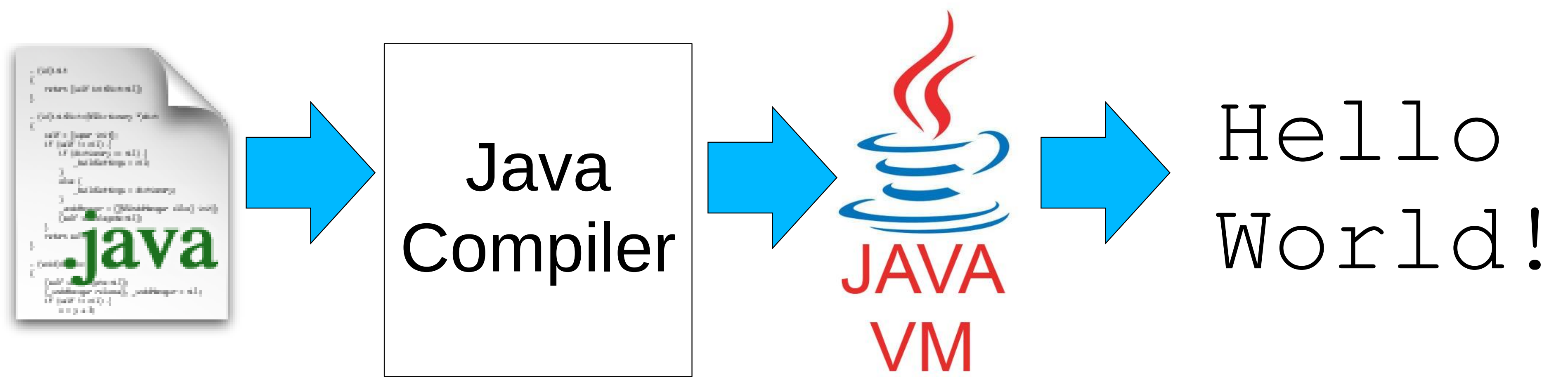
Application performance improvements through VM parameter modification after runtime analysis

Prof. Dr. Kenneth Kent, Prof. Dr. André Hinkenjann, Nicolas Neu
University of New Brunswick, Bonn-Rhein-Sieg University, IBM Canada
Faculty of Computer Science
ken@unb.ca , andre.hinkenjann@h-brs.de, nicolas.neu@unb.ca,

Problem Description

The Java Virtual Machine, or JVM, is needed to execute Java bytecode. While early implementations showed a poor performance, current versions serve as a powerful runtime environment not only for Java, but also for other languages compiling to Java bytecode like Scala or Clojure. Features like automatic Garbage Collection make programming easier for the developer but also causes performance hits compared to programming methods with manual memory management.



This is why a lot of the VM behavior can be finetuned by adjusting one or more parameters when starting the application. You can for example adjust the garbage collection policy, the heap size or the Just-In-Time compilation. The problem is to decide, which to use for which type of application. Preferably this should be possible with the Java VM treated as a black box. This way the developer can disregard the inner workings of the VM and concentrate on his own implementation when optimizing.

Project Goal

The VM allows us to gather a lot of information about the runtime behavior of an application by providing heapdumps, stackdumps and detailed logfiles. Analyzing this data can help us to categorize applications depending on their runtime behavior. Possible categories are for example long/short running applications or IO/computation heavy applications. Each of them need a specialized parameter configuration for maximum performance. Another important factor is that performance may not only refer to the speed but can also mean memory consumption or average data throughput rate.

With these categorizations in mind, the goal is to devise a ruleset which maps a certain runtime behavior to an optimal parameter configuration which can be used to fine tune the VM.

Verification

Since there are many different types of applications, it is important to verify the results with a wide range of benchmarks. After running a benchmark, it is checked whether the runtime behavior matches the pattern of one of the rules in the ruleset. If this is the case, the benchmark is run again with the proposed parameters. This should results in a performance increase of some kind. Widely used benchmarks for such cases are for example the DaCapo benchmark suite or different Java spec benchmarks.

spec®

DaCapo