilador y herramientas de visor de montones agnósticos al lenguaje.

Krill (2018) menciona:

En lugar de convertir las estructuras de datos en los límites del lenguaje, GraalVM permite a los objetos y arrays ser utilizados directamente en lenguajes externos. Por ejemplo, se puede acceder a la funcionalidad de biblioteca de Java desde el código de Node.js (JavaScript).

Oracle (2018) señala:

La ejecución de su aplicación dentro de una máquina virtual Java viene con costos de inicio y huella. GraalVM tiene una función para crear imágenes nativas para aplicaciones basadas en JVM existentes. El proceso de generación de imágenes emplea un análisis estático para encontrar cualquier código al que se pueda acceder desde el método Java principal y luego realiza una compilación anticipada (AOT).

Manual de instalación GraalVM para Linux:

Como en cualquier lenguaje se necesita la instalación previa de las librerías para la ejecución del trabajo, para ello se describe forma breve algunas indicaciones para trabajar con GraalVM.

Recomendaciones a tener en cuenta previo a la instalación:

- a. En caso de no tener instalado el JDK, debe instalarlo pues GraalVM en pocas palabras es un producto de Oracle.
- b. Está disponible para Linux y Mac OS (ver más detalle para una instalación adecuada).

Pasos para su instalación:

1. Descargar la **versión Edición Enterprise (EE)** («Oracle Labs GraalVM: Descargar», s. f.), esta versión es creada para desarrollo mientras que la “Edición comunitaria (CE)” («Oracle Labs GraalVM: Descargar», s. f.) fue diseñada para aportar a la comunidad.
2. Una vez descargado:
 1. Extraiga los archivos.
 2. Haciendo uso de la terminal ubíquese en /bin y agregue:
`export PATH=/path/to/graalvm/bin:$PATH`
 3. Para verificar el paso anterior echo \$PATH
3. Es importante saber la versión con la que se trabaja **lli -version₁** (para ello ejecute el comando en el terminal).

A continuación, una tabla con algunos aspectos a tomar en cuenta para trabajar con algunos de los lenguajes disponibles.

Lenguaje	Interprete LLVM	JAVA	JAVA/Kotlin Application	Ruby
Librería requerida	sudo apt install clang	JAVA	Maven, GraalVM	gu install ruby, gem install chunky_png
Comando para compilación	clang -c -O1 -emit-llvm nombre.c	javac nombre.java		
Comando para ejecución	lli nombre.bc	java nombre	./run.sh	ruby -r chunky_png -e "puts ChunkyPNG::Color.to_hex(ChunkyPNG::Color('mintcream @ 0.5'))"

Anexos:

JAVA

```
public class Simple {
    public static void main(String[] args) {
        System.out.println("JAVA, hola");
    }
}
```

Figura 2. Clase de java que imprime “JAVA, Hola” desde consola

```
pc@cristian:~/Documentos/GraalVM/simple$ javac Simple.java
pc@cristian:~/Documentos/GraalVM/simple$ java Simple
JAVA, hola
pc@cristian:~/Documentos/GraalVM/simple$
```

Figura 3. Compilar desde la terminal de linux, para generar el archivo ya compilado que es el .class

Interprete LLVM

```
#include <stdio.h>

int main() {
    printf("GraalVM!\tLenguaje C\nIngrese el valor para a:\t");
    int a;
    scanf("%d", &a);
    if (a <= 10)
    {
        a *=2;
    }else{

```

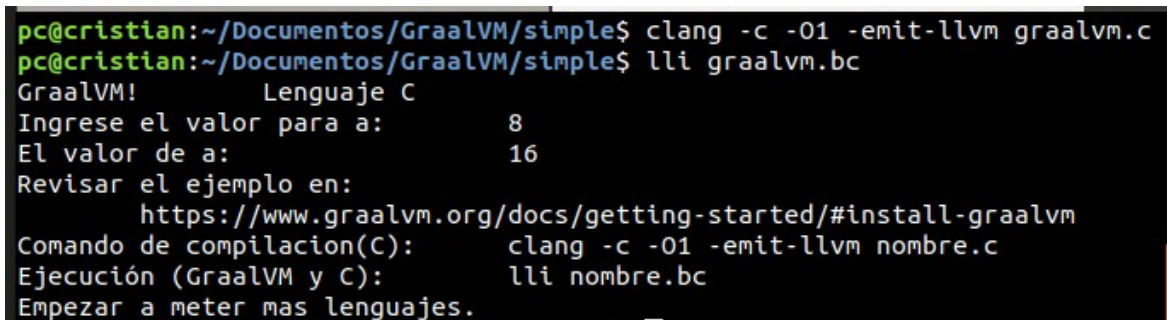
```

    a +=100;
}
printf("El valor de a:\t\t\t%d\n", a);
char link[] = "https://www.graalvm.org/docs/getting-started/#install-graalvm";
char compilacion[] = "clang -c -O1 -emit-llvm nombre.c";
char ejecucion[] = "lli nombre.bc";
printf("Revisar el ejemplo en:\n\t%s\nComando de compilacion(C):\t%s\nEjecución\n(GraalVM y C):\t%s\nEmpezar a meter mas lenguajes.\n", link, compilacion,
ejecucion);

return 0;
}

