Developing next-generation weather models in Python

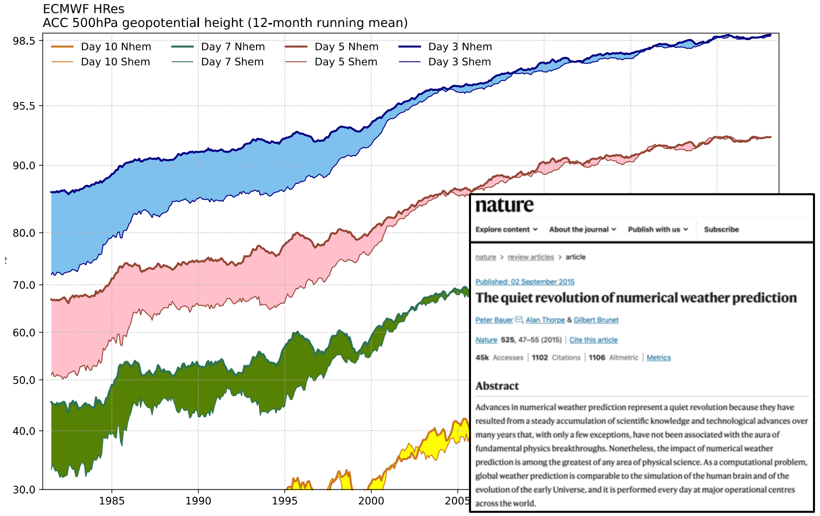
Christian Kühnlein

Platform for Advanced Scientific Computing

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Funded by the European Union
Destination Earth implemented by ECMWF esa EUMETSAT

After decades of steady progress, we are seeing challenges and opportunities



nature

Explore content · About the journal · Publish with us · Subscribe

nature > review articles > article

Published: 02 September 2015

The quiet revolution of numerical weather prediction

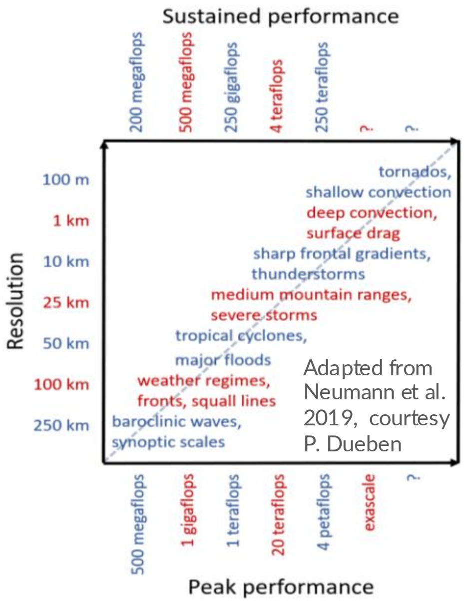
Peter Bauer, Alan Thorpe & Gilbert Brunet

Nature 525, 47–55 (2015) | Cite this article

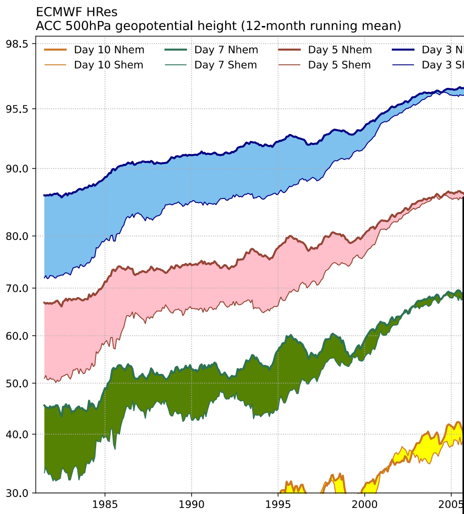
45k Accesses | 1102 Citations | 1106 Altmetric | Metrics

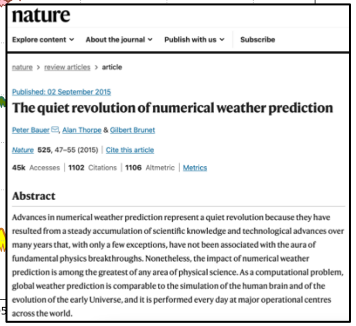
Abstract

Advances in numerical weather prediction represent a quiet revolution because they have resulted from a steady accumulation of scientific knowledge and technological advances over many years that, with only a few exceptions, have not been associated with the aura of fundamental physics breakthroughs. Nonetheless, the impact of numerical weather prediction is among the greatest of any area of physical science. As a computational problem, global weather prediction is comparable to the simulation of the human brain and of the evolution of the early Universe, and it is performed every day at major operational centres across the world.



After decades of steady progress, we are seeing challenges and opportunities




nature

Explore content · About the journal · Publish with us · Subscribe

nature > review articles > article

Published: 02 September 2015

The quiet revolution of numerical weather prediction

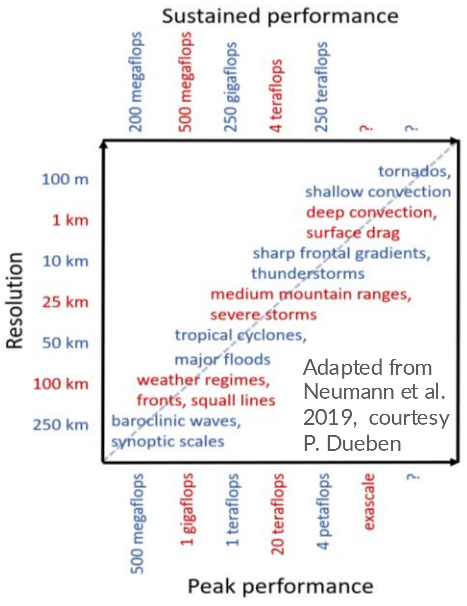
Peter Bauer, Alan Thorpe & Gilbert Brunet

Nature 525, 47–55 (2015) | Cite this article

45k Accesses | 1102 Citations | 1106 Altmetric | Metrics

Abstract

Advances in numerical weather prediction represent a quiet revolution because they have resulted from a steady accumulation of scientific knowledge and technological advances over many years that, with only a few exceptions, have not been associated with the aura of fundamental physics breakthroughs. Nonetheless, the impact of numerical weather prediction is among the greatest of any area of physical science. As a computational problem, global weather prediction is comparable to the simulation of the human brain and of the evolution of the early Universe, and it is performed every day at major operational centres across the world.




PERSPECTIVE

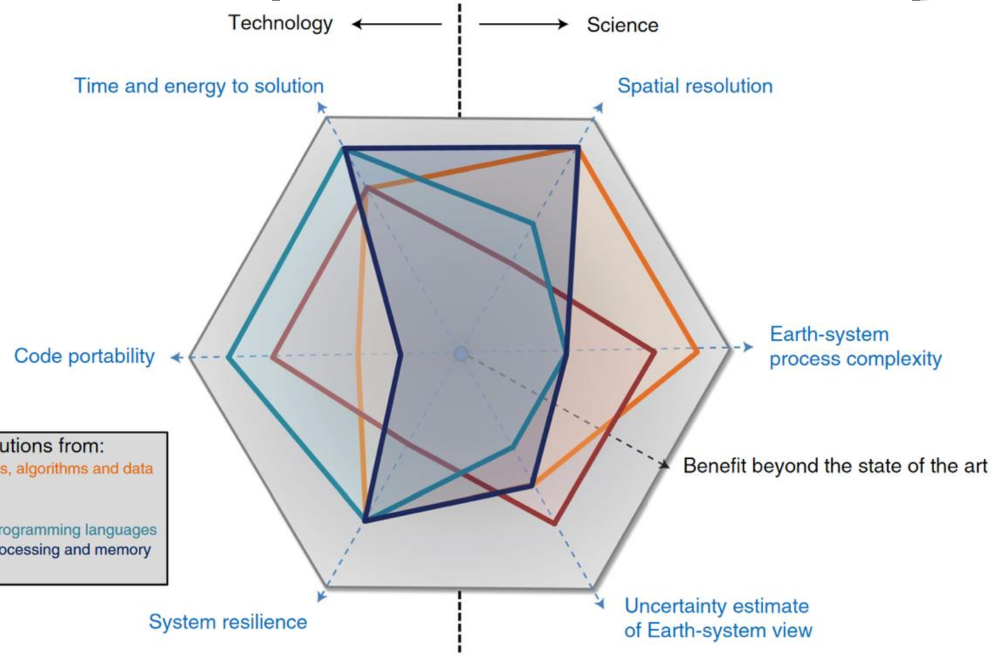
https://doi.org/10.1038/s43588-021-00023-0

nature computational science

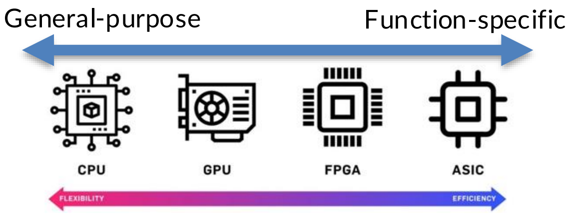
Check for updates

The digital revolution of Earth-system science

Peter Bauer[1], Peter D. Dueben[1], Torsten Hoefler[2], Tiago Quintino[3], Thomas C. Schulthess[4] and Nils P. Wedi[1]

