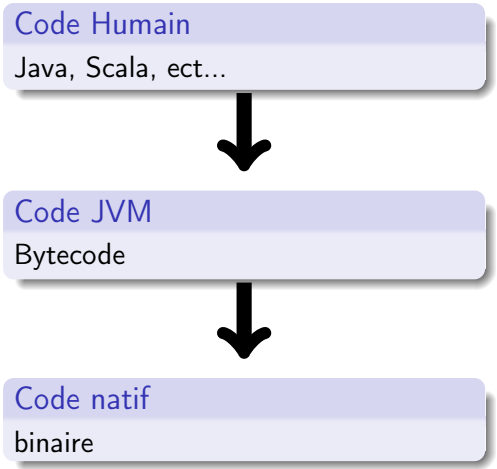
Implémentations Java : compilateur Graal vs. JIT HotSpot

Féaux De Lacroix Martin Herbert Damien


Code Humain
Java, Scala, ect...
Code JVM
Bytecode
Code natif
binaire

Exemple

Exemple java

Exemple

Exemple bytecode

JIT

AOT (Ahead-of-time)

Exemple : C
Avantage : Plus rapide (dépend de la proximité avec le langage machine), Plus de temps disponible pour l'analyse et l'optimisation

Interprété

Exemple : Python
Avantage : Programme plus compacte et portable, analyse et optimisation prend en compte des informations systèmes et du runtime

JIT (Just-in-time)

Avantage

Théoriquement, A chaud : plus rapide
Analyse et optimisation prend en compte des informations systèmes et du runtime

Désavantage

Compiler à un coût en temps et en mémoire non nul

L'écosysème GraalVM

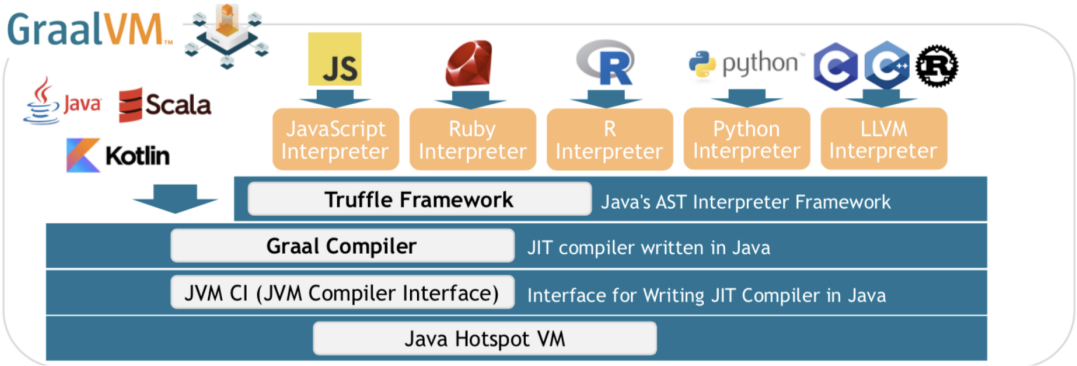


Figure – GraalVM

Tiered Compilation (JVM)

