The Journal of Open Source Software

The Glasgow Fortran Source-to-Source Compiler

Wim Vanderbauwhede[1]

1 University of Glasgow School of Computing Science

DOI: 10.21105/joss.00865

Software

- Review
- Repository
- Archive

Submitted: 23 July 2018
Published: 12 December 2018

Summary

The Glasgow Fortran Source-to-Source Compiler is a source-to-source compiler that automatically refactors legacy FORTRAN77 code into modern and maintainable Fortran-95 code that is ready for acceleration with OpenCL, OpenMP, OpenACC etc. This is the main purpose of the compiler. It also allows to translate code into C and the OpenCL kernel language to further assist with acceleration of code, and is an essential component of our toolchain for automatic parallelization of Fortran code using OpenCL (Vanderbauwhede & Davidson, 2018).

The compiler is entirely self-contained and written in Perl.

The compiler is intended to work on typical scientific simulation code, but it works on the complete NIST FORTRAN78 test suite.

Our compiler successfully generates refactored code for *all* tests, and the refactored code compiles correctly and passes all tests (2887 tests in total). The tests are available in the repository under `tests/NIST_F78_test_suite`.

Furthermore, we tested the compiler on four real-word physics simulation models:

- The 2-D Shallow Water example from the book "Ocean Modelling for Beginners: Using Open-Source Software" by Jochen Kämpf. (188 loc), available in `tests/ShallowWater2D`.
- The Large Eddy Simulator, a high-resolution turbulent flow model (1,391 loc)
- The shallow water component of Gmodel, an ocean model (1,533 loc)
- Flexpart-WRF, a version of the Flexpart particle dispersion simulator that takes input data from WRF (13,829 loc)
- The Linear Baroclinic Model, an atmospheric climate model (39,336 loc)

Each of these models has a different coding style, specifically in terms of the use of common blocks, include files, etc that affect the refactoring process. All of these codes are refactored fully automatically without changes to the original code and build and run correctly. The performance of the original and refactored code is the same in all cases.

References

Vanderbauwhede, W., & Davidson, G. (2018). Domain-specific acceleration and auto-parallelization of legacy scientific code in fortran 77 using source-to-source compilation. *Computers & Fluids*. doi:10.1016/j.compfluid.2018.06.005