Optimizing Execution Runtimes of R Programs

Sascha Plazar[*], Peter Marwedel[*], Jörg Rahnenführer[†]
Department of Computer Science[*], Department of Statistics[†]
TU Dortmund University, D-44221 Dortmund, Germany
{sascha.plazar | peter.marwedel}@tu-dortmund.de [*]
rahnenfuehrer@statistik.tu-dortmund.de[†]

Abstract

The *GNU R* language is very popular in the domain of statistics. Its functional character supports the rapid development of statistical algorithms and analyses. Statisticians around the world profit from the immense R package archive *CRAN* where researchers offer their algorithms in form of R programs for free usage.

One of the disadvantages of R is that programs have to be evaluated and processed by a runtime interpreter. Such an interpretation requires a lot of time and delays the execution. Thus, a lot of computing power is wasted compared to imperative languages like *ANSI C*, which can be automatically optimized and translated to machine code by a sophisticated compiler.

This abstract proposes an approach which exploits various optimizations and the workflow of toolchains for imperative languages to accelerate R programs. To this end, we are proposing a toolchain which is divided into four phases. Phase 1 applies source level optimizations on R. Phase 2 transforms such optimized R code and libraries to C code. In the next phase, the generated C in turn can be optimized, employing existing and newly developed optimization techniques. In the final phase, a standard compiler will translate the C code into machine code for a fast execution on a host machine. Our goal is to speed up R programs automatically on average by a factor of 50 or better.

In a case study, we manually applied the optimizations *common subexpression elimination* (*CSE)* and *dead code elimination* (*DCE*) to R programs to evaluate their positive impact on the programs' execution times. CSE replaces multiple occurrences of the same expressions by a single variable holding the same value. Applied to *strMCMC*, a function for estimating graphical models with a Markov chain Monte Carlo approach, CSE was able to remove eight expressions which otherwise would have to be recomputed several times. DCE removes code which would be executed on no account. By applying DCE to the same program, three `if`-statements inside the commonly used `which()` function could be removed which always evaluate to false. These optimizations reduced the overall execution time by 10% and 5%, respectively.

In order to demonstrate the advantages of avoiding a time consuming interpretation of R programs to achieve high performance, we exemplarily translated pieces of R code into C. For this purpose, we evaluated the hot spot of the frequently used R package *rda* for *Regularized Discriminant Analysis*. By translating a single `for` loop of rda's `apply()` function and compiling it with the *GCC* compiler, we were able to speed up this function by a factor of 90. This led to a total reduction of 71% concerning the overall runtime of the rda package.

These excellent results attest that our envisioned toolchain will be highly effective for accelerating R programs. The results also show that a speedup by a factor of 50 is achievable by optimizing R programs and translating them into an imperative language in order to generate efficient machine code.