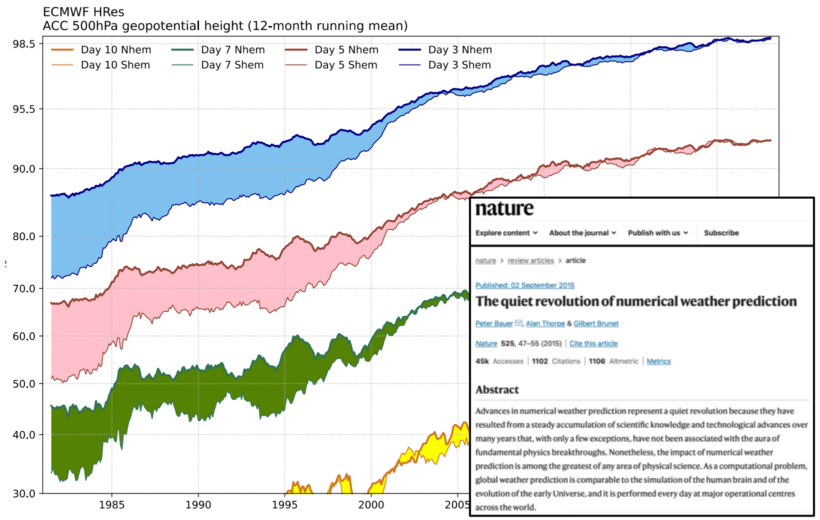


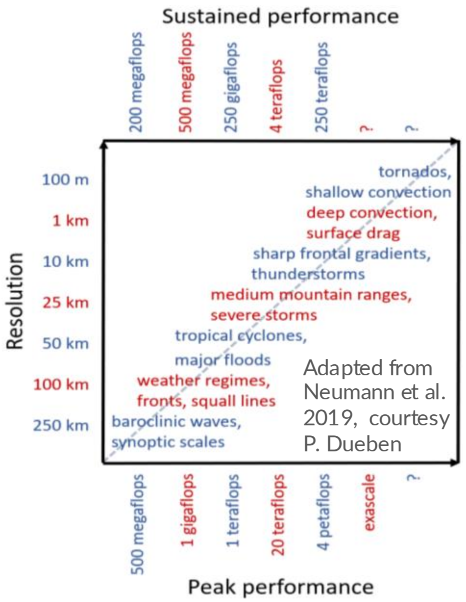
Individual contributions from:
- Numerical methods, algorithms and data structures
- Machine learning
- Domain-specific programming languages
- Heterogeneous processing and memory architectures

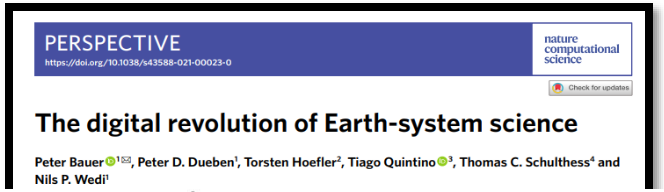


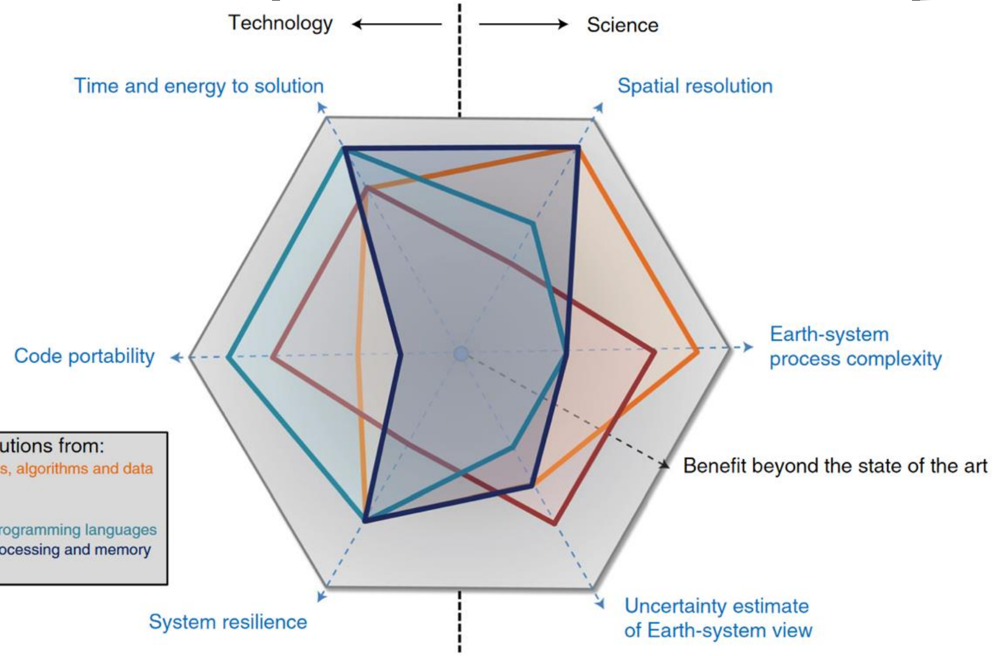
Source: venturebeat.com

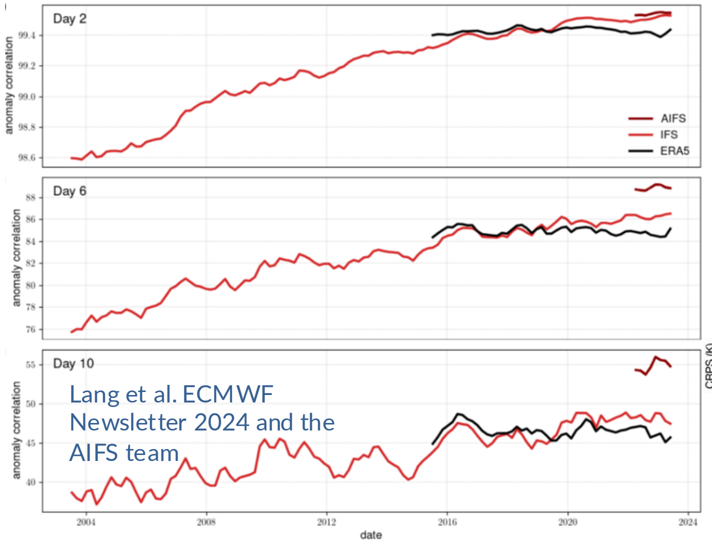
After decades of steady progress, we are seeing challenges and opportunities







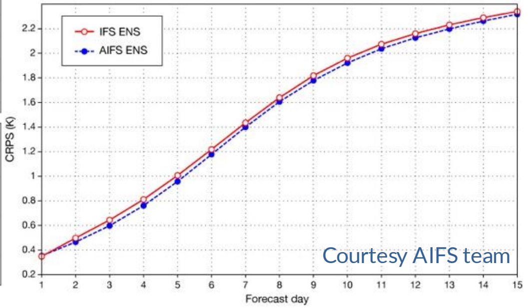




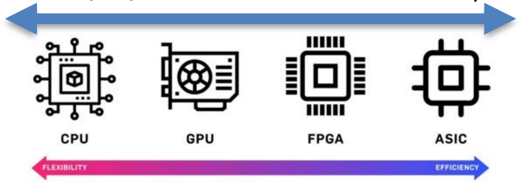
A transformation of weather forecasting is ongoing using various forms of data-driven approaches!

Individual contributions from:

- Numerical methods, algorithms and data structures
- Machine learning
- Domain-specific programming languages
- Heterogeneous processing and memory architectures



