An Automated Fortran Code Refactoring Tool to Facilitate Acceleration of Numerical Simulations

Wim Vanderbauwhede
School of Computing Science, University of Glasgow, UK

University of Glasgow

Outline

- Background
- This work
- How could this is be useful for you?
- Features
- How it works
- Status
- Practicalities
- Further work

Background

- Most code for numerical simulation of weather/climate models is written in Fortran
- In particular the older codebases, written in F77, are difficult to maintain or modify
- With the advent of multicore CPUs and GPGPUs and technologies such as OpenMP and OpenCL, there is a growing interest in acceleration of numerical simulations
- With the current state of affairs, this usually requires a considerable manual rewrite of the code.

University of Glasgow

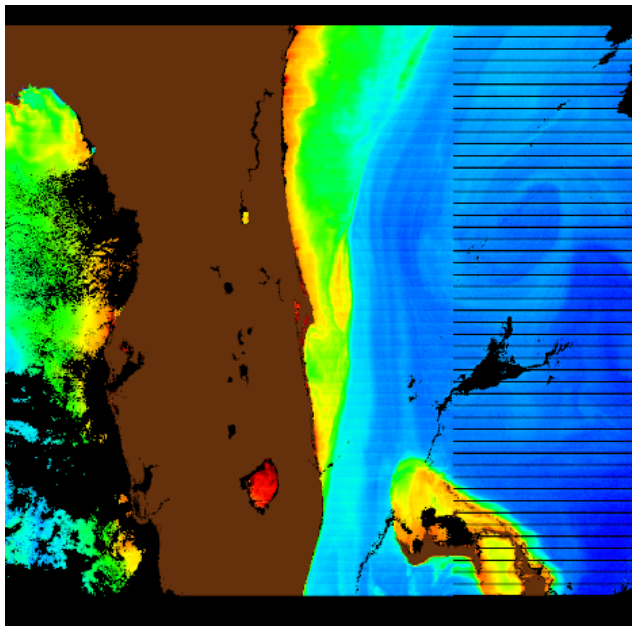
This work

- We have created an automatic refactoring tool for Fortran code
- Mainly useful for F77, but works for F95 too
- This tool converts F77 into F95
- It also supports annotations to extract subroutines and translate parts of the code to C

University of Glasgow

How could this is help you?

- Are you using F77 code?
- Would you like to make it run faster?
- Would you like to upgrade to F95?
- But you don't have the time or the skill?



University of Glasgow

Features

- F77 to F95 translation: support for
  - F95 syntax, including types, parameters and operators
  - modules
  - INTENT attributes
  - proper DO ... END DO -loops
  - Preserves comments
- Subroutine extraction
  - simply add an annotation

    ```
    !$ACC SUBROUTINE particles_main_loop
    ```

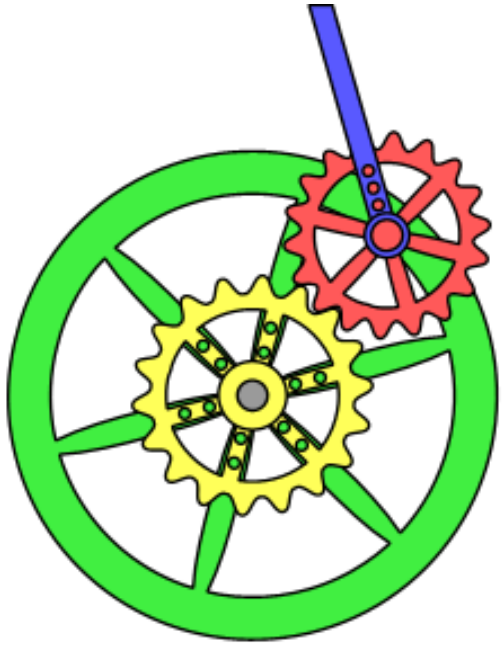
  - Extracted subroutine and any routine in its call tree are refactored so that there are no shared global variables
- C translation
  - We use Mark Govett's *F2C_ACC* for C translation
  - You simply annotate any subroutine or function to be translated:

    ```
    !$ACC TRANSLATE C
    ```

- Call graph generation

How it works

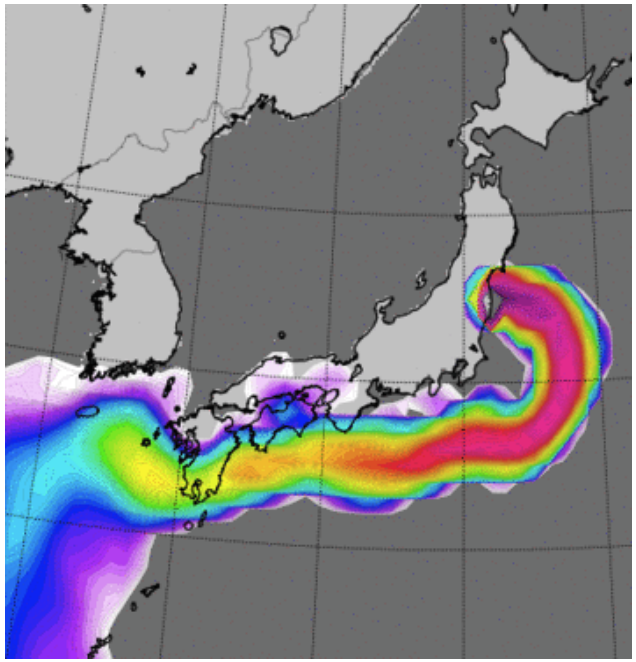
- Inventory of source code by following function/subroutine call tree and include statements
- Source-to-source compiler
- Line-by-line parse, recursive descent
- Analysis of global (/COMMON/) variables and parameters
- Loop analysis to transform GOTOs and labeled DO into DO ... END DO
- IO direction analysis to determine INTENT



University of Glasgow

Status

- Tested refactoring and subroutine extraction on **Flexpart-WRF**, a Lagrangian Particle Dispersion Model.
  - Fortran 77, 103 files, 13578 lines of code
  - Automatically refactored in **3s**
  - Refactored code: Fortran 95, 103 files, 13893 lines of code
  - Extracted subroutine particles_main_loop(), refactored all its children
- Tested the call graph generation with **WRF v3.4**
  - Fortran 90, 613 files, 544394 lines of code
  - Call graph generated in **20s**



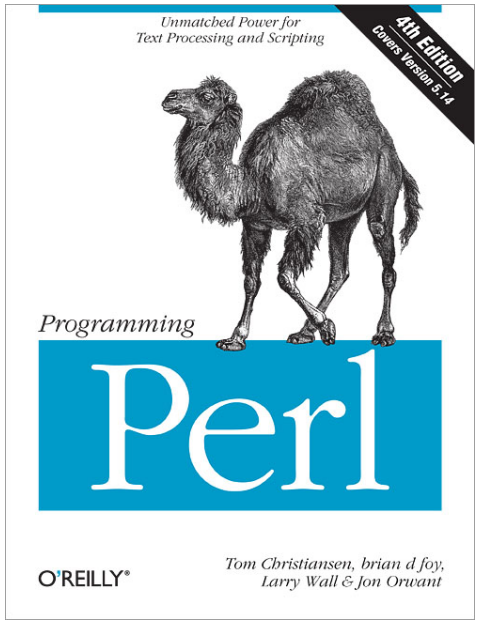
Practicalities

- Code is written in Perl (requires v5.8), easy to install, no dependencies, no compilation
  - See the INSTALL file
- Works on Linux, OS X and similar systems

```
$ refactorF4ACC.pl flexpart_wrf
```

- Available on GitHub

```
git://github.com/wimvanderbauwhede/RefactorF4Acc.git
```



Further work

- More advanced module and interface support
- Full integration with C translation
- Automatic OpenMP or OpenACC annotations