

Benefits of the LLVM compiler infrastructure

LLVM (Low-Level Virtual Machine) back-end in co-operation with LLVM 'native' Clang front-end can compile e.g. C and C++ code faster in most cases than old and somewhat stagnant GCC compiler. Especially time needed for parsing, type checking, semantic analysis and building AST (Abstract Syntax Tree) seem to be reduced drastically. In addition lower memory use is achieved and generated executable code runs faster and is smaller in size. Therefore we can conclude that LLVM speeds up both developers and end-users work.

LLVM infrastructure is highly modular. The benefits are that e.g. linker, optimizer, code generator, assembler and archiver are separate tools in the toolchain. This makes implementing further improvements to them easier. LLVM's modular design also features compile-time and link-time optimization.

Due to modular architecture LLVM infrastructure also consists of LLVM-GCC front-end which features GCC (GNU Compiler Collection) front-end but uses LLVM optimizer and code generator. Both Clang and LLVM-GCC front-ends generate object files that are compatible with object files compiled with GCC. The added benefit of LLVM-GCC is that it supports more programming languages than Clang but the disadvantage is that it is not quite as fast as Clang. Clang has also many impressive features such as expressive diagnostics.

The LLVM's highly modular code library is easier to understand and maintain than GCC's. This allows creation of new compilers e.g. to support other languages. One of the reasons for the simpler codebase is that LLVM is still rather new compiler architecture compared to GCC which codebase has become very large during the years and continuously more complex. The LLVM codebase is also well documented.

The reusable codebase has allowed creation of many interesting projects. One of these is Clang static code analyzer build on top of LLVM and Clang. It analyzes source code in order to find bugs. The tight IDE integration allows visual means to display suggestions how to improve code. Another great project is LLDB which is a new debugger which is faster than GDB.

LLVM is likely to have a bright future because it is modern, efficient and has an easily hackable and reusable codebase. The many benefits of its will lure more and more developers and companies to use it. Applications that already benefit from LLVM are e.g. separate implementations of OpenGL and OpenCL which performance is improved using LLVM's JIT (Just-In-Time compiler).