```

Figura 4. Programa escrito en C, con finalidad de usar el interprete de GraalVM



```

pc@cristian:~/Documentos/GraalVM/simple$ clang -c -O1 -emit-llvm graalvm.c
pc@cristian:~/Documentos/GraalVM/simple$ lli graalvm.bc
GraalVM!      Lenguaje C
Ingrese el valor para a:      8
El valor de a:      16
Revisar el ejemplo en:
      https://www.graalvm.org/docs/getting-started/#install-graalvm
Comando de compilacion(C):      clang -c -O1 -emit-llvm nombre.c
Ejecución (GraalVM y C):      lli nombre.bc
Empezar a meter mas lenguajes.

```

Figura 5. Desde terminal de linux, se escribe lo siguiente para ejecutar el código de la Figura 4.

Para una mejor comprensión puede guiarse con los ejemplos simples proporcionados por GraalVM, en este enlace <https://www.graalvm.org/docs/reference-manual/polyglot/> aquí encontrará un ejemplo con cada lenguaje con el que podrá trabajar; además de los comandos necesarios para compilar y ejecutar.

Banco de Preguntas.

1. Que es GraalVM y para que sirve

GraalVM es una máquina virtual políglota sirve para ejecutar aplicaciones escritas en JavaScript, Python, Ruby, R, lenguajes basados en JVM como Java, Scala, Kotlin, Clojure y lenguajes basados en LLVM como C y C ++.

2. ¿En que Front-end esta basado la LLVM de GraalVM y que función cumple?

La Front-end de la LLVM de GraalVM está basado en el proyecto Sulong, y su función es que puede ejecutar C, C++ y otros lenguajes de programación que pueden transformarse a código de bits.

3. ¿Qué es un Truffle en GraalVM?

Es una biblioteca de código abierto para crear implementaciones de lenguajes de programación como intérpretes para los árboles de sintaxis abstracta (AST) auto-modificables

Bibliografía:

Seaton, C. (2018) *GraalVM* [Documento]. Recuperado de <https://www.graalvm.org/>

Krill, P. (2018) *Oracle ha entregado una versión de producción de GraalVM* [Noticia]. Recuperado de <https://cioperu.pe/articulo/25679/oracle-ha-entregado-una-version-de-produccion-de-graalvm/>

Oracle (2018) *GraalVM* [Documento]. Recuperado de <https://www.graalvm.org/docs/why-graal/>

Latorre, G. (2010) *JVM - JDK - JRE - Conceptos Fundamentales de la P.O.O.*[Blog]. Recuperado de <https://gl-ept-programacion-ii.blogspot.com/2010/03/jvm-jdk-jre-conceptos-fundamentales-de.html>

campusMVP (2018) *Una introducción a los compiladores: cómo hablar con una computadora (pre-Siri)* [Artículo]. Recuperado de <https://www.campusmvp.es/recursos/post/una-introduccion-a-los-compiladores-como-hablar-con-una-computadora-pre-siri.aspx>

Schatz, R. (2018) *Sulong* [Artículo].disponible en <https://github.com/oracle/graal/tree/master/sulong>

Douglas, S. (2018) *The Truffle Language Implementation Framework* [Artículo]. Disponible en <https://github.com/oracle/graal/tree/master/truffle>

Delgado, Pérez, P. (2015) *Clang Un compilador de código abierto* [Documento]. Disponible en https://rodin.uca.es/xmlui/bitstream/handle/10498/16912/clang_tutorial.pdf

Oracle Labs GraalVM: Descargar. (s. f.). Recuperado 19 de noviembre de 2018, de <https://www.oracle.com/technetwork/oracle-labs/program-languages/downloads/index.html>

Implement Your Language for GraalVM. (s. f.). Recuperado 19 de noviembre de 2018, de <https://www.graalvm.org/docs/graalvm-as-a-platform/implement-language/#ide-setup>