TensorFlow
Keras
PyTorch



Source: venturebeat.com

Preparing the existing Integrated Forecasting System at ECMWF for GPUs

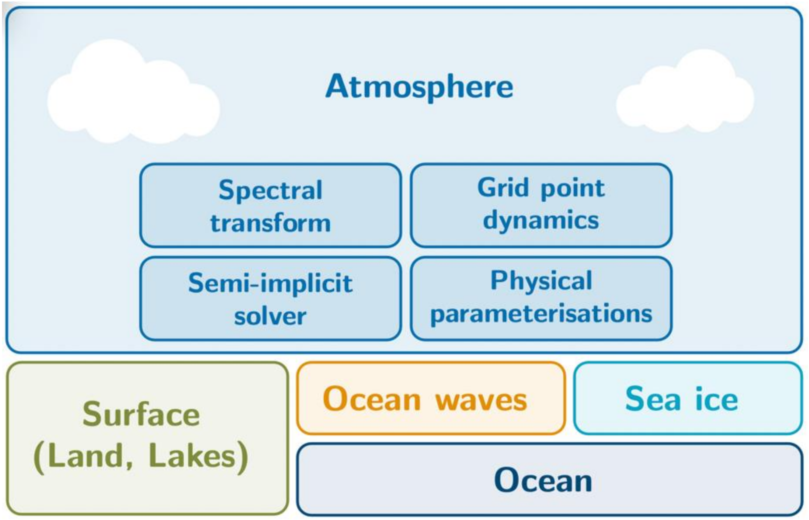
Standalone components

ecTrans
Spectral Transform

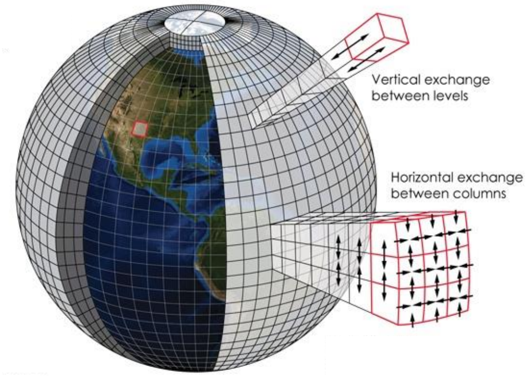
ecRad
Radiation model

ecWAM
Wave model

ecLand
Land surface model



Source-to-source



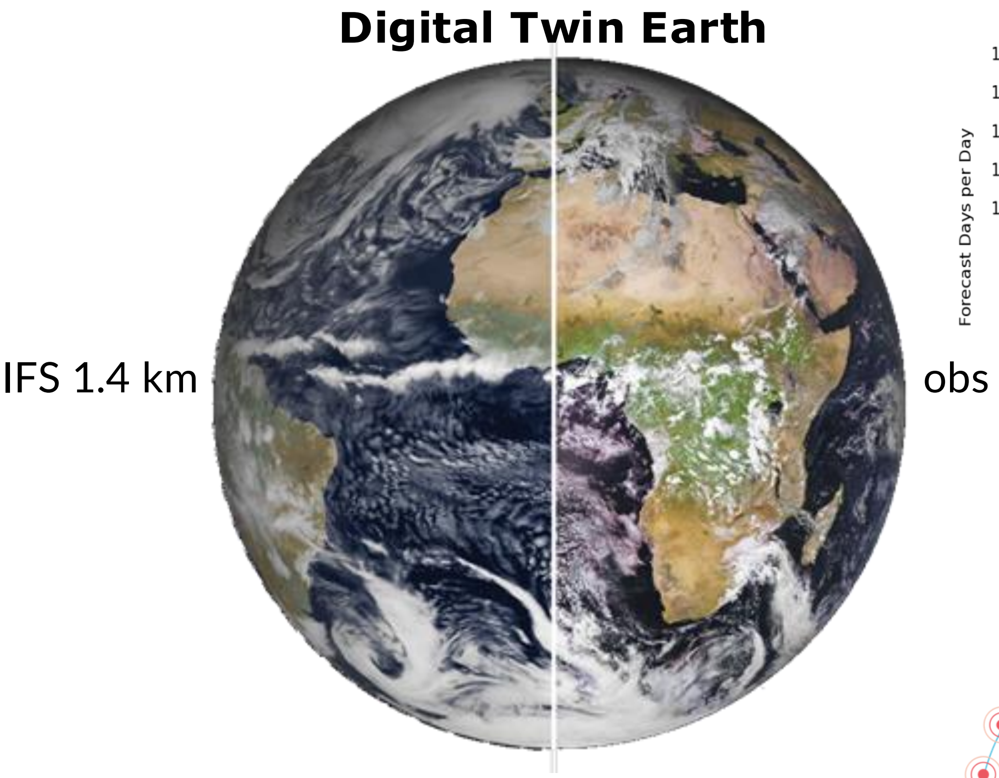
Single-column recipes
Optimise compute kernels
Schedule data transfers

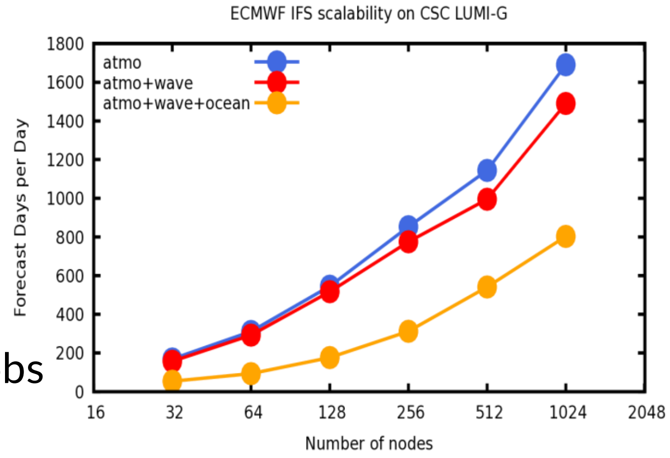
GPU-enabled data structures
Generalised data transfer API

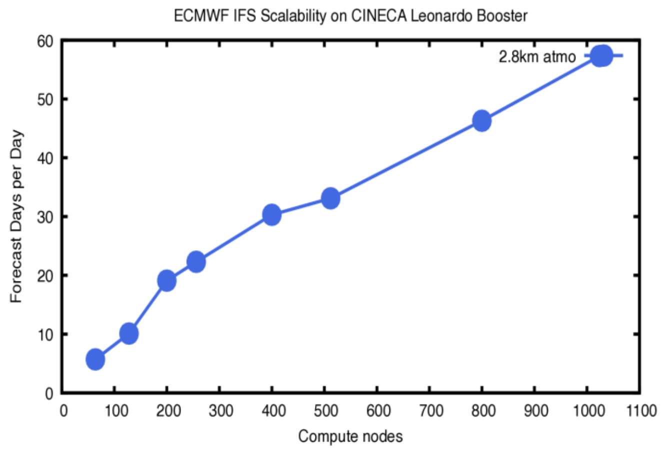
Atlas | FIELD API | Loki S2S tool

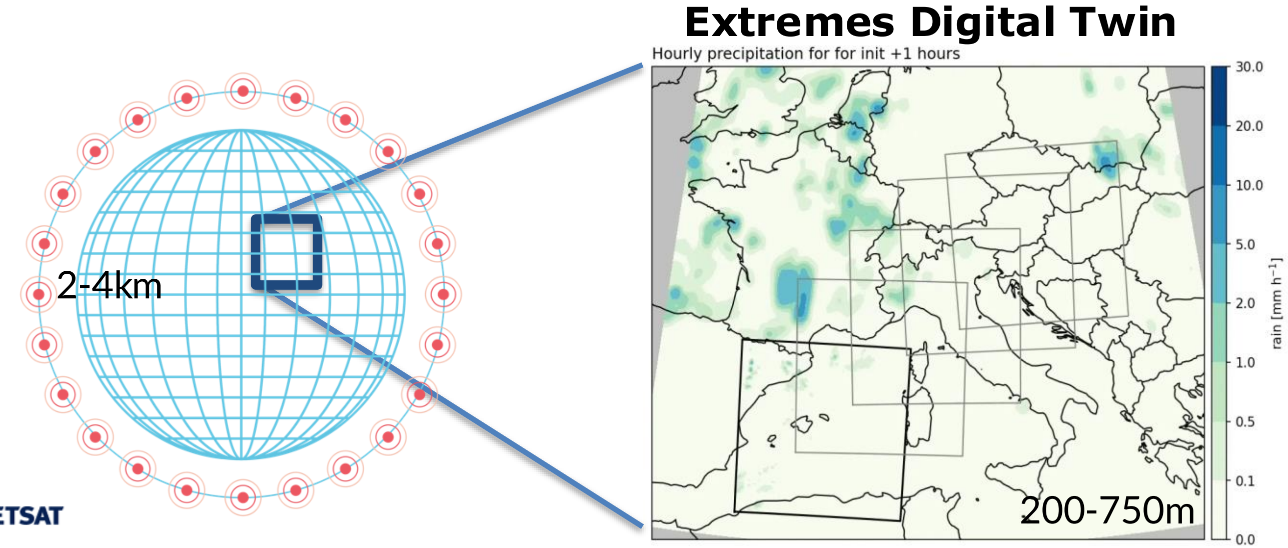
The front-end remains in **Fortran** and hybrid GPU/CPU execution is enabled mostly by means of **directives/pragmas**.

The Integrated Forecasting System in Destination Earth









ECMWF member states using e.g. AROME

Funded by the European Union Destination Earth implemented by ECMWF esa EUMETSAT

Slide courtesy Nils Wedi, ECMWF

Two ECMWF model & software development streams

- **Operational IFS**: ECMWF software efforts are in full swing to prepare the spectral-transform IFS forecast model with the SISL (semi-implicit semi-Lagrangian) integration and the IFS physical parametrization package for **hybrid CPU & GPU execution**. The Loki **automatic code translation** tool is developed and applied to restructured model components and various technical infrastructure packages such as ECMWF Atlas library. The front-end remains in **Fortran** and hybrid execution is enabled mostly by means of **directives/pragmas**.

- Future fully portable high-resolution model for the IFS developed on longer time scale: We **build on FVM** and develop the forecast model entirely in **Python with the domain-specific GT4Py framework** and leverage also other libraries/tools. This project happens in close collaboration with partners at CSCS, ETH Zurich and MeteoSwiss.

