Joerg Henrichs (Bureau of Meteorology)

Title: Using PSyclone in Ocean Modelling – The Evolutionary Way

Chris Dearden, Rupwer W. Ford, Joerg Henrichs, Andrew Porter, Sergi Siso, Simon Mueller

Abstract:

PSyclone, a tool for code generation and transformation (https://github.com/stfc/PSyclone), is being developed with the UK Met Office to address challenges in the development of modern NWP models. PSyclone focuses on the three Ps:

Portability – using a single source science code from which code that works on different hardware architectures can be created,

Performance – create optimised code for different hardware architectures, and

Productivity – make it easier for natural scientists to write code for large scale parallel machines.

The driving force behind the development of PSyclone is the separation of concerns: a natural scientist should be able to keep on developing code, while independently, an HPC specialist can work on optimising the code for various platforms. PSyclone achieves this by a set of transformations which can be applied to the single source science code, so that optimised code for a specific system is created. Using transformations, PSyclone can create distributed memory versions of LFRic or add OpenMP directives for threading for example. The set of transformations can be developed by an HPC expert, while science development can continue independently.

PSyclone was originally designed for the Met Office's next generation NWP model LFRic. Its revolutionary approach requires that either the simulation code be designed from scratch or an investment to be made in refactoring of existing code. However, PSyclone also offers an evolutionary approach: it can read in existing Fortran code, apply transformations to the source code, and write out optimised Fortran source code.

In this presentation I will be showing the usage of PSyclone as an evolutionary tool: how to use it with a large, existing source code base to automatically create optimised code. Using the NEMO Ocean model as example, I will show how we created a version of NEMO that is able to run on GPUs by adding OpenACC directives with only minor source code changes, including current performance results. We will also discuss lessons learned in this process.