Runtime Code Generation for Interpreted Domain-Specific Modeling Languages — Software Appendix

Tom Meyer, Tobias Helms, Tom Warnke, Adelinde M. Uhrmacher

Abstract

This archive contains the needed software to execute the experiments and reproduce the results described in the paper "Runtime Code Generation for Interpreted Domain-Specific Modeling Languages" published at the Winter Simulation Conference 2018. The experiments illustrate the effectiveness of runtime code compilation of rate expressions used by the interpreted domain-specific modeling languages ML-Rules based on microbenchmarks and macrobenchmarks.

1 General Information

Pi_Name: Tom Meyer, Tobias Helms, Tom Warnke, Adelinde M. Uhrmacher

Pi_Email: {tom.meyer, tobias.helms, tom.warnke, adelinde.uhrmacher}@uni-rostock.de

Pi_Affiliation: Modeling and Simulation Group[1], Institute of Computer Science, University of Rostock, 18051 Rostock, Germany

Location: Institute of Computer Science, Albert-Einstein-Strae 22, 18059 Rostock, Germany

Funding: This research is supported by the German Research Foundation (DFG) via research grant ESCeMMo (UH- 66/14).

Date: 2018

2 Archive Structure

models: contains the ML-Rules[2] models used by the experiments

[1] `https://mosi.informatik.uni-rostock.de`

[2] Maus, C., S. Rybacki, and A. M. Uhrmacher. 2011. Rule-based multi-level modeling of cell biological systems. BMC Systems Biology 5(166).

repo: contains the software libraries (including sources) of ML-Rules and the JMH[3] experiments

.mvn, mvnw, mvnw.cmd: scripts and library to start the experiments

pom.xml: Maven[4] project used to start the experiments thereby loading all needed software dependencies automatically

3 System Requirements

Java Runtime Environment 8[5]

4 How to start the experiments

Extract all files and directories of the archive to a new directory.

Windows: execute `mvnw.cmd exec:java@microbenchmark` or `mvnw.cmd exec:java@macrobenchmark` in a terminal

Linux: execute `mvnw exec:java@microbenchmark` or `mvnw exec:java@macrobenchmark` in a terminal

5 Results

A summary of the experiment results is directly written to the terminal at the end of an experiment. The average runtime in microseconds per iteration/replication is shown for the configurations. The results using runtime compilation of valid expressions are marked with "Compiled".

The following mapping is applied for the test names of the microbenchmarks and the arithmetic expressions described in the paper:

- `constantTest`: `1 + 2`
- `variableTest`: `x + y`
- `conditionTest`: `if x > 5 then 1 else 2`
- `complexTest`: `(x1 + x2) - ((x3 / x4) + x2 + x4)`

The macrobenchmarks need $\approx$ 30 minutes to be computed on an average PC. Frequency scaling (like Intel Turbo Boost® or Intel SpeedStep®) should be disabled.

[3] `http://openjdk.java.net/projects/code-tools/jmh/`
[4] `https://maven.apache.org/`
[5] `https://www.java.com`