Seeking portability and productivity for NWP model code

Christian Kühnlein (ECMWF)

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

NOAA Weather Forecasting Office, Unifying Innovations in Forecasting Capabilities Workshop 2023
Keynote: Emerging Technologies & Opportunities: GPU and Earth System Modeling

ECMWF

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Platform for Advanced Scientific Computing

ETH zürich

UIFCW 2023
A UFS Collaboration Powered by EPIC

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101093054.

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 parallel 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 and GPU execution is enabled mostly by means of OpenACC directives.

Future IFS with new dynamical core: We are rewriting (from existing Fortran) and further developing the forecast model in Python with the domain-specific library GT4Py, in close collaboration with partners at CSCS and ETH Zurich. The forecast model is building on finite-volume non-hydrostatic dynamical core with the IFS physical parametrizations.

GT4Py domain-specific library

❑ Software programming implementation using the domain-specific library GT4Py (Gridtools for Python) for stencil computations in grid-point NWP and climate models (Afanasyev et al. 2021).

❑ https://github.com/GridTools/gt4py (public, open source)

❑ 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**

GridTools 4python

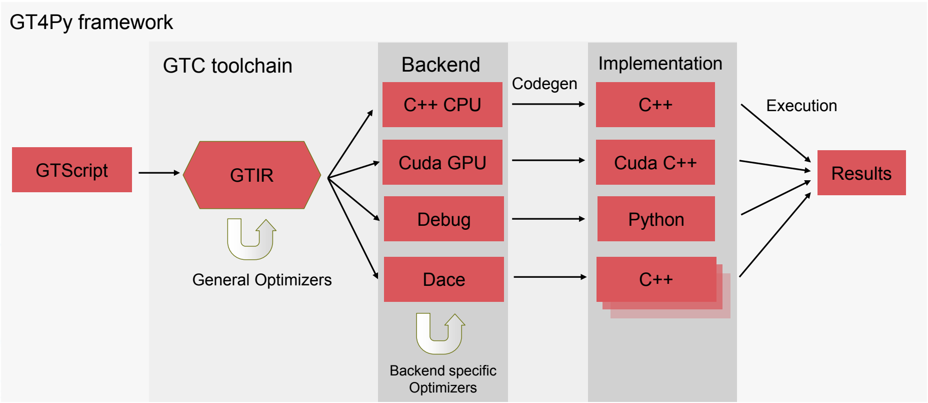
❑ GT4Py is embedded in the **Python** eco-system

- Portable and productive programming environment
- Broad and comprehensive selection of modules and libraries
- Most popular data scientist language
- Low barrier of entry for domain scientists and academia

GT4Py domain-specific library

Established **GT4Py “Version 1”** for structured (I, J, K) grids.
See Afanasyev et al. 2021; Ben-Hun et al. 2022.

→ *Results in subsequent slides are based on structured (I, J, K) grid model.*



GT4Py framework

GTC toolchain

Backend

Codegen

Implementation

Execution

GTScript

GTIR

General Optimizers

C++ CPU

Cuda GPU

Debug

Dace

Backend specific Optimizers

C++

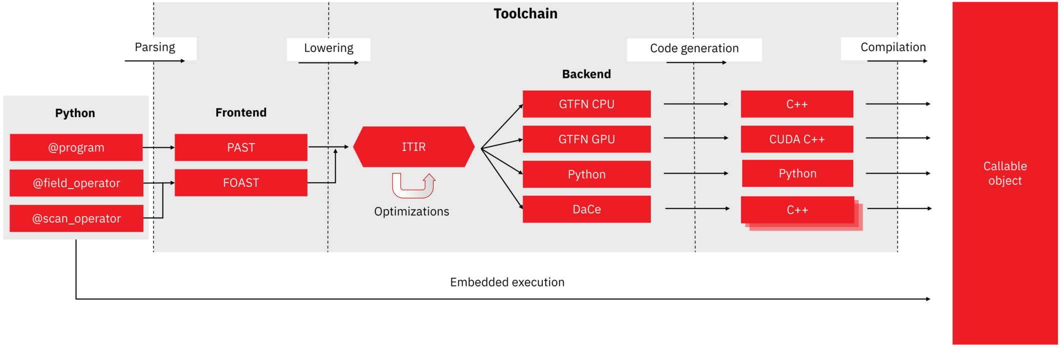
Cuda C++

Python

C++

Results

Declarative GT4Py with new interface & toolchain and supporting horizontally unstructured, e.g., (IJ, K), grids.
This is an ongoing development, see e.g. Bianco et al. PASC2023 poster.



