A JVM-based Compiler Strategy for the R Language

Helena Kotthaus[⋆], **Sascha Plazar, Peter Marwedel**

Computer Science 12, TU Dortmund University
[⋆]Contact author: helena.kotthaus@tu-dortmund.de

Keywords: *R* Language Optimization, *Java*, Compiler

The *R* programming language has become invaluable for analysis and evaluation of statistical methods. *R* is a multi-paradigm language with functional characteristics, a dynamic type system and different object systems. These characteristics support the development of statistical algorithms and analyses at a high-level of abstraction. Like for many dynamic languages, *R* programs are processed by an interpreter. Especially in the domain of statistical learning algorithms and bioinformatics, e.g. when analyzing high-dimensional genomic data, this interpretation often leads to an unacceptably slow execution of computation-intensive *R* programs. Our goal is to optimize the execution runtime of such *R* programs. Therefore, we plan to develop a JVM-based compiler strategy including an *R*-interpreter written in *Java* and an extensible just-in-time (JiT) compiler.

With the use of an *R*-interpreter written in *Java* [1], the execution of *R* programs could be optimized: By targeting the JVM, JiT compilation is enabled within the interpreter code. However, transferring the *R* interpretation process to the JVM does not automatically lead to high optimization potential, because *R* programs still need to be interpreted. Additional source level optimizations should be applied to the intermediate representation (IR) produced by the *R*-parser. Although JiT compilation could already speed up the interpretation process, the native JiT-compiler is not aware of the *R* language and its specific optimization needs. In order to push more aggressive optimizations, the JiT-compiler should be extended by knowledge about *R* characteristics to enable language specific low-level optimizations and generate highly optimized machine code. For this purpose, the Graal JiT-compiler [2], which is especally designed for extensibility, should be employed.

On our poster we present our optimization ideas and development plans for the JVM-based compiler strategy for the *R* Language.

References

[1] Bertram, A. (2012). JVM-based Interpreter for the R Language for Statistical Computing. `http://code.google.com/p/renjin`.

[2] Würthinger, T. (2011). Extending the graal compiler to optimize libraries. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, pp. 41–42.