Loki v0.1.2: A Source-to-Source Translation Tool for Numerical Weather Prediction Codes and more

Michael Staneker, Ahmad Nawab, Balthasar Reuter, Michael Lange
Research Department, ECMWF, Reading (UK) and Bonn (Germany)
{firstname}.{lastname}@ecmwf.int

Motivation

- Achieve **perfomance portability** across a broad range of HPC architectures including **accelerators** (such as GPUs) from a **single code base**
- Perform **static code analysis/linting** on Fortran source code

Challenge

- Different **programming paradigms** and environments
- **Hardware-specific** optimization (loop order, memory layout, ...)
- **Large** and **complex legacy code** needs to be adapted

Solution

Develop **source-to-source translation** tool for Fortran source code to:

- Apply bespoke transformation recipes
- Perform static code analysis

Open development on Github

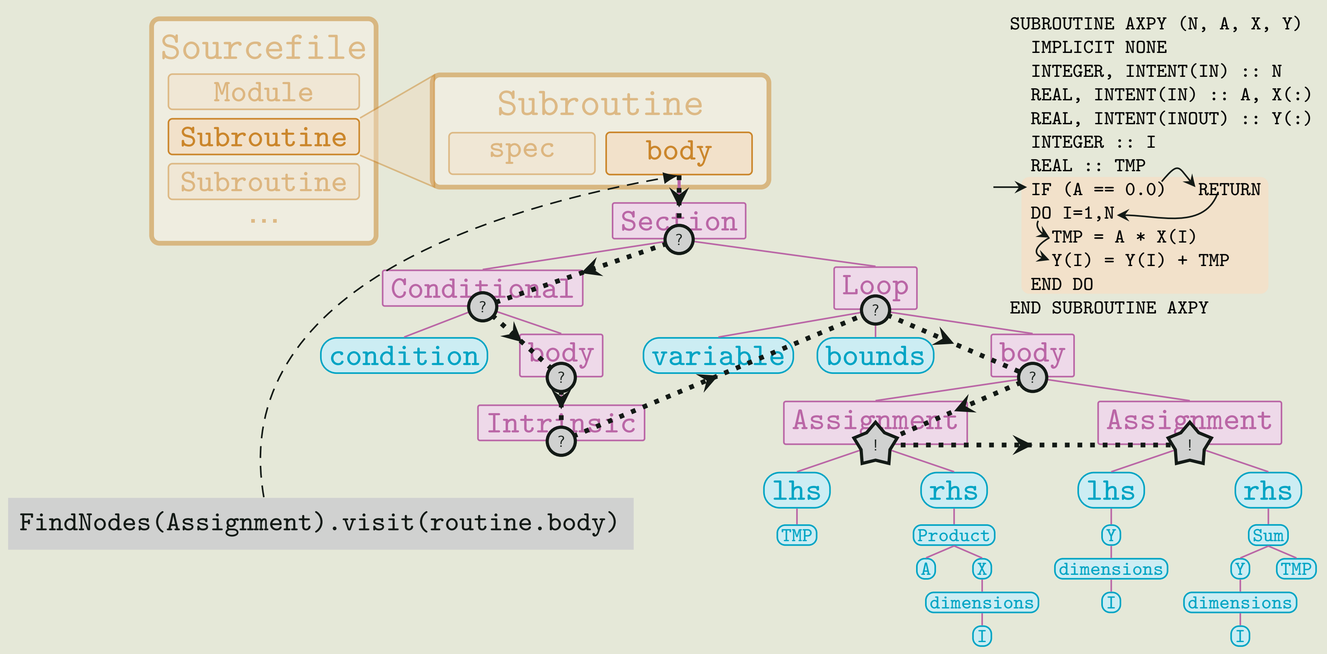
Source code & bug tracker
github.com/ecmwf-ifs/loki