Toolchain

Parsing

Lowering

Code generation

Compilation

Python

@program

@field_operator

@scan_operator

Frontend

PAST

FOAST

ITIR

Optimizations

Backend

GTFN CPU

GTFN GPU

Python

DaCe

C++

CUDA C++

Python

C++

Callable object

Embedded execution

ECMWF

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

PASC
Platform for Advanced Scientific Computing

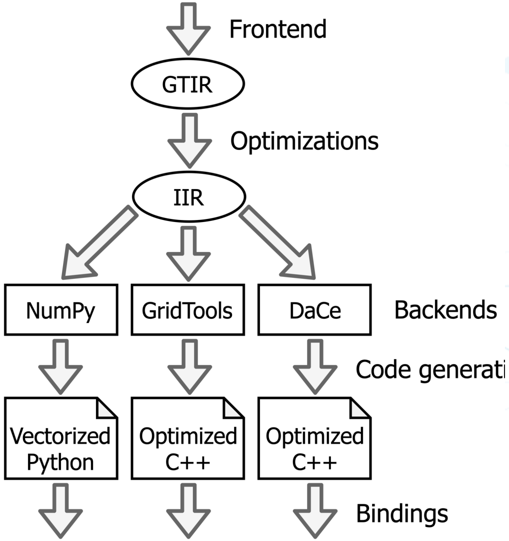
ETH zürich

esiwace

UIFCW 2023
A UFS Collaboration Powered by EPIC

GT4Py domain-specific library

```
@gtscript.stencil(backend="...")
def laplacian(
    in_phi: gtscript.Field[float], out_lap: gtscript.Field[float]
):
    with computation(PARALLEL), interval(...):
        out_lap[0, 0, 0] = - 4 * in_phi[0, 0, 0] \
            + in_phi[-1, 0, 0] + in_phi[1, 0, 0] \
            + in_phi[0, -1, 0] + in_phi[0, 1, 0]
```



```
laplacian(phi, lap, origin=(1, 1, 0), domain=(nx-2, ny-2, nz))
```

Schematic from Ubbiali et al. in prep. 2023

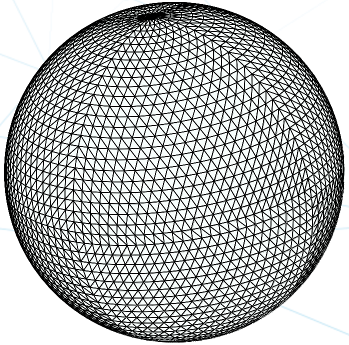
GT4Py domain-specific library

□ Three comprehensive GT4Py based NWP and climate model software development projects (all coming from original Fortran implementations that were optimized for CPUs):

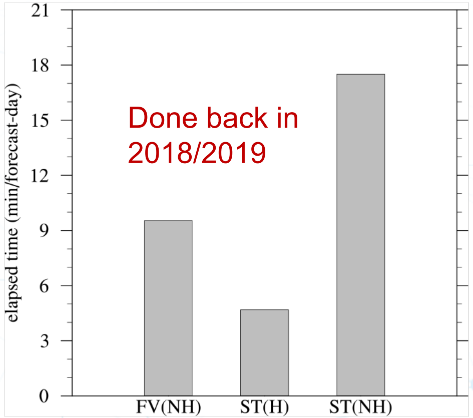
- **Pace** is a GT4Py Version 1 implementation of the **FV3GFS / SHiELD** atmospheric model of NOAA and GFDL by Allen Institute for AI (AI2), ETH Zurich, and CSCS.
- **ICON** atmospheric model dynamics and physics incrementally ported to declarative GT4Py by MeteoSwiss, EXCLAIM project at ETH Zurich and CSCS.
- **IFS-FVM** porting and further development in Version 1 & declarative GT4Py by ECMWF, CSCS, and PASC-funded project KILOS at ETH Zurich.

