Seeking portability and productivity for NWP model code with GT4Py

Christian Kühnlein (ECMWF)

Till Ehrengruber (CSCS), Stefano Ubbiali, Nicolai Krieger, Lukas Papritz, Alexandru Calotoiu, Heini Wernli (all ETH Zurich)

20th ECMWF Workshop on HPC in Meteorology

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Funded by the
European Union

ECMWF

Platform for Advanced Scientific Computing

ETH zürich

Seeking portability and productivity

- Emerging technologies offer great potential for higher numerical resolution and energy efficiency. At the same time, ESMs face an increasingly diverse landscape of supercomputing architectures.
- Efficient execution requires targeted hardware-specific optimization. Serving various hardware inevitably involves more complex code that needs to be organized to maintain productivity.

Two streams of development at **ECMWF**:

Operational IFS: Main ECMWF scalability & portability efforts prepare the spectral-transform forecast model for hybrid CPU+GPU execution. Automatic code translation tools are developed and employed, accompanied by restructuring core model components and various technical infrastructure packages. Fortran is largely maintained for the time being and hybrid execution is enabled mostly by means of directives/pragmas.

Dynamical core for future IFS: We rewrite (from initial Fortran code) and further develop forecast model in Python with the domain-specific library GT4Py. The forecast model is building on finite-volume non-hydrostatic dynamical core FVM with the IFS physical parametrizations. This project happens in close collaboration with partners at CSCS and ETH Zurich.

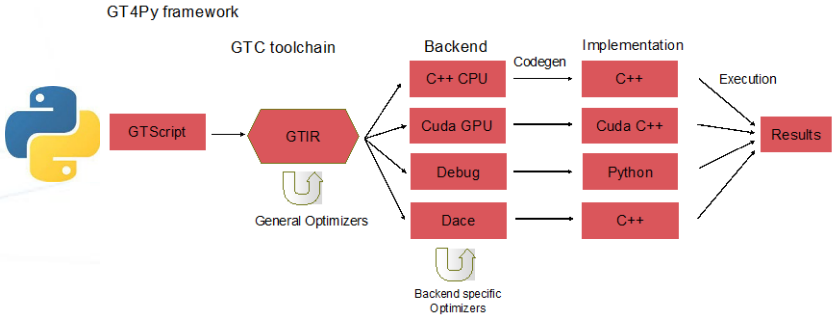
ECMWF CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre Platform for Advanced Scientific Computing ETH zürich esiwace

GT4Py domain-specific library

- https://github.com/GridTools/gt4py (public, open source) and see Afanasyev et al. 2021, Ben-Hun et al. 2022

- GT4Py works as an optimizing compiler for multiple backends:
  - Code generation optimized for a specific architecture
  - Backend selects HPC implementation strategy (e.g., parallelization, memory layout, etc)
  - Backends can be added to provide efficient implementations for new technologies / architectures
  - Leverages knowledge of the typical computation patterns in the domain

- GT4Py is embedded in the **Python** eco-system
  - Established, portable and productive
  - Broad and comprehensive selection of modules and libraries
  - Favourable choice with respect to AI applications
  - Low barrier of entry for domain scientists and academia



- Two versions of GT4Py

V1: gt4py.cartesian: 3D structured (I, J, K)

V2: gt4py.next: 2D horizontally unstructured and vertically structured (IJ, K)

GT4Py domain-specific library

- Three comprehensive GT4Py based NWP and climate model software development projects (originating from Fortran implementations that were optimized for CPUs):

- **Pace** is a GT4Py (V1) implementation of the **FV3GFS / SHiELD** atmospheric model of NOAA and GFDL by Allen Institute for AI (AI2) with ETH Zurich and CSCS. → See Oliver Elbert's talk