Future model development and coding practices

Mathematical Formulation

↓

Discretization

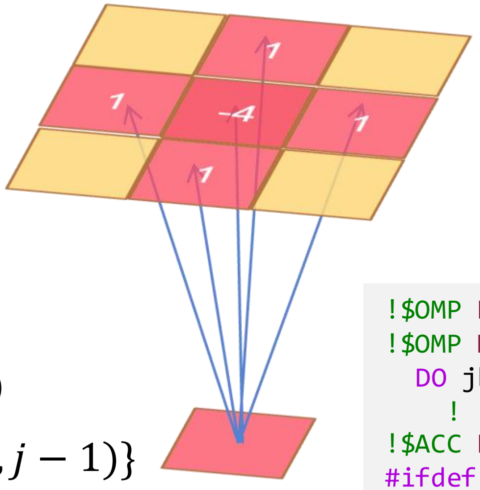
↓

Prototype Implementation

↓

HPC Implementation

$$\Psi = \Delta_{x,y}\phi$$



$$\Psi(i,j) = \frac{1}{h^2}\{-4 * \phi(i,j) + \phi(i+1,j) + \phi(i-1,j) + \phi(i,j+1) + \phi(i,j-1)\}$$

```python
import numpy as np

def laplacian(inp: np.ndarray):
    return (
        -4.0 * inp[1:-1, 1:-1]
        + inp[2:, 1:-1] + inp[:-2, 1:-1]
        + inp[1:-1, 2:] + inp[1:-1, :-2]
    )
```

```fortran
!$OMP PARALLEL
!$OMP DO PRIVATE(jb,i_startidx,i_endidx,je,jk)
  DO jb = i_startblk, i_endblk
    ! ...
!$ACC PARALLEL IF( i_am_accel_node .AND. acc_on )
#ifdef __LOOP_EXCHANGE
    !$ACC LOOP GANG
    DO je = i_startidx, i_endidx
      !$ACC LOOP VECTOR
      DO jk = slev, elev
#else
!CDIR UNROLL=3
    !$ACC LOOP GANG
    DO jk = slev, elev
      !$ACC LOOP VECTOR
      DO je = i_startidx, i_endidx
#endif
        ! here goes the laplacian
      END DO
    END DO
!$ACC END PARALLEL
  END DO
!$OMP END DO NOWAIT
!$OMP END PARALLEL
```

Inspired by slides from O. Fuhrer (MeteoSwiss)

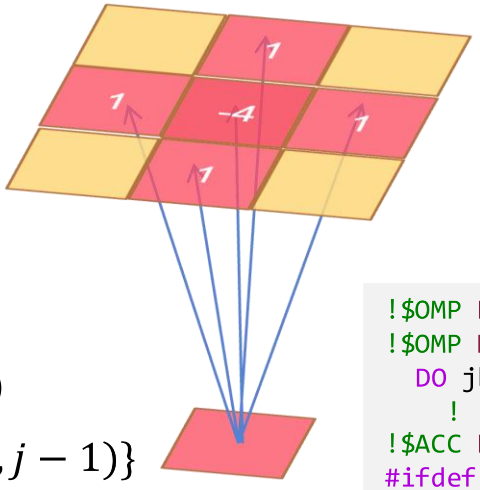
Future model development and coding practices

Mathematical Formulation

$$\Psi = \Delta_{x,y}\phi$$

Discretization

$$\Psi(i,j) = \frac{1}{h^2}\{-4 * \phi(i,j) + \phi(i+1,j) + \phi(i-1,j) + \phi(i,j+1) + \phi(i,j-1)\}$$



Prototype Implementation

```python
@gtx.field_operator
def lap(inp: gtx.Field[gtx.Dims[IDim, JDim, KDim], float]):
    return -4.0 * inp + inp(I-1) + inp(I+1) + inp(J-1) + inp(J+1)
```

HPC Implementation

```fortran
!$OMP PARALLEL
!$OMP DO PRIVATE(jb,i_startidx,i_endidx,je,jk)
  DO jb = i_startblk, i_endblk
    ! ...
!$ACC PARALLEL IF( i_am_accel_node .AND. acc_on )
#ifdef __LOOP_EXCHANGE
   !$ACC LOOP GANG
    DO je = i_startidx, i_endidx
     !$ACC LOOP VECTOR
      DO jk = slev, elev
#else
!CDIR UNROLL=3
   !$ACC LOOP GANG
    DO jk = slev, elev
      !$ACC LOOP VECTOR
      DO je = i_startidx, i_endidx
#endif
        ! here goes the laplacian
      END DO
    END DO
!$ACC END PARALLEL
  END DO
!$OMP END DO NOWAIT
!$OMP END PARALLEL
```

Inspired by slides from O. Fuhrer (MeteoSwiss)

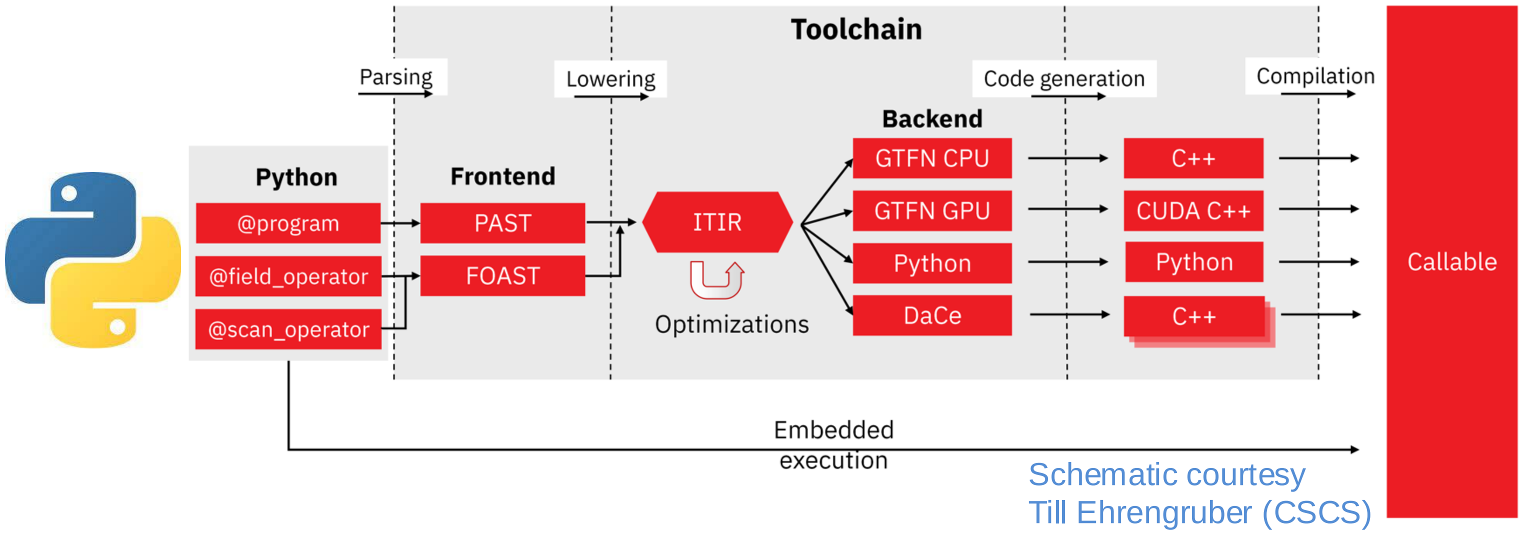
GT4Py high-performance framework in Python

- ❑ The goal is to provide a productive programming environment to write and maintain performant weather and climate model code

- ❑ https://github.com/GridTools/gt4py, https://pypi.org/project/gt4py/ (public, open source, BSD-3 license)

- ❑ GT4Py (GridTools for Python) works as an optimizing compiler for various backends
    - Code generation optimized for a specific architecture
    - Backend selects HPC implementation strategy (e.g., parallelization, memory layout, data flows)
    - Backends for new technologies can be added without any change to the application
    - DaCe (Data-Centric Parallel Programming, Ben-Nun et al. 2019, 2022) framework takes key role in optimization

GT4Py high-performance framework in Python



Schematic courtesy
Till Ehrengruber (CSCS)

- Two main versions of GT4Py
    - gt4py.cartesian: established version supporting 3D structured (I, J, K) grids
    - gt4py.next: new version additionally supporting horizontally unstructured (IJ, K) grids (ongoing development)

Domain-specific GT4Py framework in Python

- **GT4Py** is embedded in the **Python** eco-system
  - Versatile, portable, productive programming language
  - Broad selection of modules/libraries
  - Enables new user and developer workflows
  - Low barrier of entry for domain scientists and academia
  - Favourable choice with respect to ML/AI applications

GridTools 4python

GridTools/**atlas4py**

5 Contributors · 1 Issue · 1 Star · 3 Forks

spcl/**dace**

DaCe - Data Centric Parallel Programming

69 Contributors · 25 Used by · 15 Discussions · 497 Stars · 129 Forks

TensorFlow

Keras

PyTorch

CuPy

NumPy

earthkit

ecmwf/**pyflow**

A high level Python interface to ecFlow allowing the creation of ecFlow suites in a modular and "pythonic" way

9 Contributors · 3 Used by · 8 Stars · 7 Forks

scikit learn

xarray

pandas

pyFFTW/**pyFFTW**

A pythonic python wrapper around FFTW

28 Contributors · 938 Used by · 377 Stars · 108 Forks

GT4Py domain-specific library

➢ **Comprehensive atmospheric model applications are rewritten/developed in Python with GT4Py**