Starting point

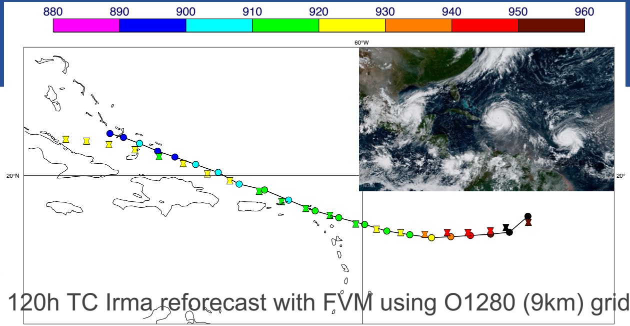
- FVM original **Fortran code with hybrid MPI & OpenMP parallelization targeting CPU based supercomputers**
- Carefully optimized with good computational performance, e.g., against H and NH spectral-transform IFS for dynamical cores
- Operated on IFS octahedral grid provided by ECMWF Atlas library support



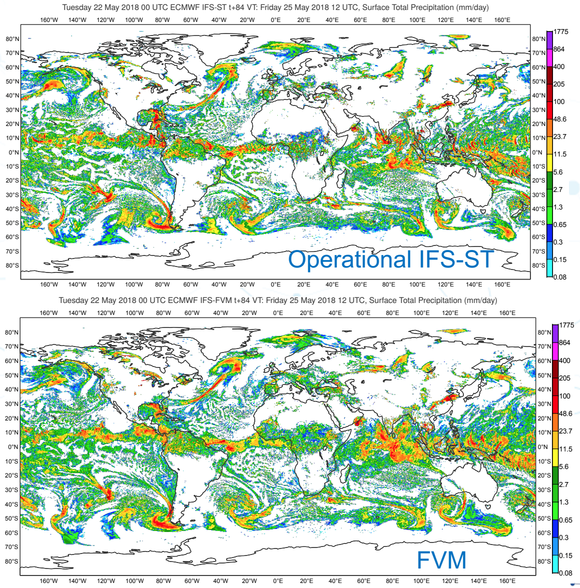
O24



Dynamical core time-to-solution for DCMIP2016 baroclinic wave with TCo1279 / O1280 (9km) grid and L137 on 350 nodes (12800 cores) of ECMWF's former Cray XC40



120h TC Irma reforecast with FVM using O1280 (9km) grid



Tuesday 22 May 2018 00 UTC ECMWF IFS-ST t+84 VT: Friday 25 May 2018 12 UTC, Surface Total Precipitation (mm/day)
Operational IFS-ST
Tuesday 22 May 2018 00 UTC ECMWF IFS-FVM t+84 VT: Friday 25 May 2018 12 UTC, Surface Total Precipitation (mm/day)
FVM
1775
864
400
205
100
48.6
23.7
11.5
5.6
2.7
1.3
0.65
0.3
0.15
0.08

