Hessisches Kompetenzzentrum
für Hochleistungsrechnen

exaFOAM - Performance Analysis and Modeling of OpenFOAM

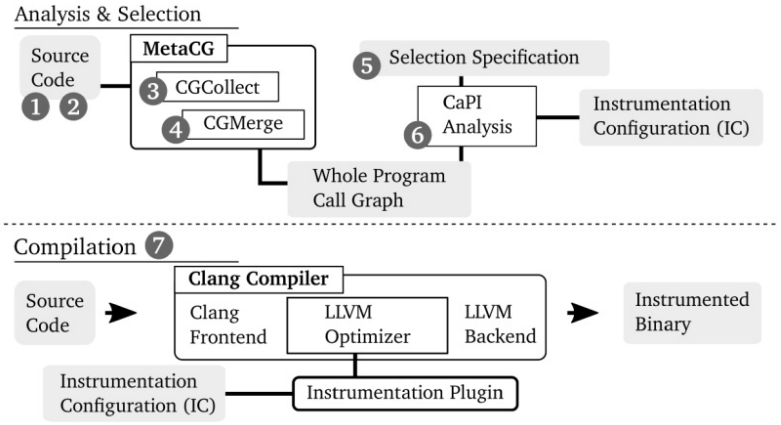

Figure 1: Our instrumentation toolchain consists of the following steps: (1) Preparation of the target code's build system. (2) Generation of a compilation data base for Clang-based tools. (3) Translation-unit local CG construction, given the MetaCG workflow. (4) Whole-program CG construction, manually combining relevant sourc files. (5) Definition of the selection specification. (6) Execution of the CaPI analysis to create the IC. (7) Compilation of target code with IC instrumentation.

Project Manager
Sebastian Kreutzer

Principal Investigator
Dr. Christian Iwainsky

Project Term
2021 - 2022

Clusters
Lichtenberg Cluster Darmstadt

Software
OpenFOAM

Institute
Institute of Scientific Computing

University
Technische Universität Darmstadt

Introduction

The exaFOAM project (https://exafoam.eu) aims at overcoming the current limitations of Computational Fluid Dynamics (CFD) technology, especially in what concerns the exploitation of massively parallel HPC architectures. This involves the development and validation of a range of algorithmic improvements, across the entire CFD process chain (preprocessing, simulation, I/O, post-processing). The performance assessment of both existing and new code is essential in order to find scalability issues and guide further development. To this end, we are developing specialized performance measurement tooling, aimed at large-scale scientific codes. The use of HPC resources is vital in the assessment of these new tools and their application on OpenFOAM benchmark cases.

Methods

As part of this project, we are working with OpenFOAM version v2106. The main goals of the first project period was an initial performance analysis of newly developed benchmark cases, designed to mirror the behavior of larger-scale industrial use cases. For this, we employed the Score-P tool suite, which enables the generation of detailed profiles of the investigated programs. Score-P requires the insertion of profiling probes into the code by the compiler, a process called instrumentation. Since the use of too many probes can degrade the program's performance significantly and thus skew results, we have

developed static analysis based tooling in order to select only relevant parts of the code for instrumentation. This newly developed tool, based on principles of the existing InstRO project, uses the Clang/LLVM compiler infrastructure for analysis and code transformation. We combined this technology with the use of the performance modeling tool Extra-P, which was used to find potential scalability issues.

Results

We have developed a new instrumentation tool called CaPI (https://github.com/tudasc/CaPI). This tool employs a domain-specific language that can be used to specify the selection of instrumented code sections in concise manner. For the application of OpenFOAM, we have implemented specialized selection modules, used for the detection of computational kernels and MPI communication. Our evaluation showed that this tool is effective in eliminated excessive measurement overhead. We have used this newly developed technology to perform initial performance assessment of the exaFOAM benchmark cases, which we expect to publish at a later date.

Discussion

We have successfully developed a tool that aids in the performance analysis of large-scale scientific codes. We are in the process of improving this tool further, by addressing usability issues and extending the use of static analysis capabilities. While the application of CaPI was currently limited to smaller benchmark codes, we hope to employ it for larger-scale experiments using the exaFOAM industrial use cases. Furthermore, we hope to run more extensive performance modeling experiments in order to better assess the exascale readiness of OpenFOAM.

Publications

Kreutzer, S.; Iwainsky, C.; Lehr, J.P.; Bischof, C.: ”Compiler-assisted Instrumentation Selection for Large-scale C++ Codes”, C3PO’22 Workshop - Third Workshop on Compiler-Assisted Correctness Checking and Performance Optimization for HPC, Jun 2022, Hamburg, Germany https://tubiblio.ulb.tu-darmstadt.de/135961/

Kreutzer, S.: ”Using link-time call graph embedding to streamline compiler-assisted instrumentation selection”, May 2022, London, UK, https://llvm.org/devmtg/2022-05/

Last Update: 2023-03-16 01:26