- **Pace** (Ben-Nun et al. 2022; Dahm et al. 2023) is the GT4Py.cartesian implementation of the FV3GFS/SHiELD model of GFDL and NOAA

  https://github.com/NOAA-GFDL/pace

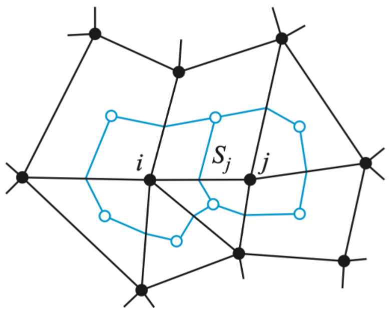
- **ICON4Py is** the GT4Py.next implementation of the ICON model at MeteoSwiss and in the EXCLAIM project at ETH Zurich

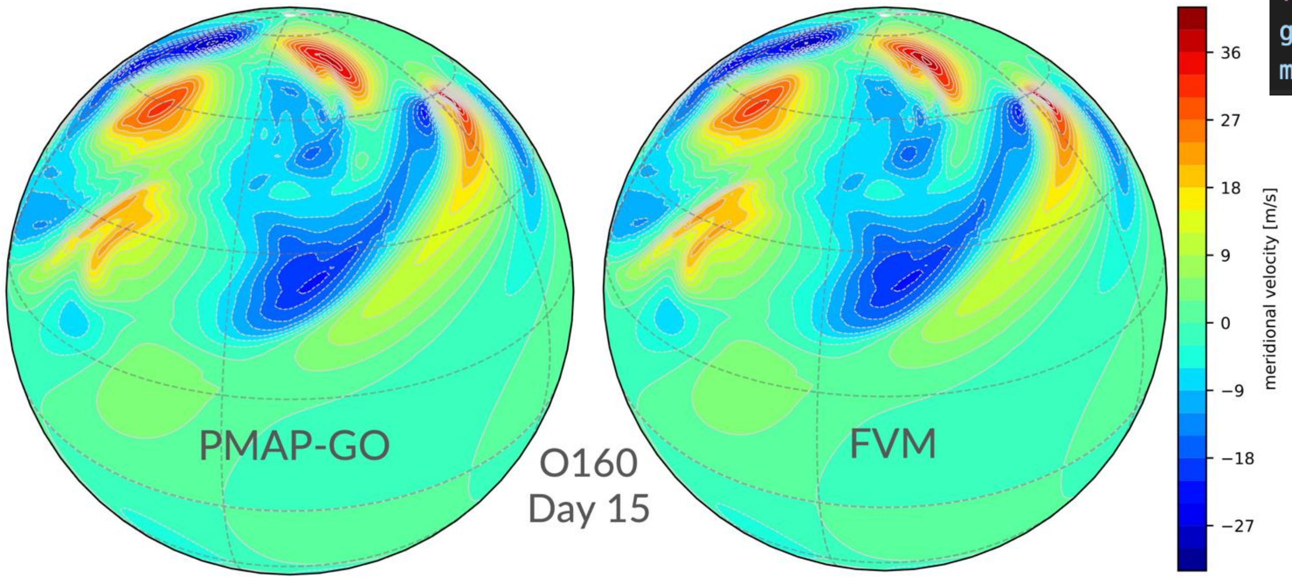
  https://github.com/C2SM/icon4py

GT4Py domain-specific library

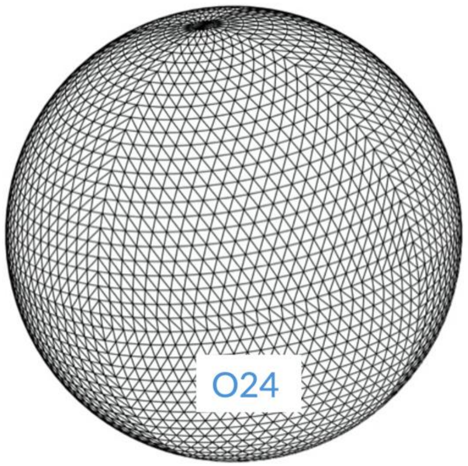
- **PMAP** (Portable Model for Multi-Scale Atmospheric Prediction) is an advancement of the **FVM** model from ECMWF. Currently two configurations exist:
  - **PMAP-LES** is structured grid **LES** model currently using GT4Py.cartesian
  - **PMAP-GO** is **Gl**obal horizontally unstructured with the **O**ctahedral grid using GT4Py.next

PMAP-GO: Global FVM on Octahedral grid fully in Python with GT4Py.next

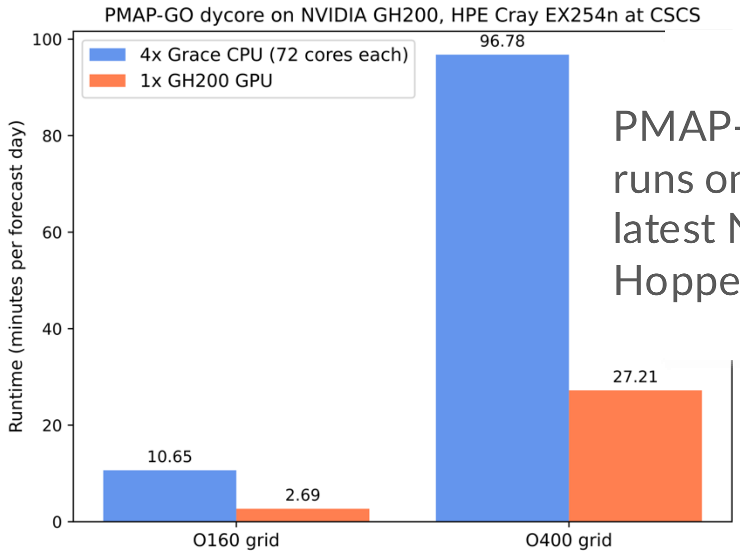





```
from atlas4py import StructuredGrid
grid = StructuredGrid("O24")
mesh = AtlasMesh.generate(grid)
```