Snapshot total precip at 84h

ECMWF

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

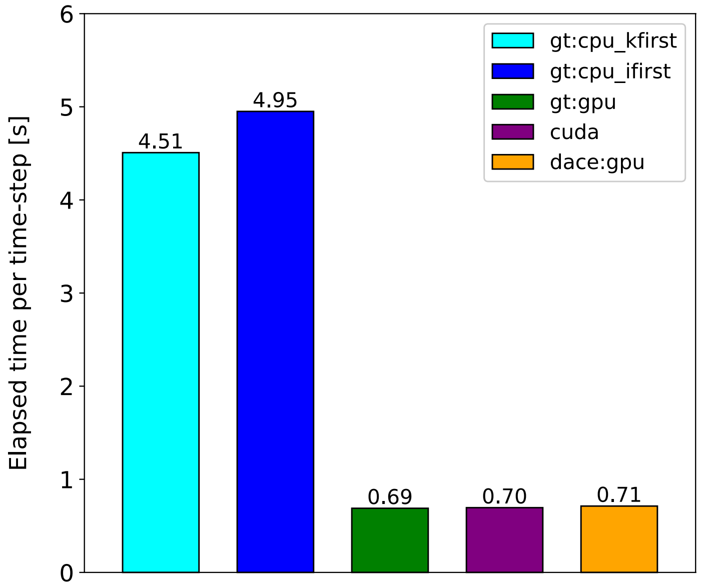
PASC
Platform for Advanced Scientific Computing

ETH zürich

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

UIFCW 2023
A UFS Collaboration Powered by EPIC

Seamless execution of structured-grid FVM with various GT4Py backends

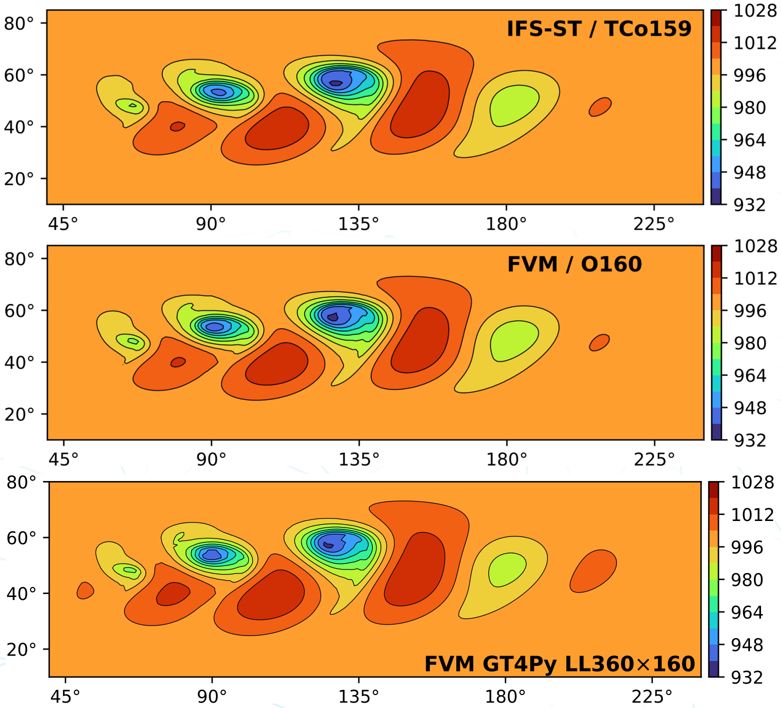


The **single model domain/user interface** in Python with embedded GT4Py for stencil computations **seamlessly drives various backends** by a simple switch, e.g.,:

- ❖ **gt:cpu_ifirst**: C++ CPU where array layout is (I, J, K)
- ❖ **gt:cpu_kfirst**: C++ CPU where array layout is (K, I, J)
- ❖ **gt:gpu**: Gridtools CUDA C++
- ❖ **cuda**: native CUDA C++
- ❖ **dace:gpu**: GPU backend leveraging the Data-Centric Parallel Programming Framework (DaCe, Ben-Hun et al. 2019, 2022). See https://github.com/spcl/dace .

