JOHANNES KEPLER
UNIVERSITY LINZ

**DI Manuel Rigger, M.Phil.**
Institute for System Software

T +43 732 2468 4356
F +43 732 2468 4345
manuel.rigger@jku.at

Secretary:
**Birgit Kranzl**
Ext 4341
birgit.kranzl@jku.at

Master's Thesis / Project in Software Engineering

**On-stack Replacement for Sulong**

| | |
|---|---|
| Student: | ... |
| SKZ/Matr.Nr.: | ... / ... |
| Email: | ... |
| Advisors: | DI Manuel Rigger, M.Phil. |
| | DI Dr. Matthias Grimmer |
| Start date: | ... |

Sulong is an LLVM IR interpreter on top of the JVM that can be used to execute code written in C, C++, and other languages [1]. To reach the performance of native code, it uses a dynamic compiler to compile frequently executed functions to machine code.

However, program parts that are executed often but not as part of a function call are either compiled too late or never at all, resulting in high start-up costs and/or low peak performance. An example are long-running loops that are executed as part of the main() function.

On-stack replacement (OSR) is a well-known technique to address this problem, as it allows switching between different representations of a function (e.g. interpreted and compiled) while this function is executed. The Truffle language implementation framework on which Sulong is based provides an OSR mechanism [4]; however, it requires that loops are identified before execution.

The goal of this thesis is to implement OSR for Sulong based on Truffle's OSR mechanism. To this end, loops in the LLVM IR should be identified, as the IR level only contains branch statements. The loop information should then be used to change the interpreter to benefit from Truffle's OSR mechanism.

The scope of this thesis is as follows:

- Implementation of a loop detection mechanism in Sulong.
- Changing the interpreter to benefit from detected loops using Truffle's OSR mechanism.
- An evaluation of the changes in terms of warm-up and peak performance on benchmarks.

Explicit non-goals are:

- Exploiting loop information in every possible case. A simple solution that works in the majority of the cases is preferred to a complicated one that works in every case.

**JOHANNES KEPLER**
**UNIVERSITY LINZ**
Altenberger Straße 69
4040 Linz, Austria
www.jku.at
DVR 0093696

[1] Manuel Rigger, Matthias Grimmer, Christian Wimmer, Thomas Würthinger, and Hanspeter Mössenböck. 2016. Bringing low-level languages to the JVM: efficient execution of LLVM IR on Truffle. In Proceedings of the 8th International Workshop on Virtual Machines and Intermediate Languages (VMIL 2016). ACM, New York, NY, USA, 6-15. DOI: https://doi.org/10.1145/2998415.2998416

[2] Stephen J. Fink and Feng Qian. 2003. Design, implementation and evaluation of adaptive recompilation with on-stack replacement. In Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization (CGO '03). IEEE Computer Society, Washington, DC, USA, 241-252.

[3] Thomas Würthinger, Christian Wimmer, Andreas Wöß, Lukas Stadler, Gilles Duboscq, Christian Humer, Gregor Richards, Doug Simon, and Mario Wolczko. 2013. One VM to rule them all. In Proceedings of the 2013 ACM international symposium on New ideas, new paradigms, and reflections on programming & software (Onward! 2013). ACM, New York, NY, USA, 187-204. DOI=http://dx.doi.org/10.1145/2509578.2509581

[4] https://graalvm.github.io/graal/truffle/javadoc/com/oracle/truffle/api/nodes/LoopNode.html