Jan Lehr | jan.lehr@sc.tu-darmstadt.de | Scientific Computing | S1|22 Alexanderstraße 2 | 64283 Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

SCIENTIFIC COMPUTING

Software Development for Compiler-Based Instrumentation

Compiler, Performance Analysis, High-Performance Computing, Software Engineering

Contact

Jan Lehr
jan.lehr@sc.tu-darmstadt.de

Scientific Computing
S1|22 Alexanderstraße 2
64283 Darmstadt

Date
August 29, 2017

Introduction

With the increasing complexity of hardware, detailed information about program behavior is crucial to determine whether a software uses the machine efficiently. However, obtaining this information can already be very challenging and an adventure on its own. Providing the developer or a specialized performance analyst with a set of configurable tools that allow her to adapt the measurement methodology to the needs at hand is of great interest. To this end, the InstRO framework has been developed at the group for Scientific Computing.

The framework facilitates compiler infrastructure to automatically generate instrumentation configurations suitable for both overview and detailed measurements. It is currently mainly built as a source-to-source translator, based on the ROSE compiler infrastructure. While the source-to-source capabilities are of great interest, the possibilities to include the instrumentation facilities into the actual compiler are an interesting and promising research direction.

General task description

You will implement program analyses for the InstRO framework with the Clang/LLVM compiler infrastructure. Based on the analyses' results, suitable instrumentation points are generated and inserted into the source code. Depending on your preference you will work on the source-to-source translator implementation or a tighter integration into the LLVM optimizer. The position is highly focused on implementation and software engineering.

What you will be doing

Implement	new analyses in Clang/LLVM to automatically generate suitable instrumentation.
Implement	transformation passes that allow for more efficient instrumentation.
Test	the implemented algorithms for correctness.
Evaluate	the quality of the implemented algorithms on both synthetic test-cases and real world applications.

Qualifications

Required	| Good command of C++11/C++14 | Working with git | Basic knowledge about compilers
Additional	| Parallel paradigms (OpenMP, MPI) | Good command of Python, Cmake, Autotools | Development in Linux environment | Clang/LLVM
Courses	| Compiler Tooling | Introduction to compiler construction | Systems and parallel programming | Architecture and design of computer systems