Bachelor’s Thesis

Partial Support for PThreads in Sulong

Student:	Florian Kraml
SKZ/Matr.Nr.:	521 / k01256528
Email:	floriankraml@gmail.com
Advisor:	DI Jacob Kreindl, Bsc.
Start Date:	Feb. 5, 2019

JOHANNES KEPLER
UNIVERSITY LINZ

DI Jacob Kreindl, Bsc.
Institute for System Software

P +43 732 2468 4370
F +43 732 2468 4345
jacob.kreindl@jku.at

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

Sulong [1,2] is an interpreter for LLVM IR, an intermediate representation of source code that can be produced by the Clang [3] compiler for the C family of programming languages. Implemented in Java, it is based on the Truffle [4,5] framework for implementing interpreters for programming languages and part of the GraalVM [6] project.

By targeting LLVM IR, Sulong is able to execute programs implemented in several popular programming languages for which an LLVM back-end is available, e.g. C, C++ and Fortran. Unfortunately, programs often use libraries that are only available as native binaries on users’ devices, e.g., system provided libraries. Sulong usually uses Truffle’s native function interface to call functions from such libraries directly. While this approach is known to work for many common libraries, certain libraries that wrap low-level system features cannot be supported that way.

One example of such a library is Pthreads [7,8], which provides the POSIX Thread API. It provides functions to create and manage threads and mutexes for synchronization between them. However, Sulong is a Java program running on the Java Virtual Machine (JVM). The JVM exposes thread management functionality to Java programs using a dedicated API. By circumventing this API, Sulong would risk corrupting the internal state of the JVM.

The goal of this thesis is to identify functions of Pthreads that can cause problems when invoked directly, intrinsify them in Sulong, and ensure that Sulong’s execution engine is thread-safe. The intrinsified functions should instead be implemented using Java’s own thread API for Java programs. As a result of this project, users should be able to execute programs that use common functions of Pthreads on Sulong.

The concrete goals of the thesis are as follows:

- Identify functions from the public API of Pthreads that are commonly used in real-world applications and can be implemented in Sulong using Java's concurrency API.
- Implement these functions in Sulong.
- Evaluate whether some functions of the public Pthreads API are safe to call directly and therefore need not be intrinsified.
- Identify and fix concurrency issues in Sulong's implementation.
- Select or implement a larger application that makes extensive use of libpthreads and demonstrate that Sulong is able to execute it correctly.
- Provide an extensive test-suite to demonstrate and verify the functionality of thread management using Pthreads in Sulong.

Optional goals for this project are as follows:

- Evaluate Sulong's run-time performance on real-world multi-threaded applications.
- Identify whether programs commonly use synchronization primitives not included in Pthreads and evaluate whether these are safe to call directly or need to be intrinsified as well.
- Also intrinsify these functions where necessary.

Explicit non-goals of this thesis are:

- Complete support for all features of Pthreads in Sulong.
- Find and/or fix issues in Sulong that are not related to concurrency.
    - However, any bugs of this category that are found should be reported.

The code written as part of this thesis is intended to be merged into the Sulong project, therefore it must meet the project's standards with respect to code quality, documentation and test coverage. The code must also be able to pass the Sulong project's requirements for being merged [9].

The progress of the thesis should be discussed with the adviser on at least a bi-weekly basis. The final version of the written thesis should be submitted before September 30, 2019.

[1] M. Rigger, R. Schatz, R. Mayrhofer, M. Grimmer, and H. Mössenböck, “Sulong, and Thanks for All the Bugs: Finding Errors in C Programs by Abstracting from the Native Execution Model,” in Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, New York, NY, USA, 2018, pp. 377–391.

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

[3] https://clang.llvm.org/

[4] T. Würthinger et al., “One VM to Rule Them All,” in Proceedings of the 2013 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software, New York, NY, USA, 2013, pp. 187–204.

[5] https://github.com/oracle/graal/tree/master/truffle

[6] https://www.graalvm.org/

[7] http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/pthread.h.html

[8] IEEE Std 1003.1c-1995

[9] https://github.com/oracle/graal/blob/master/sulong/docs/CONTRIBUTING.md