Documentation
sites.ecmwf.int/docs/loki/

Jupyter Notebook Tutorials
github.com/ecmwf-ifs/loki/tree/main/example

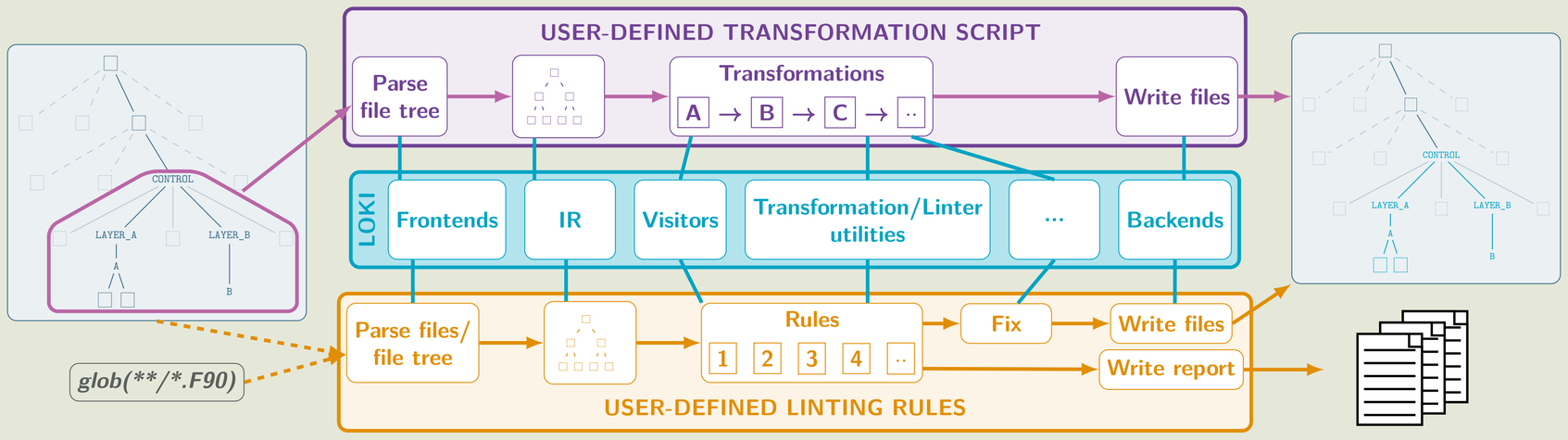
Loki: overview and internal representation

- Loki is a Python package to develop **source-to-source translation recipes** for Fortran codes, such as the IFS
- It offers an API to **encode custom transformations or analysis/linting pipelines**
- Loki uses *FParser*[1] to obtain parse tree of Fortran source files
- It then builds a bespoke two-level **internal representation (IR)** of the parsed code based on abstract syntax trees:
  - Custom control flow tree (Fortran-tinted)
  - Expression trees based on *Pymbolic*[2]
- Loki **manages type information** in the changing IR tree
- The Loki IR is **traversed** and **transformed** using **visitors**
- Finally, **multiple backends** can be used to generate Fortran, C, Python or CUDA-Fortran code



Typical Loki transformation and/or linting pipeline

- **Loki scheduler** can be used to batch process large file trees whilst ensuring any dependencies are respected
- **Multiple** transformations or linting rules can be easily combined into **pipelines**
- Loki's **IR** also allows for **interprocedural analysis**, thereby unlocking highly sophisticated linting/analysis capabilities