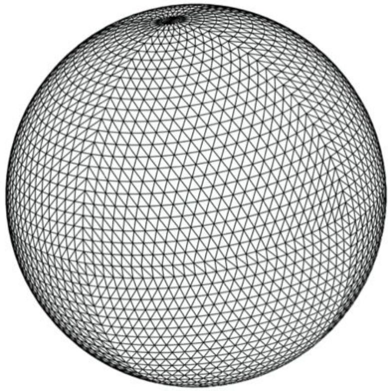



```
@field_operator
def advection_scheme_upwind(
    rho: Field[[Vertex], float],
    dt: float,
    vel: tuple[Field[[Vertex], float], Field[[Vertex], float]],
    vol: Field[[Vertex], float],
    dual_face_orientation: Field[[Vertex, V2EDim], float],
    dual_face_normal: tuple[Field[[Edge], float],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

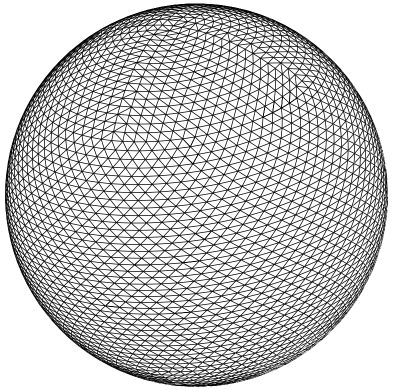


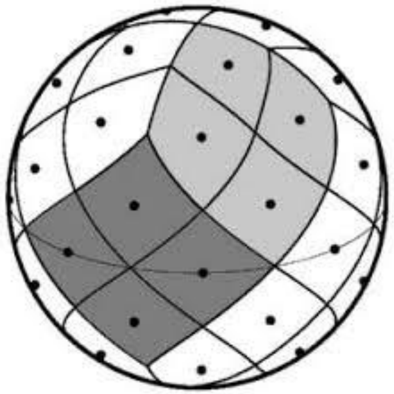
PMAP-GO completely runs on GPU including the latest Nvidia GH200 Grace Hopper Superchip at CSCS

Enabling new global grids and numerical schemes in PMAP



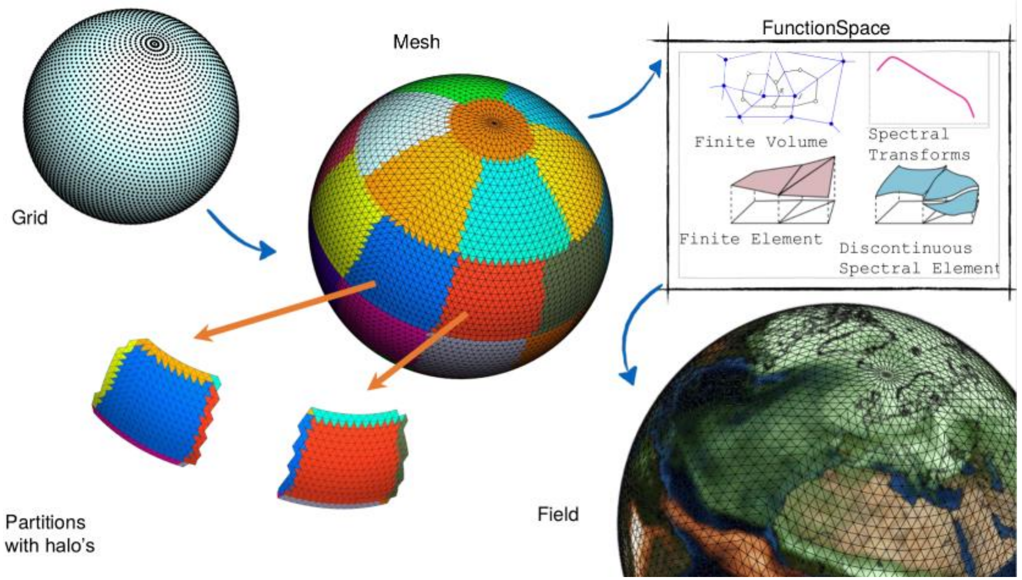
Octahedral grid (operational IFS grid)



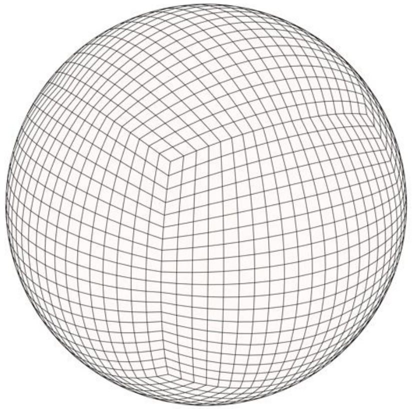


HEALPix grid

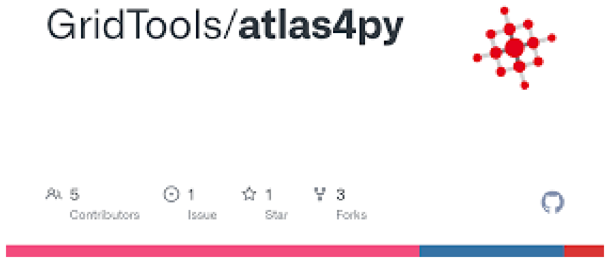
From Gorski et al. 2005



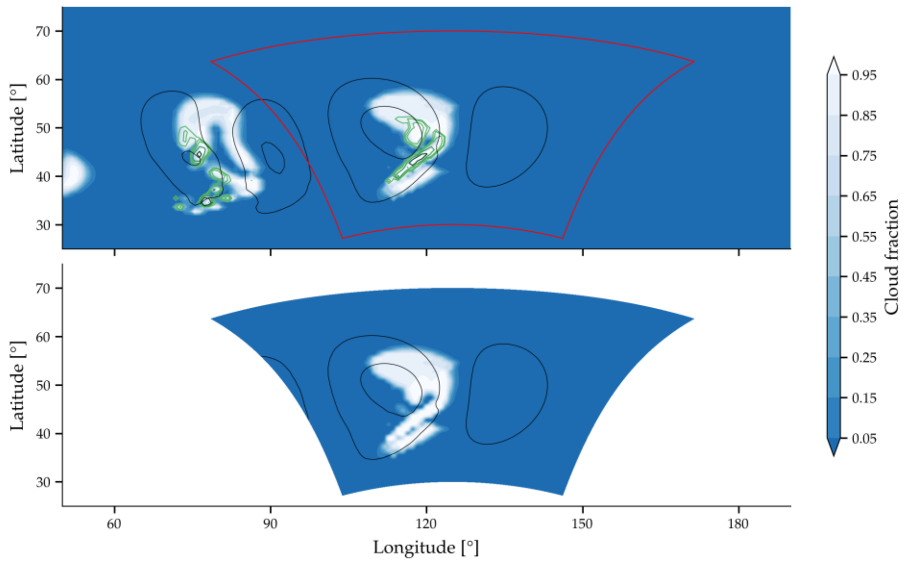
ECMWF Atlas framework (Deconinck et al. 2017)

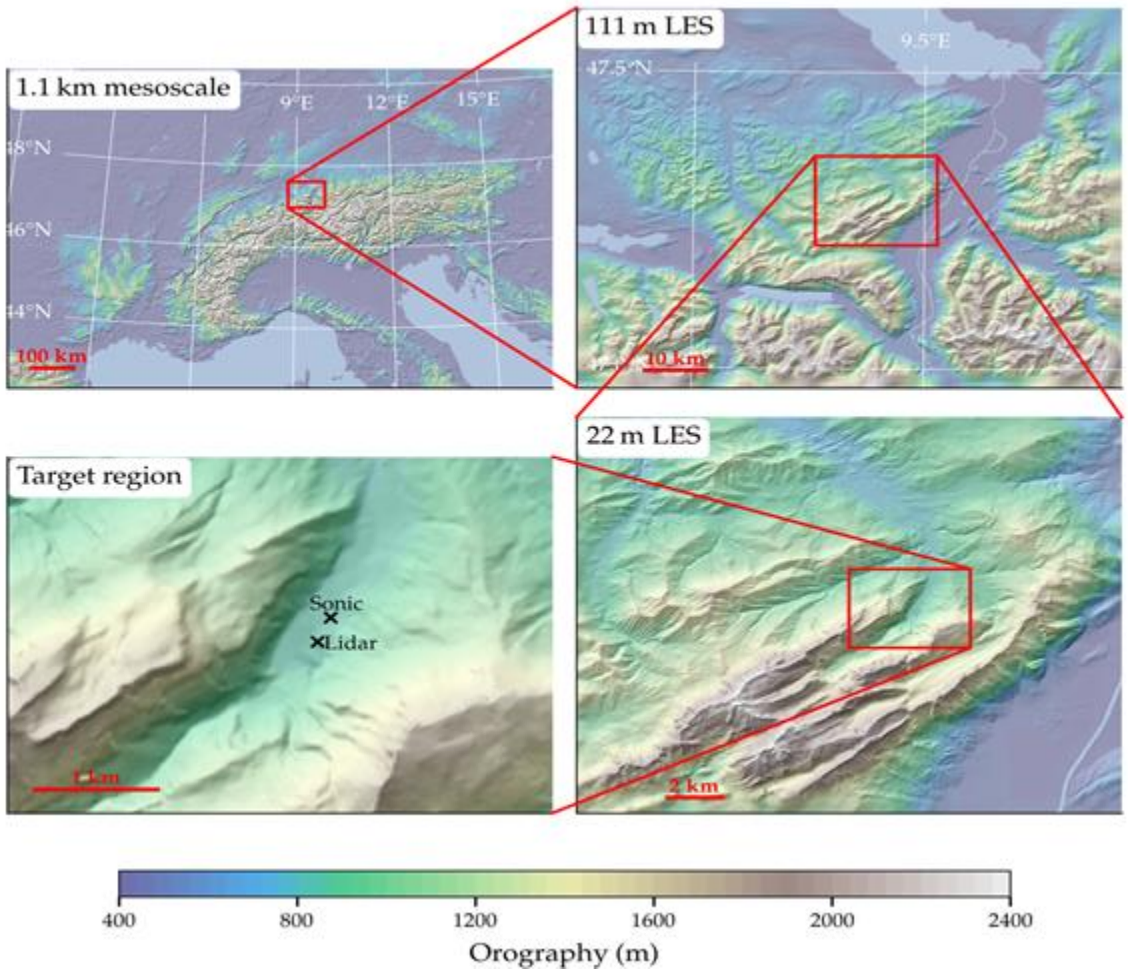


Cubed-sphere grid (ongoing evaluation)



Developing PMAP-LES LAM functionalities for member states and research



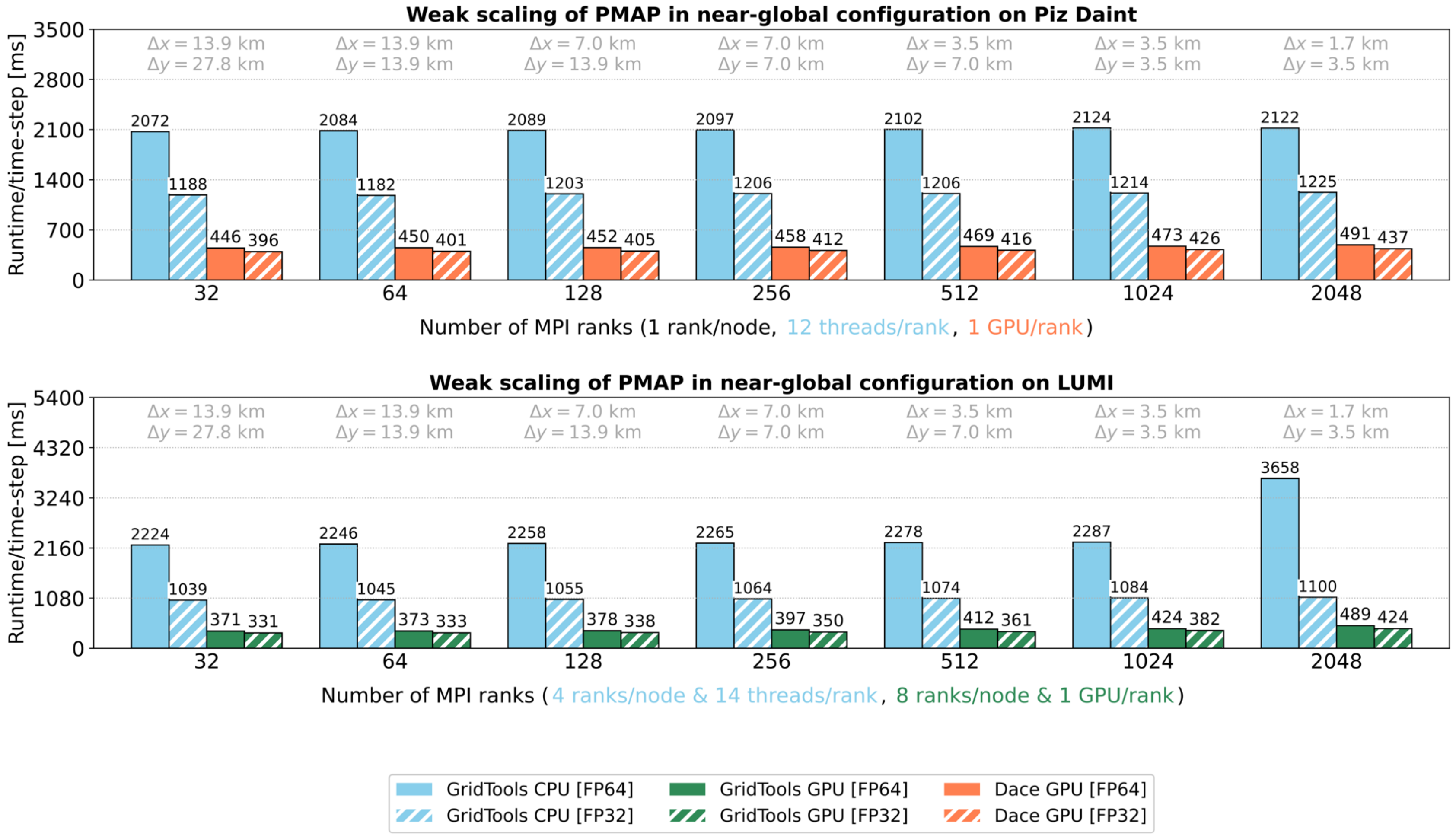


- LAM (Limited-Area Model) funtionalities relevant for some of ECMWF's member states and ETH Zurich research
