Native

Eine der Kernfunktionen von GraalVM ist die Erstellung von nativen Images. Für die Ausführung von Interpretierten Sprachen wird üblicherweise eine Laufzeitumgebung benötigt. Java gehört mit seinem Byte-Code ebenfalls zu dieser Klasse, weswegen wir in den vorherigen Übungen eine Laufzeit installiert haben. Native Anwendungen kommen ohne solche Installationen aus. Ziel dieser Übung ist zu sehen, wie wir native Images erstellen können und welche Performanceunterschiede es gibt.

Für die Erstellung müssen wir zunächst einige Libraries installieren, die unter der Haube für den Compilierungsvorgang verwendet werden:

apt-get update
apt-get install build-essential libz-dev zlib1g-dev

(Auf Windows ist es etwas komplizierter, hier braucht man die Visual Studio Build Tools 2017. Alternativ kann GraalVM auch auf eine Docker-Installation für den Compile-Vorgang zurückgreifen)

Anschließend installieren wir die GraalVM-Unterstützung:

gu install native-image

Wenn wir nun eine Binary installieren wollen, dann können wir dies einfach mit dem native-image Tool machen. Nehmen wir uns wieder unser einfaches HelloWorld-Beispiel vom Anfang des Workshops:

javac HelloWorld.java
native-image HelloWorld

Der Prozess dauert einen kleinen Moment. Wir sehen, wie er im ersten Schritt prüft, wie das System aufgebaut ist und ob alle benötigten Komponenten für den Prozess existieren. Anschließend analysiert er den Quellcode und führt seine Optimierungen (z. B. inlining oder Tree-shaking) durch. Am Ende purzelt die fertig ausführbare Datei heraus:

./helloworld

Wir sehen, dass das Hello World sehr schnell auf dem Terminal erscheint.

time ./helloworld
time java HelloWorld

In der Tat ist das neue Image erheblich schneller. Schon bei dem kleinen HelloWorld ergibt sich ein Unterschied von mehreren hundert Prozent. Bei einem großen Projekt ergeben sich hier erhebliche Performanceunterschiede.