One single user / domain interface ⟷ multiple targeted backends

Validating the GT4Py based dynamical core against references models



Spectral-transform IFS (**ECMWF operational dynamical core**) on octahedral grid

FVM original **Fortran** code on octahedral grid (runs on **CPUs only**)

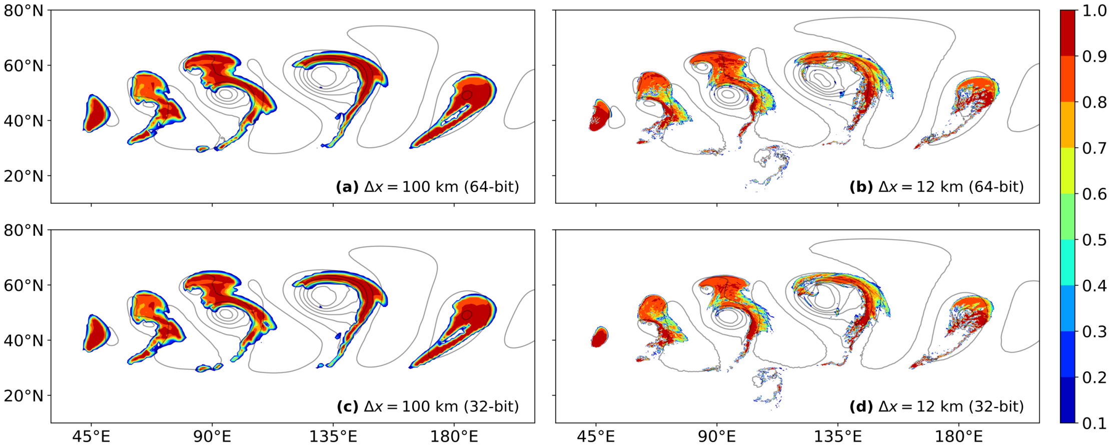
FVM implemented entirely in **Python with GT4Py** Version 1 on regular lat-lon grid (runs with **various backends on CPUs and GPUs**)

Shown is pressure on lowest level (hPa) at day 10 of DCMIP2016 baroclinic wave

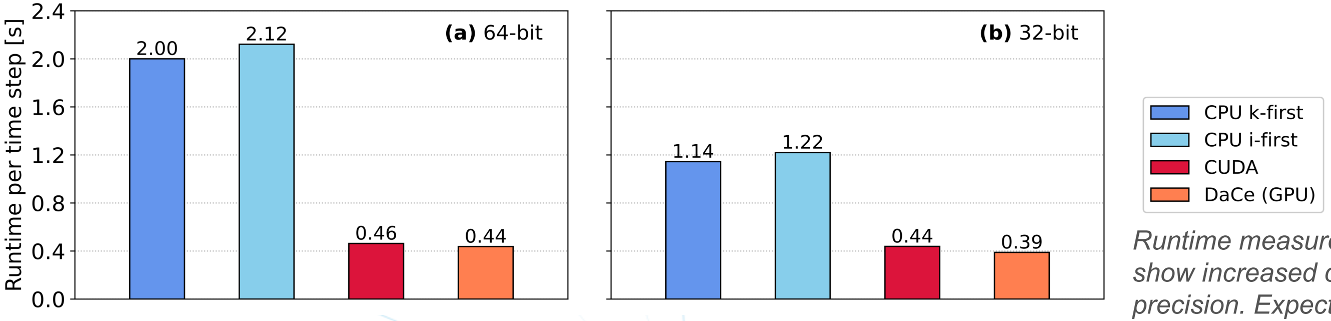
ECMWF CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre pasc Platform for Advanced Scientific Computing ETH zürich esiwace UIFCW 2023 A UFS Collaboration Powered by EPIC

Reduced precision

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 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.