- Structured quadrilateral grid, rotated spherical coordinates
- Large-eddy simulation (LES) schemes and capabilities (Krieger et al. in prep.; Kühnlein et al. in prep)

Portability and scalability of PMAP on diverse architectures and supercomputers

CSCS Piz Daint

CSC LUMI-G

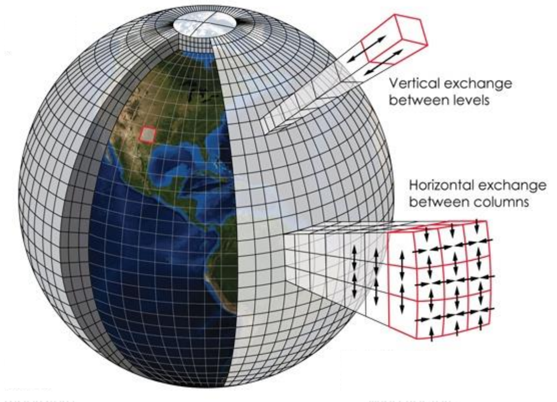


- Tested Nvidia vs AMD GPUs, vs CPUs, GridTools vs DaCe GPU backends, 32 vs 64 bit
- Optimization of PMAP, GT4Py and the distributed model using GHEX (Generic exascale-ready library for halo-exchange operations) is an ongoing process!

Portable IFS physical parametrizations in Python with GT4Py

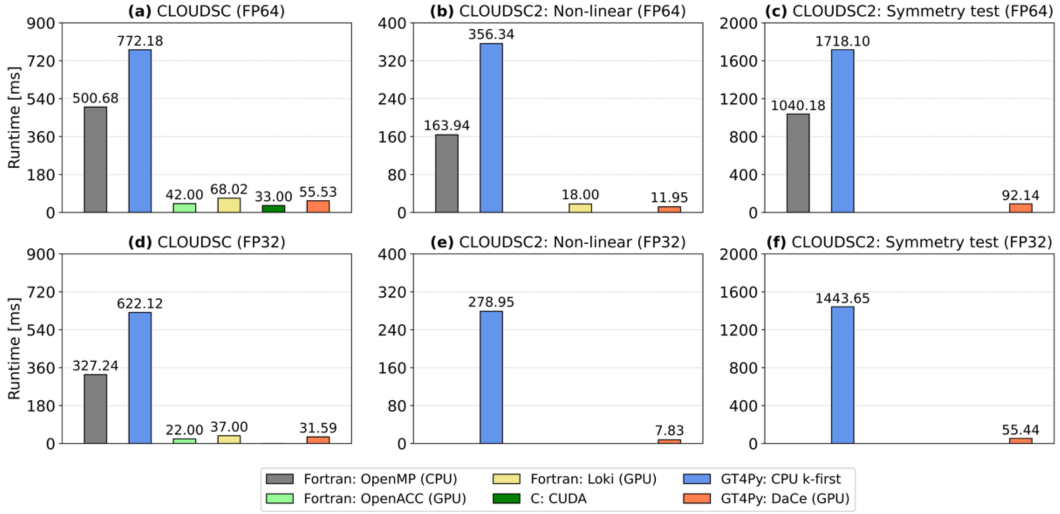
- Developed procedures for manually **porting IFS physical parametrizations to Python with GT4Py** and conducted comprehensive performance study using the CLOUDSC in Ubbiali et al. (https://gmd.copernicus.org/preprints/gmd-2024-92/)
- CLOUDSC CY49R1 ready and coupled, from here each IFS cycle will be updated and validated using the established procedures
- **ecRad porting to Python with GT4Py** has started (G. Vollenweider and S. Ubbiali at ETH Zurich)