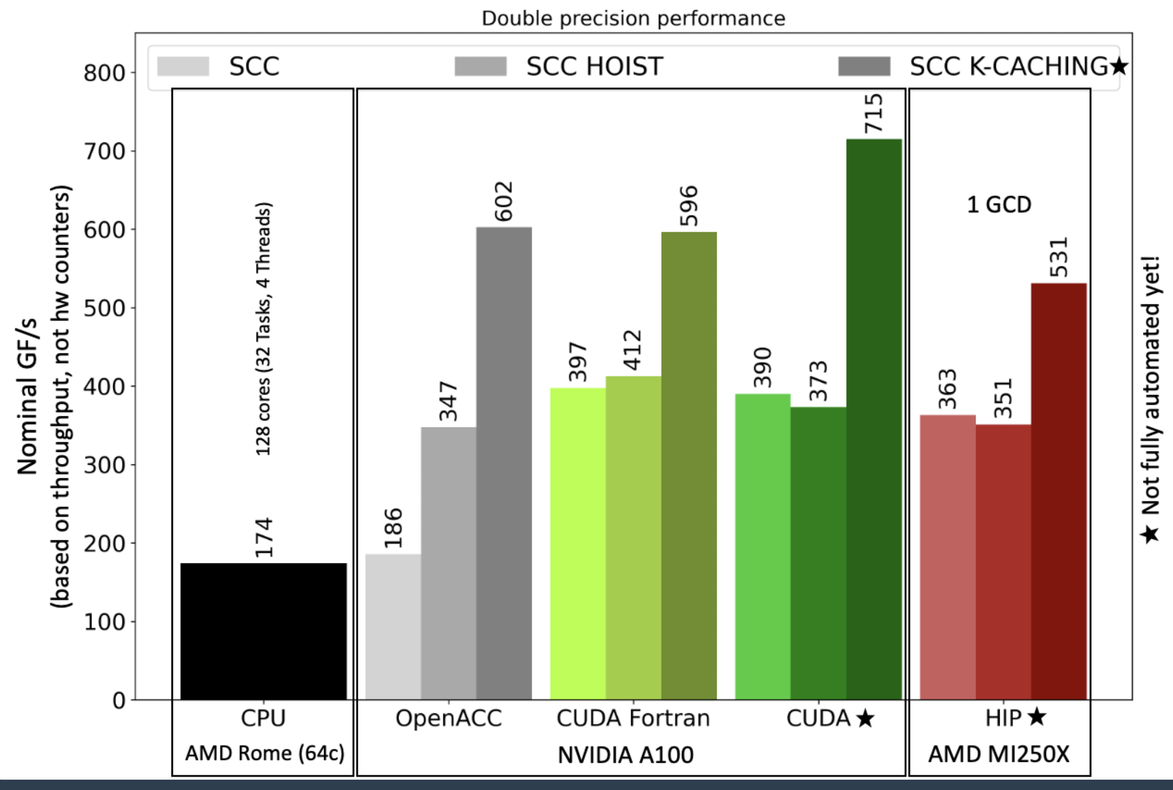


CLOUDSC: cloud microphysics dwarf on GPU

- Representative for class of single-column algorithms, thus proxy for parts of the IFS
- **SCC**: Single-Column Coalesced transformation, loop flip and array demotion for SIMT layout

github.com/ecmwf-ifs/dwarf-p-cloudsc

- **SCC HOIST**: Pre-allocate temporary device arrays in driver
- **SCC K-CACHING**★: manual loop fusion and further array demotion



User base

- The user base for Loki is growing rapidly as it continues to be adopted more widely throughout **ECMWF** and **Météo-France**
- Loki is a key component for **Destination Earth**

stories.ecmwf.int/destination-earth

Outlook and Plans

- Further develop transformations for **reducing memory allocations** for temporaries on GPUs, e.g. hoisting, pool allocators, etc
- **Fully automate Fortran to C transpilation** to unlock C-based programming environments, e.g. CUDA, HIP, SYCL, etc
- More sophisticated **data flow analysis** capabilities
- Fully automate **highly performant k-caching** transformation
- Extend transformation recipes to kernels with **horizontal communication i.e. stencils**
- **Enable parallel execution** of Loki frontend and transformation/linting utilities

1. Science and Technology Facilities Council. *fparser.* https://github.com/stfc/fparser.
2. Klöckner, A. & Contributors. *Pymbolic.* https://github.com/inducer/pymbolic.

The work presented in this poster has been produced in the context of the European Unions Destination Earth Initiative and relates to tasks entrusted by the European Union to the European Centre for Medium-Range Weather Forecasts implementing part of this Initiative with funding by the European Union.