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

Binary Analysis for Instrumentation Influence Assessment

Compiler, Binary Analysis, Performance Analysis, High-Performance Computing

Contact

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

Scientific Computing
S1|22 Alexanderstraße 2
64283 Darmstadt

Date
August 29, 2017

Introduction

To carry out performance measurements, one available method is to augment code with additional measurement instructions, so called instrumentation. This is often done at the source level, which results in potential interaction with the compiler. The additional instructions, specifically function calls, can influence compiler decisions for optimization and code generation. Consequently, it is of interest to be able to determine the changes that occurred in the binary after the instrumentation was added. Theoretically, this change can be detected by performing binary comparison.

One of the occurring challenges is the comparison of two potentially different graphs. These graphs reflect the possible control flow paths through the program. Differently optimized programs lead to differently generated arrangements of the nodes of those graphs, referred to as basic blocks of the control-flow graph (CFG).

General task description

We develop a framework that is used to validate measurement results obtained with a particular binary. Currently, the framework applies simple baseline comparison of recorded runtime metrics. In our research we want to explore the possibilities to indicate perturbation and measurement influence statically. Therefore, you would implement CFG-level executable comparison within an existing Python-based framework. The position includes both literature research and implementation, with the focus strongly on implementation.

What you will be doing

Evaluate the available literature for usefulness in the given context.

Implement the algorithms found in literature in the given Python framework.

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 Python | Working with git | Interest in HPC | Willingness to read Assembler

Additional | Knowledge of compilers | Good command of C++ | Development in Linux environment

Courses | Introduction to compiler construction | Systems and parallel programming | Architecture and design of computer systems