- **Land-surface and other parametrizations** will be addressed in 2025-2026, porting will be accelerated by **automatic code translation tool Loki**

- PMAP **physics-dynamics interface in Python** will enable flexibility and various coupling strategies



Exploring GT4Py for the NWP domain using ECMWF microphysics schemes

Piz Daint, Nvidia P100, CSCS, Switzerland



FP64

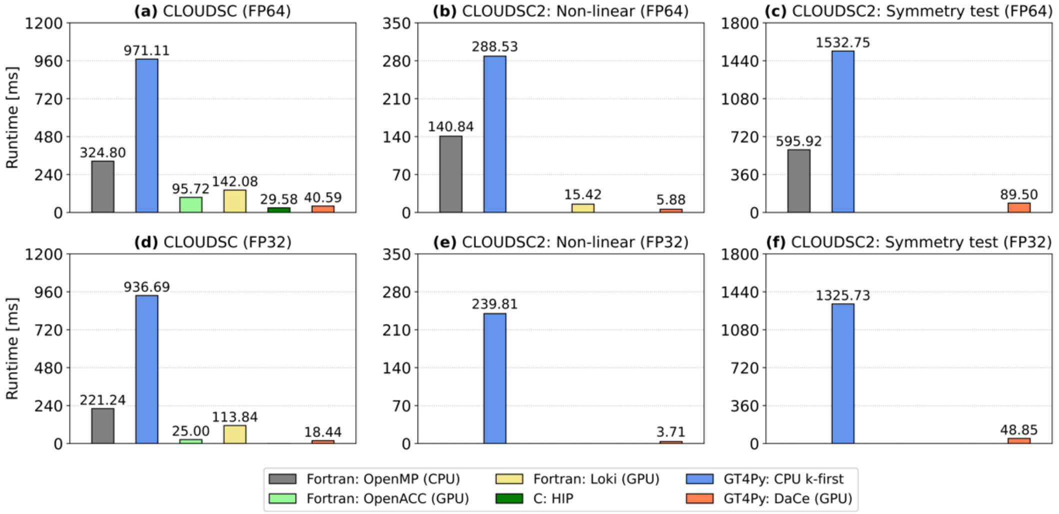
FP32

CLOUDSC

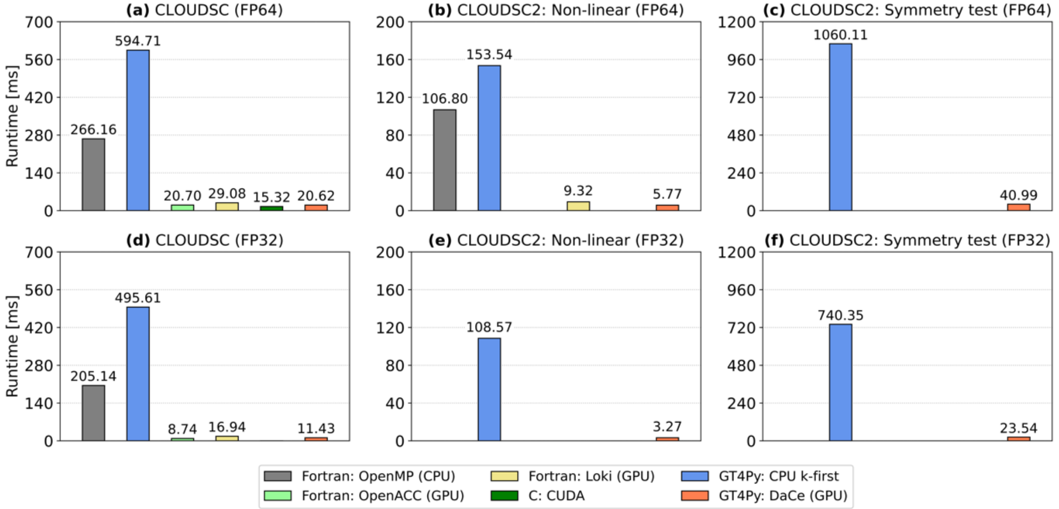
simplified nonlinear

TL/AD symmetry test

LUMI-G, AMD MI 250X CSC, Finland

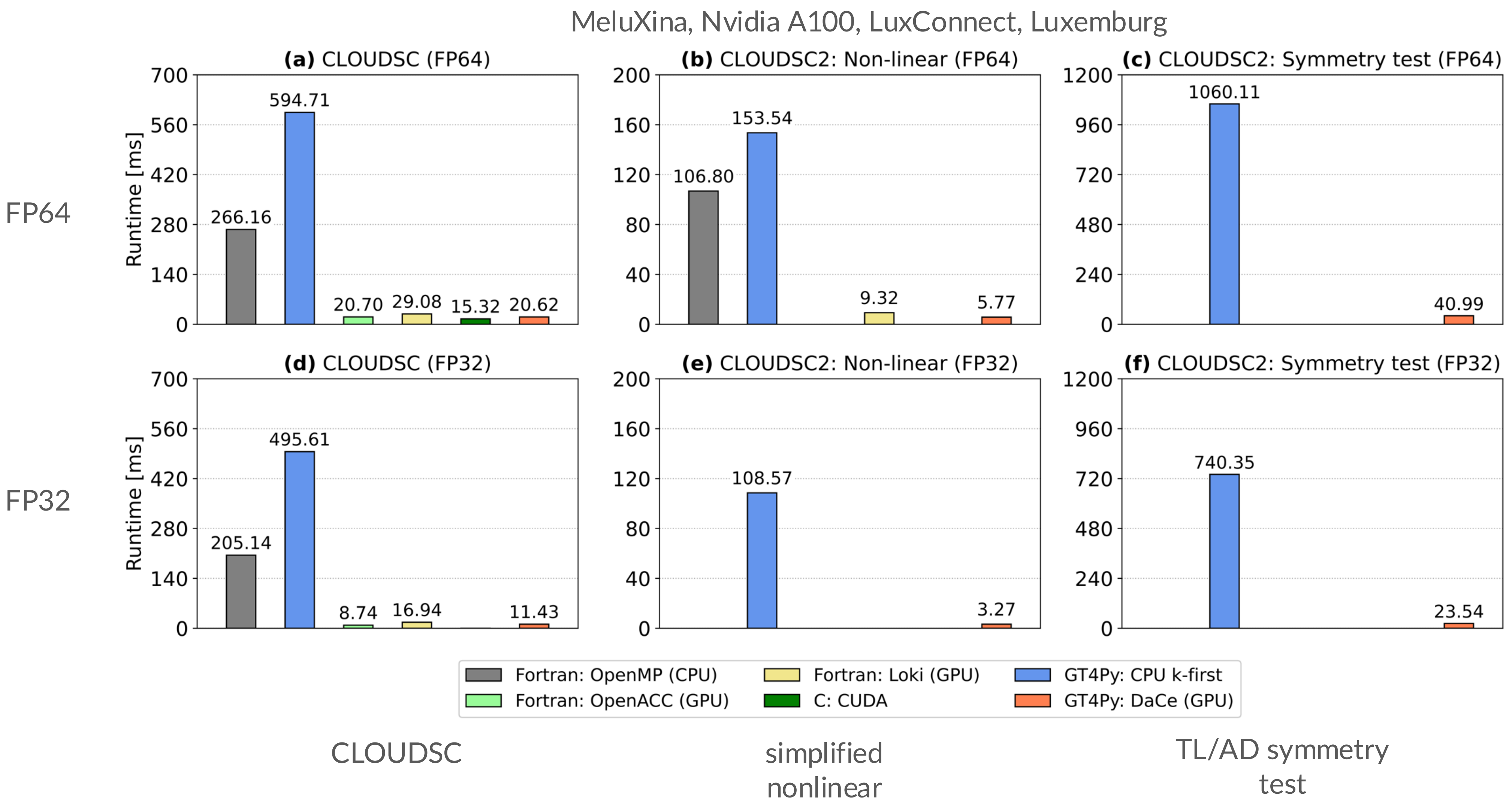


MeluXina, Nvidia A100, LuxConnect, Luxemburg

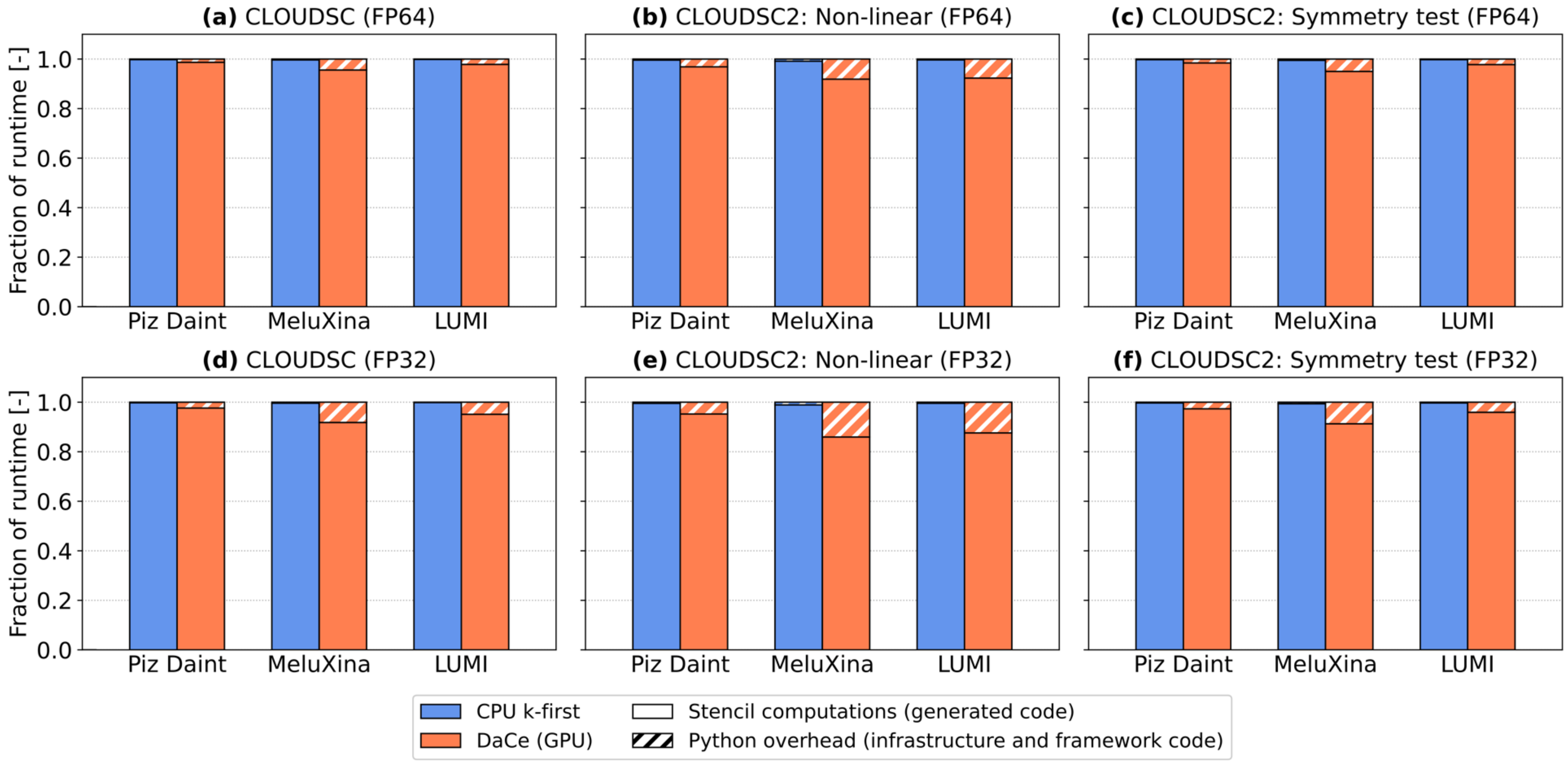


- Performance testing Python implementations of CLOUDSC, simplified nonlinear CLOUDSC2, tangent-linear CLOUDSC2, and adjoint CLOUDSC2 with GT4Py
- GPU vs CPUs, 64-bit vs 32-bit precision, various GT4Py backends
- Ubbiali et al. (https://gmd.copernicus.org/preprints/gmd-2024-92/)

Exploring GT4Py for the NWP domain using ECMWF microphysics schemes



Exploring GT4Py for the NWP domain using ECMWF microphysics schemes



Exploring the Python overhead vs time spend in stencils (generated low-level code)

ECMWF | EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS

The strength of a common goal