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

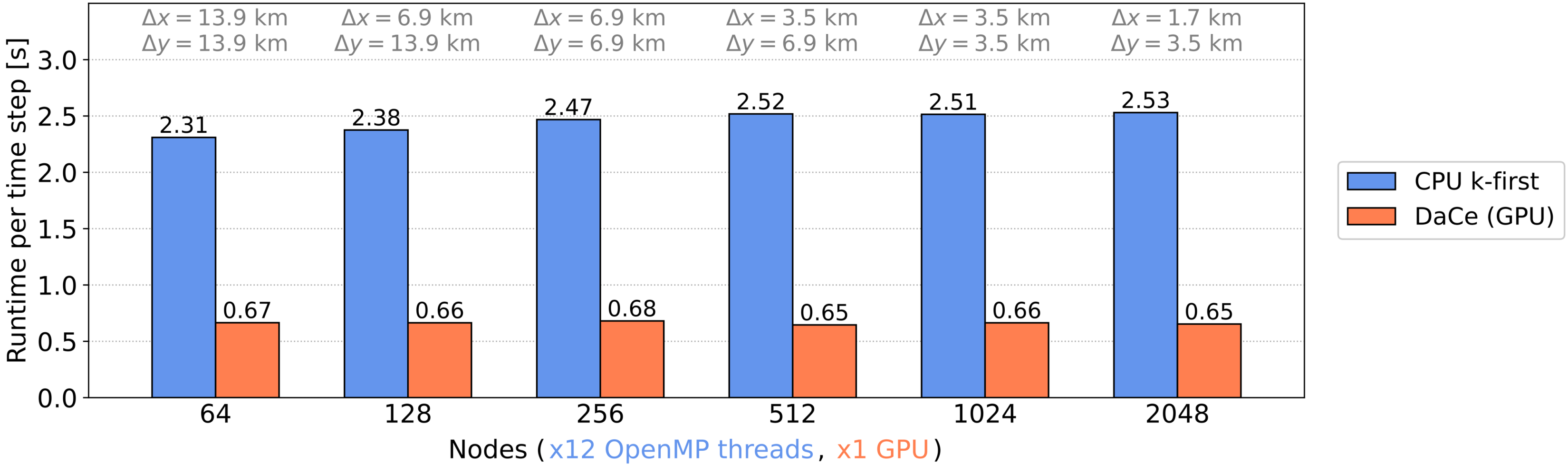
PASC
Platform for Advanced Scientific Computing

ETH zürich

esiwace

UIFCW 2023
A UFS Collaboration Powered by EPIC

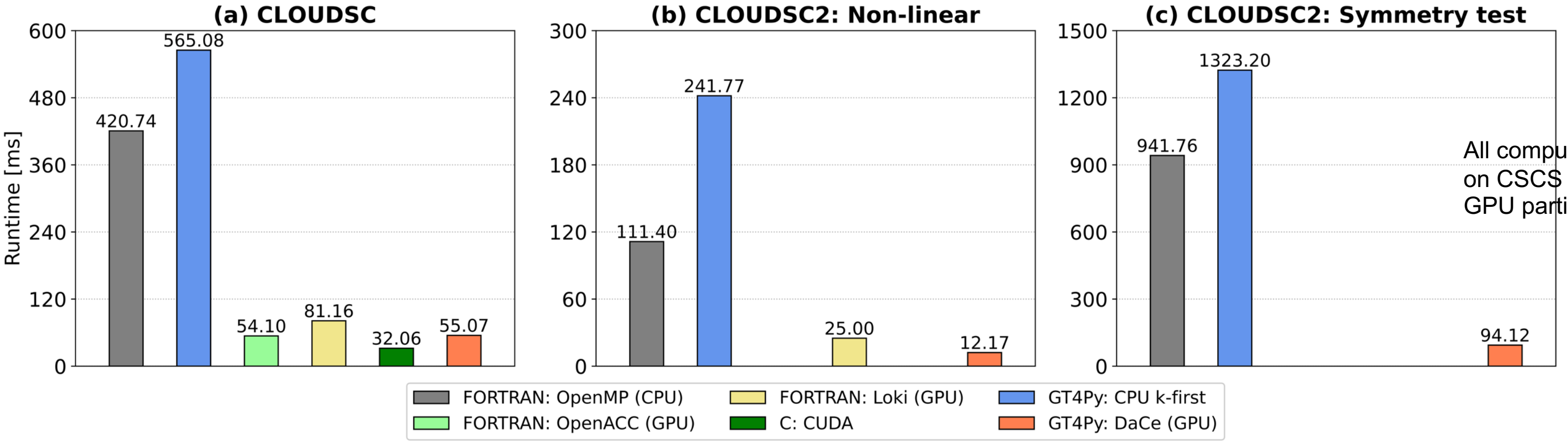
Distributed model



- **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.
- 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 UIFCW 2023 A UFS Collaboration Powered by EPIC

GT4Py for ECMWF parametrizations and TL/AD model code



All computations performed on CSCS Piz Daint (CPU or GPU partitions)

- In the PASC funded KILOS project at ETH Zurich, we have been exploring the porting of IFS physical parametrizations to Python with GT4Py. Performance comparisons for ECMWF’s prognostic cloud scheme CLOUDSC (a).