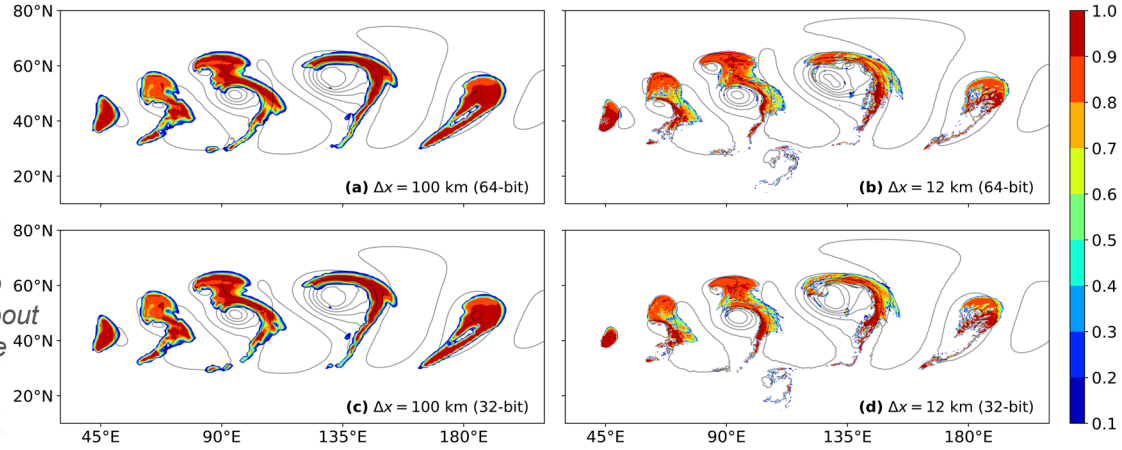
- **ICON** atmospheric model dynamics and physics incrementally ported to declarative GT4Py (V2) by MeteoSwiss, EXCLAIM project at ETH Zurich and CSCS.

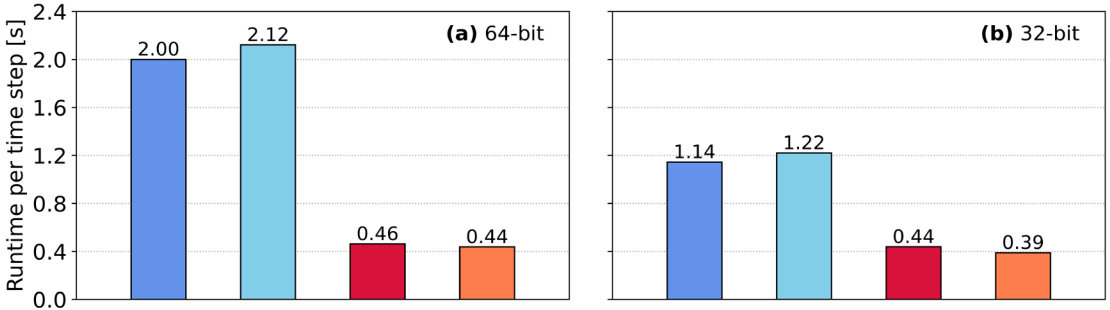
- **FVM for IFS** porting and further development with GT4Py (V1+V2) by ECMWF, CSCS, and PASC-funded project KILOS at ETH Zurich.

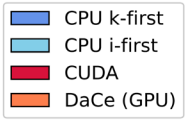
Some results with 3D structured-grid model using GT4Py V1

As with the original Fortran FVM code, we can run the GT4Py based FVM with either 64-bit or 32-bit precision.

Results for DCMIP2016 moist baroclinic wave with dynamical core (nearly-global configuration) coupled to ECMWF cloud scheme. Shown are cloud fraction at about 2 km above the surface (shading) and surface pressure (contour levels with 10 hPa interval) at day 10.





