­

LOG2410

Machines virtuelles, WebAssembly, et JVM

Département de génie informatique et génie logiciel

**Fabrice *NDUI (*1914377*)***

**Trimestre :** Hiver 2019

Date de remise (07/04/19)

# Description de la librairie

**Machines virtuelles, WebAssembly et JVM:**L'une des différences fondamentales qui distinguent les paradigmes de programmation des langages C++ et Java tient au fait que le C++ est un langage qui est habituellement compilé vers le langage machine spécifique de la plateforme sur laquelle le code doit être exécuté, alors que le Java est plutôt compilé vers un langage intermédiaire qui est ensuite interprété par une machine virtuelle. Récemment, afin d'améliorer la performance des applications s'exécutant dans un fureteur internet, un nouveau standard de machine virtuelle a été proposée afin d'interpréter un langage intermédiaire appelé WebAssembly. Quelles sont, selon vous, les principales différences entre la machine virtuelle de Java (JVM) et celle proposée pour interpréter le WebAssembly ? Pour quelle raison la JVM n'a-t-elle pas été utilisée comme machine virtuelle pour le WebAssembly ? Comment la performance des deux types de machines virtuelles se compare-t-elle actuellement ? Selon vous, la conception de code C++ destiné à être compilé et exécuté sur une machine virtuelle WebAssembly doit-elle différer de la conception de code C++ conçu pour s'exécuter de façon native sur une plateforme spécifique ? Justifiez votre réponse.

Definition d’une machine virtuelle

Différence entre langage interprèté et langage compile

Avantages du jvm

Inconvénient du jvm

Avantages du javascript

Inconvenient du javascript

Avantafe du webassembly

Inconvenient du webassembly

Quelles sont, selon vous, les principales différences entre la machine virtuelle de Java (JVM) et celle proposée pour interpréter le WebAssembly ?

La machine virtuelle utilise pour interpreter le webAssembly est en fait le meme utilsé pour interpreter javascript, donc il est déjà testé et rapide, et cette machine virtuelle est disponible sur tous les navigateurs récents. De l’autre côté, JVM n’est pas disponible sur les navigateurs et necessite un applet afin de permettre la communication entre l’environement du client et le navigateur. Cette option a été populaire pendant une dizaine d’annee avant que javascript ne revienne en force.

Une autre grosses differences est que la machine virtuelle sur le navigateur, est en fait un interpreteur car les differents languages de haut niveau sont compilés en .wasm qui est de bas niveau.

JVM lui compile le java ou meme autre language de haut niveau en un format bytecode qui est plus eloigné du langage machine que wasm, Il est ensuite interpreté pour l’environnement spécifique du client.

Ce qui se traduit par une meilleure vitesse avec .wasm puisqu’il est, il n’y a pas de garbage collector

There are things that your specific version of the JVM will do faster than your specific C++ compiler, and things that your specific C++ compiler will do better than your specific JVM.

 A garbage collector can, at almost any time, decide to pause the process and clean up memory. In almost all cases, this is fast enough to be no problem, but for vital real-time stuff, it is a problem.

Multi-core performance

Startup time

Memory use

Today Wasm runs just 20% slower than native code execution. This is, by all means, an astonishing result. It’s a format that’s compiled into a sandbox environment and runs within a whole lot of constraints to make sure it has no security vulnerabilities or is very hardened against them. The slowdown is minimal compared to truly native code. What’s more, it will be even faster in the future.

Comment la performance des deux types de machines virtuelles se compare-t-elle actuellement ?

JVM est environ a 10% d’un language natif, comparé à 20% pour le webassembly