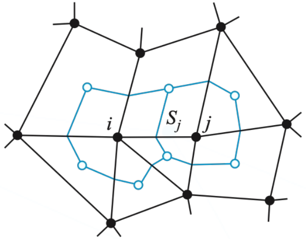
- CLOUDSC2 is the simplified cloud scheme with tangent-linear (TL) and adjoint (AD) model used in 4DVAR assimilation. Timings for running the TL/AD symmetry test on CPUs and for the first time on GPUs is shown in (c). Study is currently extended and prepared for publication (Ubbiali et al. in prep. 2023).

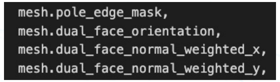
ECMWF CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre pasc Platform for Advanced Scientific Computing ETH zürich esiwace UIFCW 2023 A UFS Collaboration Powered by EPIC

Upcoming: FVM implementation based on the declarative GT4Py

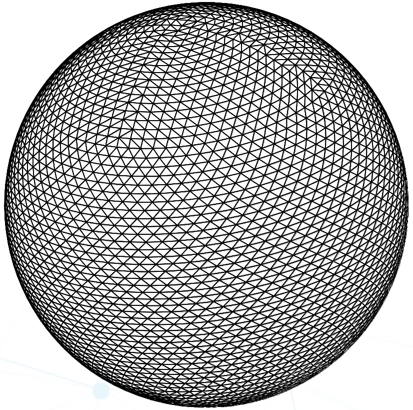
Python embedded GT4Py Field Operator example for simple Upwind scheme:

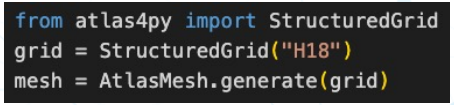
```python
@field_operator
def advection_scheme_upwind(
    rho: Field[[Vertex], float],
    dt: float,
    vel: tuple[Field[[Vertex], float], Field[[Vertex], float]],
    vol: Field[[Vertex], float],
    dual_face_orientation: Field[[Vertex, V2EDim], float],
    dual_face_normal: tuple[Field[[Edge], 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)
```





```
mesh.pole_edge_mask,
mesh.dual_face_orientation,
mesh.dual_face_normal_weighted_x,
mesh.dual_face_normal_weighted_y,
```





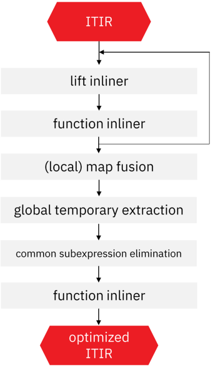
```python
from atlas4py import StructuredGrid
grid = StructuredGrid("H18")
mesh = AtlasMesh.generate(grid)
```

Three main operator concepts:

@program: Sequence of (stateful) operator calls transforming the input args and writing back the return value to a specified output field

@field_operator: covering all common patterns of finite-volume stencils with multiple field operations

@scan_operator: expressing computations with sequential dependencies such as in direct implicit schemes and physical parametrizations



Goal for 2024: Optimized declarative GT4Py-based global non-hydrostatic dynamical core in GPU distributed configuration

ECMWF

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

PASC
Platform for Advanced Scientific Computing

ETH zürich

esiwace

UIFCW 2023
A UFS Collaboration Powered by EPIC

Thank you for listening!

ECMWF

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Platform for Advanced Scientific Computing

ETH zürich

esiwace

UIFCW 2023
A UFS Collaboration Powered by EPIC