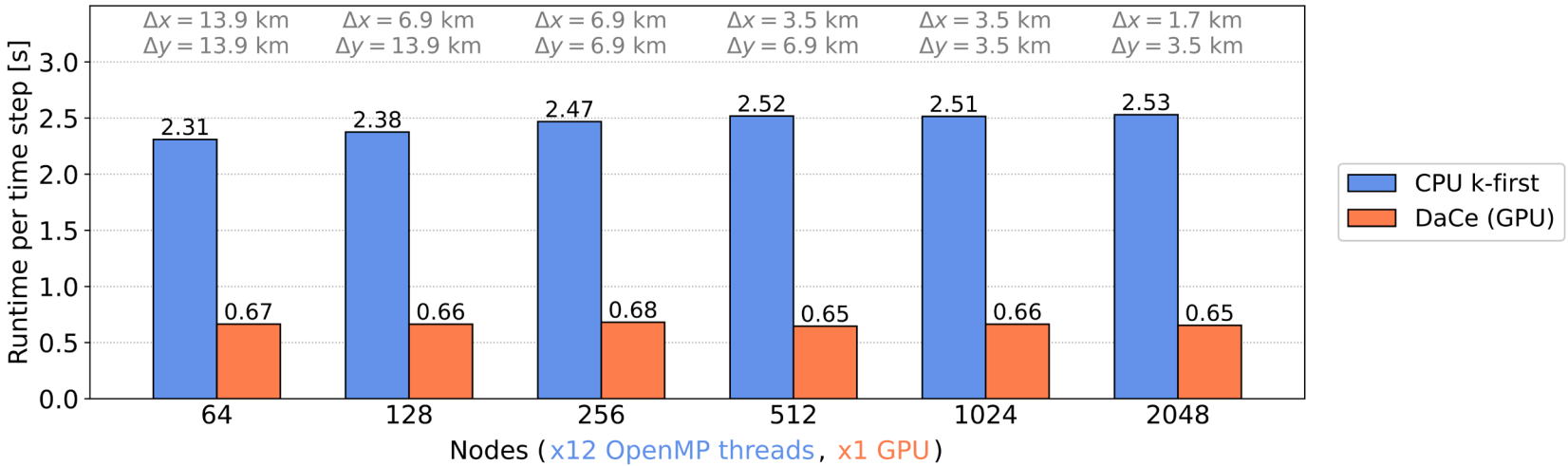


DaCe (GPU) leveraging the Data-Centric Parallel Programming Framework (DaCe, Ben-Hun et al. 2019, 2022). See https://github.com/spcl/dace

Runtime measures on CSCS Piz Daint supercomputer show increased computational performance with 32-bit precision. Expected speed-up from 64-bit to 32-bit on CPUs, but very little improvement with GPUs here.

ECMWF CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre Platform for Advanced Scientific Computing ETH zürich esiwace

Some results with 3D structured-grid model using GT4Py V1



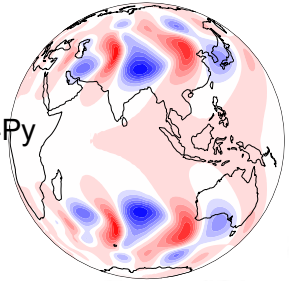
- **Weak scaling of structured-grid FVM** nearly-global configuration (latitude +-80deg) coupled to IFS cloud scheme.
- Here we test scaling across the CPU or the GPU partitions of CSCS' Piz Daint supercomputer, results on selected EuroHPC supercomputers with different GPUs (e.g. LUMI, Meluxina) will be available soon
- Halo exchanges based on GHEX -- Generic Exascale-ready Library for Halo-Exchange Operations – with Python bindings. GHEX is developed at CSCS and supported by PRACE – Partnership for Advanced Computing in Europe.