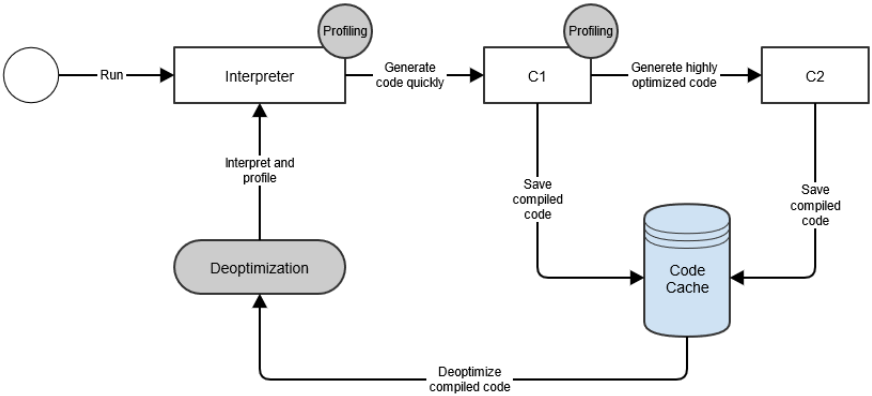


Figure – JIT

C1 compiler

Description

Compilation Rapide

Description technique

"*some* value numbering"
"inlining"
analyse des classes
Représentation intermédiaire "haute"
Représentation intermédiaire "basse" (proche de la machine)
linear scan register allocation

C2 compiler

Avantage

Haute optimisation du code

Description technique

Représentation Intermédiaire : Sea of Nodes SSA
global graph-coloring register allocator
global value numbering
constant optimisation, loop invariant
algebraic identities
method inlining
intrinsic replacement
loop transformations
array range check elimination
ect...

Graal

Avantage

Haute optimisation du code

Description technique

Similaire à C2

Plaquette marketing

"The GraalVM compiler assures performance advantages for highly-abstracted programs due to its ability to remove costly object allocations. Code using more abstraction and modern Java features like Streams or Lambdas will see greater speedups. Low-level code or code that converges to things like I/O, memory allocation, or garbage collection will see less improvement. Consequently, an application running on GraalVM needs to spend less time doing memory management and garbage collection"

Exemple de comparaison

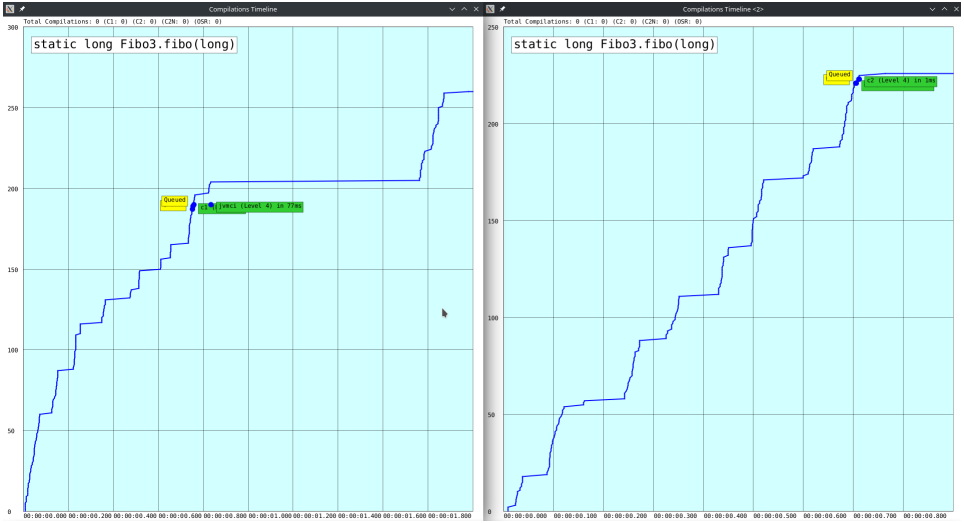


Figure – Graal à gauche, JIT HotSpot à droite

Bibliographie

[1] Graal: How to use the new JVM JIT compiler in real life by Chris Thalinger

[2] Baeldung : Tiered compilation

[3] IBM : documentation jit sdk

[4] Documentation oracle : GraalVM compiler

[5] GraalVM documentation : Introduction

[6] HotSpot documentation

[7] Linear Scan Register Allocation - MASSIMILIANO POLETTO and VIVEK SARKAR

[8] Stackoverflow - inlining

[9] Delphine Demange, Yon Fernández de Retana, David Pichardie. Semantic reasoning about the sea of nodes.

Bibliographie

[1] A Brief History of Just-In-Time - JOHN AYCOCK

[2] Sparse Conditional Constant Propagation by Mark Anastos

[3] Global code motion/global value numbering - Cliff Click