JOHANNES KEPLER
UNIVERSITÄT LINZ

Master's Thesis

Ahead-of-time Optimizations in a Java to JavaScript Compiler

Student:	Christoph Schobesberger 066 921 / 11716039 c.schobesberger@gmx.net
Advisor:	Prof. Hanspeter Mössenböck
Co-advisor:	Dr. David Leopoldseder
Start:	1. March 2021

o.Univ.-Prof. Dr.
Hanspeter Mössenböck
Institute for System Software

T +43 732 2468 4340
F +43 732 2468 4345
hanspeter.moessenboeck@jku.at

Secretary:
Karin Gusenbauer
Ext 4342
karin.gusenbauer@jku.at

Graal AOT JS is an ahead-of-time Java to JavaScript compiler that runs on top of the Graal VM [1]. In particular it uses the Substrate VM [2] to compile Java code ahead-of-time and to determine all types, methods and fields that are reachable from the entry point method using a static analysis technique. Graal AOT JS extends Substrate VM with a backend to generate JavaScript instead of machine code. Graal's IR is a hybrid superposition of a control flow graph and a data flow graph. Since Graal IR is possibly unstructured [3, 6] and JavaScript only allows structured control flow, a control flow reconstruction step is necessary. Graal AOT JS implements this with a novel algorithm called “graph tagging” [6]. However, this algorithm fails in complex graphs and has to fall back to a generic solution that is comparably slow. Latest research trends proposed alternative approaches to control flow reconstruction. One such method is the "stackifier" algorithm [4,5] implemented in LLVM [7].

The goal of this thesis is to investigate in which situations the tagging algorithm fails and why. If possible, the algorithm should be adopted such that it can handle all graphs.

Furthermore, a basic version of the stackifier should be implemented, benchmarked and compared to the tagging algorithm. Depending on the results, it should be determined which algorithm works better or if a combination works best. Additionally, optimizations for the stackifier and their effects should be considered and evaluated.

Further opportunities for performance improvement should be investigated and evaluated.

The goals of this thesis are:

- Investigation of where and why the graph tagging algorithm fails.
- Implementation of the stackifier control flow reconstruction algorithm.
- Comparison of the stackifier and tagging algorithm in terms of performance, generated JavaScript code size and complexity.
- Investigating causes of performance issues and code size contributions of the JavaScript image.

Modalities

The progress of the project should be discussed at least every two weeks with the (co-)advisor. A time schedule and a milestone plan must be set up within the first 3 weeks. It should be continuously refined and monitored to make sure that the thesis will be completed in time. The final version of the thesis must be submitted not later than 28. Februar 2022.

[1] http://openjdk.java.net/projects/graal

[2] https://github.com/oracle/graal/tree/master/substratevm

[3] Cristina Cifuentes. “Reverse Compilation Techniques” PhD Thesis [4]https://medium.com/leaning-tech/solving-the-structured-control-flow-problem-once-and-for-all- 5123117b1ee2

[5] https://github.com/llvm/llvm-project/blob/main/llvm/lib/Target/WebAssembly/WebAssem-blyCFGStackify.cpp

[6] David Leopoldseder, et al. “Java-to-JavaScript translation via structured control flow reconstruction of compiler IR” DLS 2015: Proceedings of the 11th Symposium on Dynamic Languages, October 2015

[7] https://llvm.org/