ECMWF CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre PASC Platform for Advanced Scientific Computing ETH zürich esiwace

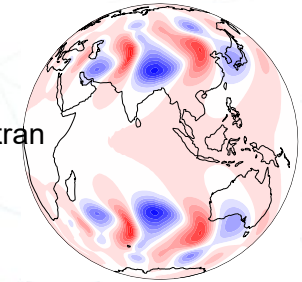
Portable global FVM dynamical core based on new declarative GT4Py V2

Baroclinic wave on day 8

GT4Py

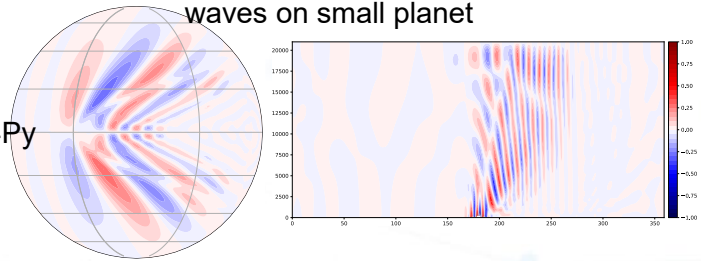


Fortran

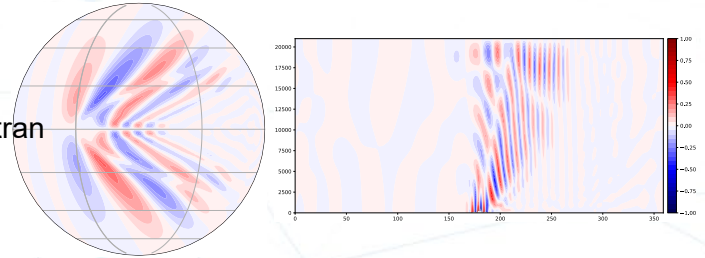


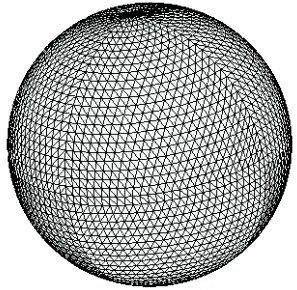
Non-hydrostatic mountain waves on small planet

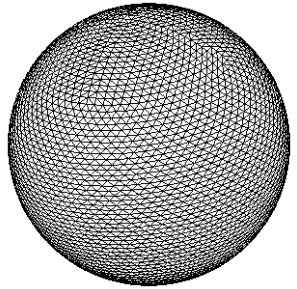
GT4Py



Fortran





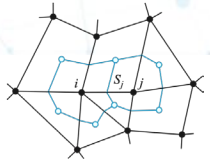


```python
from atlas4py import StructuredGrid
grid = StructuredGrid("O24")
mesh = AtlasMesh.generate(grid)
```

```python
from atlas4py import StructuredGrid
grid = StructuredGrid("H18")
mesh = AtlasMesh.generate(grid)
```

Python bindings for Atlas

GT4Py Field Operator example:




```python
@field_operator
def advection_scheme_upwind(
    rho: Field[[Vertex], float],
    dt: float,
    vel: tuple[Field[[Vertex], float], Field[[Vertex], float]],
    vol: Field[[Vertex], float],
    dual_face_orientation: Field[[Vertex, V2EDim], float],
    dual_face_normal: tuple[Field[[Edge], float], Field[[Edge], float]],
    dual_face_length: Field[[Edge], float]
) -> Field[[Vertex], float]:
    flux = upwind_flux(rho, vel, dual_face_normal, dual_face_length)
    return rho - (dt / vol) * neighbor_sum(
        flux(V2E) * dual_face_orientation, axis=V2EDim)
```

- Global non-hydrostatic dynamical core for the first time implemented in Python with new declarative GT4Py (runs on single compute node with shared memory, GPU or CPU)
- Further consolidation, extension and optimization of operators and toolchain necessary
- Distributed high-performance global dycore implementation based on declarative GT4Py in 2024. This will enable dycore computational comparison study against H and NH IFS on GPU and CPU based supercomputers.

