4TB3 Project Development Plan

David Thompson, 400117555 Rohit Saily, 400144124 Raymond Tu, 400137262 February 22, 2020

---

1 Project Description

1.1 Research Question

How does the compilation time of the P0 Compiler differ between a sequential implementation and a concurrent implementation using Goroutines in Golang?

1.2 Developmental Challenges

1.3 Test Plan

1.4 Documentation

For the documentation of our project we will be using GoDoc. GoDoc is similar to documentation generators for other programming languages, for example JavaDoc for Java and Doxygen for Python. GoDoc generates documentation in PDF or HTML format from comments in the source code. Following the standard conventions for GoDoc will provide easily readable source code and good documentation for all methods implemented in the P0 Compiler. GoDoc is chosen because it is the standard documentation standard for Golang, which is the main programming language we will be using for this project. The conventions for a standard GoDoc comment to generate HTML is easy to read, and alone would already provide readable source code to any viewer. In conclusion, GoDoc is intuitive to use and similar to existing documentation standards we have experience with, which makes it the perfect choice for this project.

GoDoc: https://blog.golang.org/godoc-documenting-go-code

2 Resources

2.1 Transcompilation

Python is generally an interpreted language, but Go is compiled before execution. This means that the Go implementation of the P0 compiler would likely run faster, even if no concurrency is used. For a fair comparison, a few methods are available.

The existing P0 compiler could be manually rewritten to Go. This method will take a long time, and will take away time for refining the Go implementation.

Our preferred route is to transpile the existing Python code to Go instead of manually translating the program. Google offers a Python to Go transpiler called https://github.com/google/grumpy. It allows for compiling Python source code to Go source code, or to compile the code and immediately run it.

If the transpiler does not work, another option would be to compile the Python code and the concurrent implementation to a common language. This is not ideal, because concurrency works differently in other programming languages, and frequently has more overhead. As a result, the efficiency of the Go implementation may be understated. A possible target language to compile both codebases to is JavaScript. There is a Python to JavaScript compiler and a Go to JavaScript compiler. `https://github.com/jashkenas/coffeescript/wiki/List-of-languages-that-compile-to-JS` Python's dynamic typing limits languages that can be selected as a target language.

2.2 Development

2.3 Testing

For testing the correctness of the P0 Compiler in Go, unit testing will be conducted. This unit testing will be done in the standard testing package that is included in Go. This testing package also includes statement-based code coverage checking that will be useful as a metric for how well the code is tested. The difference between this package and other standard unit testing for other languages such as PyTest for Python is that it does not utilize assert statements. Instead an error is thrown after a boolean check coded by the programmer. This will allow the test package to detect multiple errors in the program, because while an assert statement would stop the program after the first error, the test package in Go will instead detect the error and continue running other test cases. In conclusion, using the Go test package we can confirm the correctness of our sequential and concurrent implementation in Go before we test the differences in their runtimes and answer the defined research question.

For timing the execution time of the P0 Compiler, we can use a variety of different time libraries included in Python or Go. To be consistent, one timing library will be chosen and used for testing the execution time of both implementations of the P0 Compiler. This will ensure accurate results to draw a conclusion from to answer the research question. How these libraries will be used for testing will be elaborated on in the Test Plan section.

2.4 Poster Research

3 Division of Labour

4 Weekly Schedule

Week	Deliverables
Week 1	Development Plan and Proposal
	Setting Development Environment
Week 2	Transcompilation and Beginnning of Development
Week 3	Working Prototype
Week 4	Testing and Refinement of Prototype and Documentation
Week 5	Poster Presentation Creation
Week 6	Poster Presentation Creation