ANALYZING THE IMPACT OF RUBY'S DYNAMIC FEATURES ON PERFORMANCE

Rajika Abeyrathne

A dissertation submitted in partial fulfillment for the requirement for Master of Science degree in Advanced Software Engineering

Department of Computing

Informatics Institute of Technology, Sri Lanka

in Collaboration with

University of Westminster

2020

Abstract

Programming languages are at the center of software engineering. At the design stage of a project, best suited technologies are evaluated to implement a solution. One of the most important decisions is the programming language that would be used to implement a system. The requirements of a project are evaluated, and programming languages are chosen. The programming languages are chosen based on their features, such that they could satisfy all the requirements of the solution to be built. Language syntax, semantics and compatibility are often considered. However, in almost all the cases, performance is considered as a major deciding factor. Specially if the solution demands specific performance measures.

Ruby is a widely used programming language for web development. It is a dynamic and interpreted language which is loved by developers in the web development community. One of the major reasons for the wide usage is its proper object-oriented nature as an interpreted language and the elegant syntax. However, it is a widely known fact that its performance is not the best among other choices. The performance gap is due to various factors such as the usage of an interpreter, addition of metaprogramming with dynamic capabilities. There have been several efforts to improve the performance of Ruby. Introducing JIT was a significant addition to the traditional interpreter. This has shown promising results with various use cases. Few other previous works have been done in the research level which has also shown overall performance improvements.

This dissertation focuses on how dynamic features in Ruby affects the performance. A methodology is presented to quantify the performance gap of dynamic features in Ruby by providing empirical data. Few dynamic features were chosen to be investigated. The implementation consists of two compilers, where one would act like the Ruby interpreter. This was termed as the un-optimized compiler in this dissertation. The other compiler provides optimizations by providing several restrictions to the language design. The latter compiler was termed as the optimized compiler. Ruby source is provided as the input artifact, where the compiler pipeline converts Ruby to Go, and eventually compile Go to machine code. The baseline for performance was established using the un-optimized compiler. The optimized compiler was compared against the un-optimized compiler. The major unit of measurement was time. However, CPU, memory usage was measured against different use cases.

Keywords: Ruby, Interpreter, Performance, Dynamic features, Compiler, Go.