Y790-32707 - Assignment 1: Brief Research Interest Statements

Caner Derici

1 General Audiance

To understand certain behavioral changes under different effects, the scientific principle of experimentation applies to the computer software as well. The story of executing a program on a computer starts with a human readable code in a certain programming language. That code needs to be translated by another program, namely a compiler, into another representation that a machine can read and execute. We are working on a just-in-time (JIT) compiler, namely Pycket, for the Racket programming language. Given a program written in the Racket language, Pycket translates it to a representation that will be evaluated in later phases again by Pycket to produce the final result.

Alternatively Pycket could use the machine representation produced by Racket's own compiler instead of manually creating one to proceed to the final result. Since the Racket's compiler is designed exclusively for Racket, the representation produced by it is highly optimized. The aim of our study is therefore to modify Pycket to use the alternative representation to find out the effects of the certain optimizations that are performed by Racket's compiler to the overall performance of Pycket. As a consequence, this will allow us to understand better the individual effects of distinct optimizations to the performance of JIT compilers in the general case.

2 Expert Audiance

Observing the runtime reaction of a tracing just-in-time (JIT) compiler to the known compiler optimizations over different kinds of programs could significantly improve both the fundamental understanding and the runtime performance of the tracing JIT compilation. Currently we are working on such a compiler, namely Pycket, that is automatically generated by the RPython translator to implement the Racket language. Given a Racket program, Pycket expands the code and creates from the surface syntax an abstract syntax tree (AST) to be evaluated again by the JIT compiler to produce the final result. Our aim is to modify Pycket to generate an AST using the Racket bytecode compiler, which produces highly optimized Racket bytecode.

Running Pycket with the AST's generated from the surface syntax and from the Racket bytecode will allow us to compare and observe the runtime performance improvements that the optimizations induced by the Racket bytecode compiler. Furthermore, selectively enabling distinct optimizations in the Racket compiler and observing the difference in the performance will empirically show how each optimization affects the runtime on which kinds of programs. This we believe significantly improve the understanding of how the known compiler optimizations work with the tracing JIT compilers in the general case.