Compiler tool development for statement merging on a function-level

Applicable for students as HiWi, B.Sc. and M.Sc. Thesis
Keywords: Clang, Source Analysis, Source Transformation, Data-flow tracking

Introduction

Modern (numeric) C++ tools use expression templates for efficient code generation and to avoid, e.g., the creation of temporary objects when evaluating a statement. In a sense, long statements appear to be more optimal for expression templates.

Task

A Clang-based tool should be developed which, based on data-flow restrictions, automatically merges statements on a function-level. The strategy aims to make the statements as long as possible without too many re-computations.

1 double foo(double a, double b) {
2     double phi = a * b;
3     double phi_s = phi * .5;
4     return phi_s;
5 }

Figure 1: Left: Input function with two statements computing a value. Right: \( \phi \) can be merged into the \( \phi_s \) statement.

What you will be doing

(a) The Clang tool parses a function and generates a computational graph containing relevant values w.r.t. expression templates. The graph contains as nodes function-local values, and as edges the data flow between these values.

(b) The tool analyzes the graph to determine values that can be merged without breaking the existing semantics.

(c) It merges previously detected statements and expressions.

(d) Finally, you evaluate the approach w.r.t. performance and compiler impact.

Qualifications

• Experience with modern C++ and the CMake build system.
• Basic knowledge of compilers and related techniques.
• Knowledge of the Clang tooling library [1].
• Optional: Experience with graph analysis.

References

[1] https://clang.llvm.org/docs/LibTooling.html

Alexander Hück
alexander.hueck@tu-darmstadt.de
Office: SII22 Room 412
Alexanderstraße 2
64287 Darmstadt
Tel. 06151 16-755 77

Date: 18th June, 2020