ECMWF

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

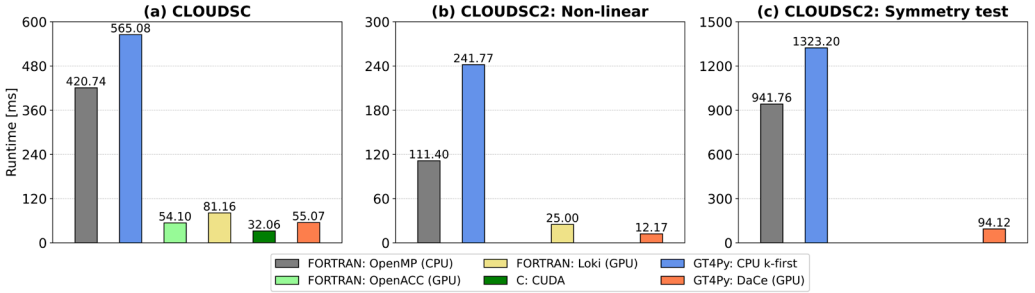
Platform for Advanced Scientific Computing

ETH zürich

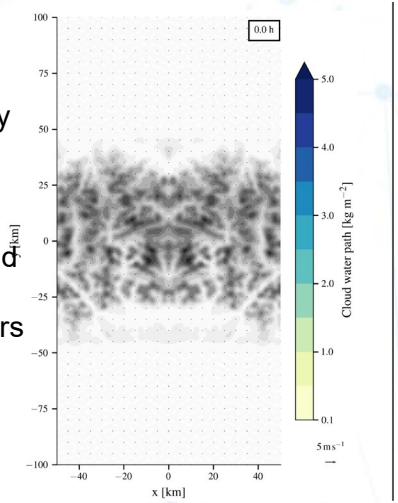
esiwace

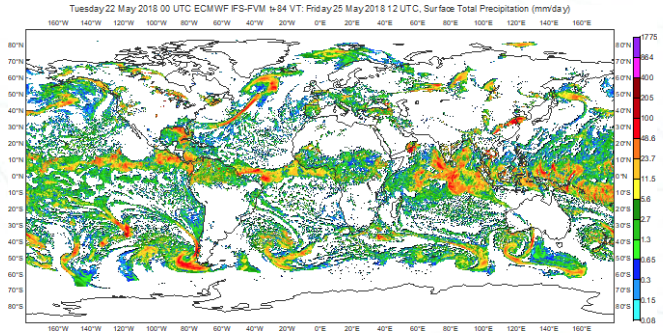
Further topics and outlook

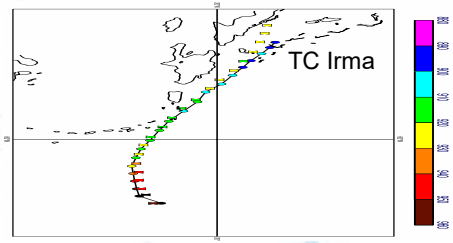
- GPU porting of IFS physical parametrizations and exploring tangent-linear/adjoint code with GT4Py; prototype using ECMWF microphysics and testing on various computing systems (Ubbiali et al. in prep 2023). To be extended to other parametrization schemes.



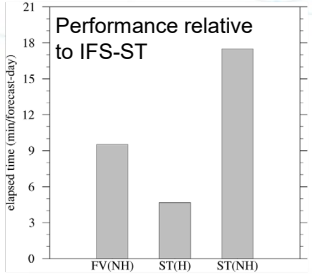
- A portable large-eddy simulation model based on Python+GT4Py for research at ETHZ and is of interest to member state partners in the ACCORD consortium







Previous FVM Fortran results



- Working towards portable high-performance Python+GT4Py global non-hydrostatic model with advanced numerics and re-establishing real forecast configuration

ECMWF
CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre
Platform for Advanced Scientific Computing
ETH zürich
esiwace

Thank you for listening!

ECMWF

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Platform for Advanced Scientific Computing

ETH zürich

esiwace