Workload Characterization of JVM Languages

Originally developed with a single language in mind, the Java Virtual Machine (JVM) ended up being targeted by numerous programming languages; its automatic memory management, just-in-time compilation, and adaptive optimizations making it an attractive execution platform. However, the garbage collector, the just-in-time compiler, and other optimizations and heuristics were designed primarily with the performance of Java programs in mind. In this article, we aim at contributing to understanding the character of the workloads imposed on the JVM by both dynamically-typed and statically-typed JVM languages. Specifically, we are looking for traits that either set the non-Java JVM languages apart, or that they have in common - both representing potential optimization opportunities.

To this end, we introduce a new set of dynamic metrics for workload characterization, along with a comprehensive and easy-to-use toolchain to collect the metrics. We apply our toolchain to applications written in six JVM languages (Java, Scala, Clojure, Jython, JRuby, and JavaScript), and discuss the findings and implications.

Author: Walter Binder (walter.binder@usi.ch)