
Eötvös Loránd Tudományegyetem
Faculty of Informatics
Software Engineering BSc

# Improving Performance of C++ Programs with Static Analysis

**Gábor Horváth**
Computer Science MSc

**Barnabás Bittner**
Software Engineering BSc

Budapest, 2017

# Contents

# 1 Introduction

Static analysis is the analysis of computer software without actually compiling and executing it as opposed to dynamic analysis[1] which involves running the application.

# 2 Technical background

In this section I will outline pieces of technology related to the main goal of this thesis. Starting from the global framework I'm integrating modules into to the abstract representation of a C++ application.

## 2.1 LLVM

LLVM formerly known as Low Level Virtual Machine is a "collection of the reusable and modular compiler technologies"[2]. LLVM started out as a university project[3], and since then it grew significantly in size and it now offers numerous subprojects which help building and maintining both commercial and open-source applications.
The essential goal of LLVM is to provide generalized optimizations to arbitrary programming languages using the LLVM Intermediate Language also known as LLVM IR, which acts as a common representation of different programming languages. This is achieved through using specific language front-ends, which transforms the given language to LLVM IR.

## 2.2 Clang

The most popular C/C++ familiy compiler front-end for LLVM is Clang which aims to excel from the open-source compilers with it exceptionally fast compile-times and user-friendly diagnostic messages.

## 2.3 Clang-tidy

## 2.4 Abstract syntax tree - AST

# 3 References


[1] D. C. L. W. N. W. BA Wichmann, AA. Canning and D. Marsh, "ndustrial perspective on static analysis," *Software Engineering Journal*, 1995. https://web.archive.org/web/20110927010304/http://www.ida.liu.se/~TDDC90/papers/industrial95.pdf.

[2] "The llvm compiler infrastructure," http://llvm.org/.

[3] C. Lattner and V. Adve, "LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation," in *Proceedings of the 2004 International Symposium on Code Generation and Optimization (CGO'04)*, (Palo Alto, California), Mar 2004. http://llvm.org/pubs/2004-01-30-CGO-LLVM.html.