ETH zürich — EXCLAIM

Global storm resolving simulations in a python framework - progress and challenges

Anurag Dipankar on behalf of the team

28 June 2023, PASC 23, Davos

C2SM Center for Climate Systems Modeling

CSCS

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

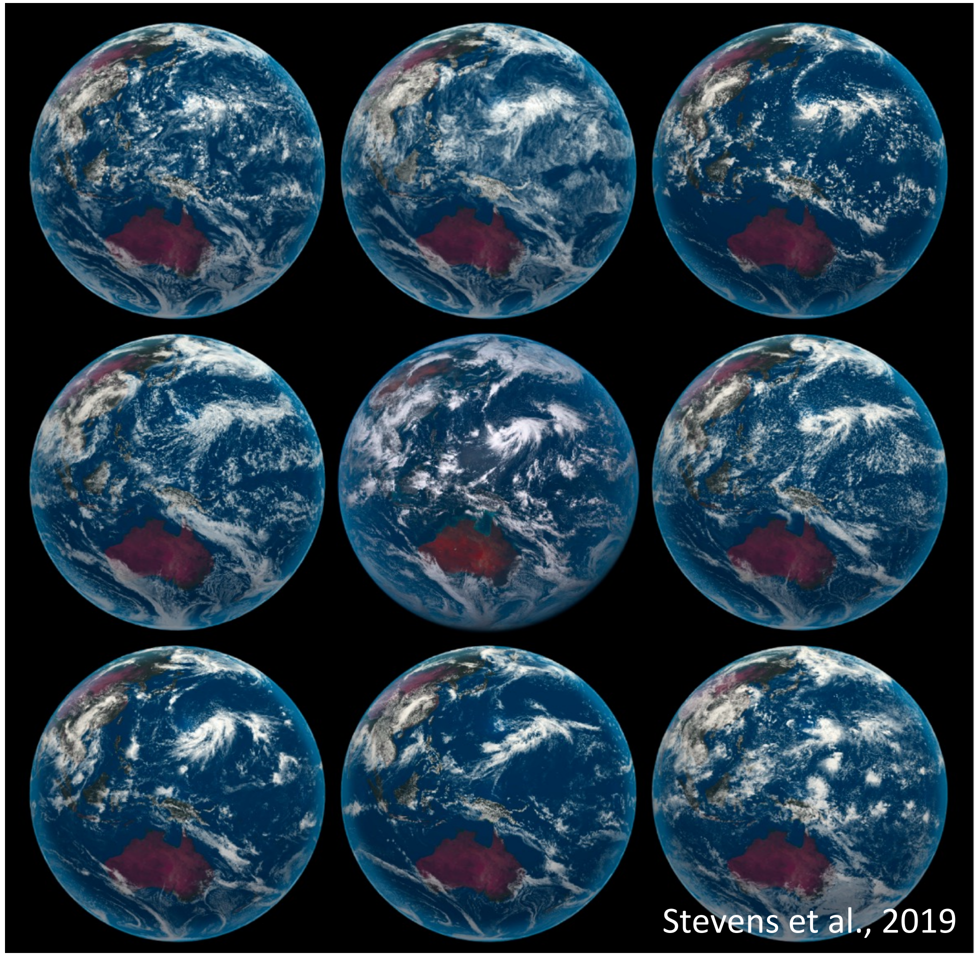
Federal Department of Home Affairs FDHA
Federal Office of Meteorology and Climatology MeteoSwiss

MeteoSwiss

SDSC

Empa
Materials Science and Technology

Global cloud resolving simulations in a python framework



XCLAIM

The EXCLAIM team

- Nicolas Gruber, Thomas Schulthess, Christoph Schär, Oliver Fuhrer

- Anurag Dipankar, Mauro Bianco, Xavier Lapillonne, Christina Schnadt Poderaj, Jacopo Canton, Nicoletta Farabullini, Brigitta Goger, Abishek Gopal, Péter Kardos, Samuel Kellerhals, Magdalena Luz, Chia Rui Ong, Praveen Kumar Pothapakula, Christopher Bignamini, Andreas Jocksh, David Leutwlyer, Will Sawyer, Christoph Müller, Matthias Röthlin, Till Ehrengruber, Enrique Gonzàlez Paredes, Ben Weber, Rico Häuselmann, Felix Thakerm Hannes Vogt, Carlos Osuna, Tamara Bandikova

Overview

- Design choices – why?
- Approach – how?
- Status
  - The black line
  - The blue line
  - The green line
- Summary

Design choices

Gain vs Cost

Typical user

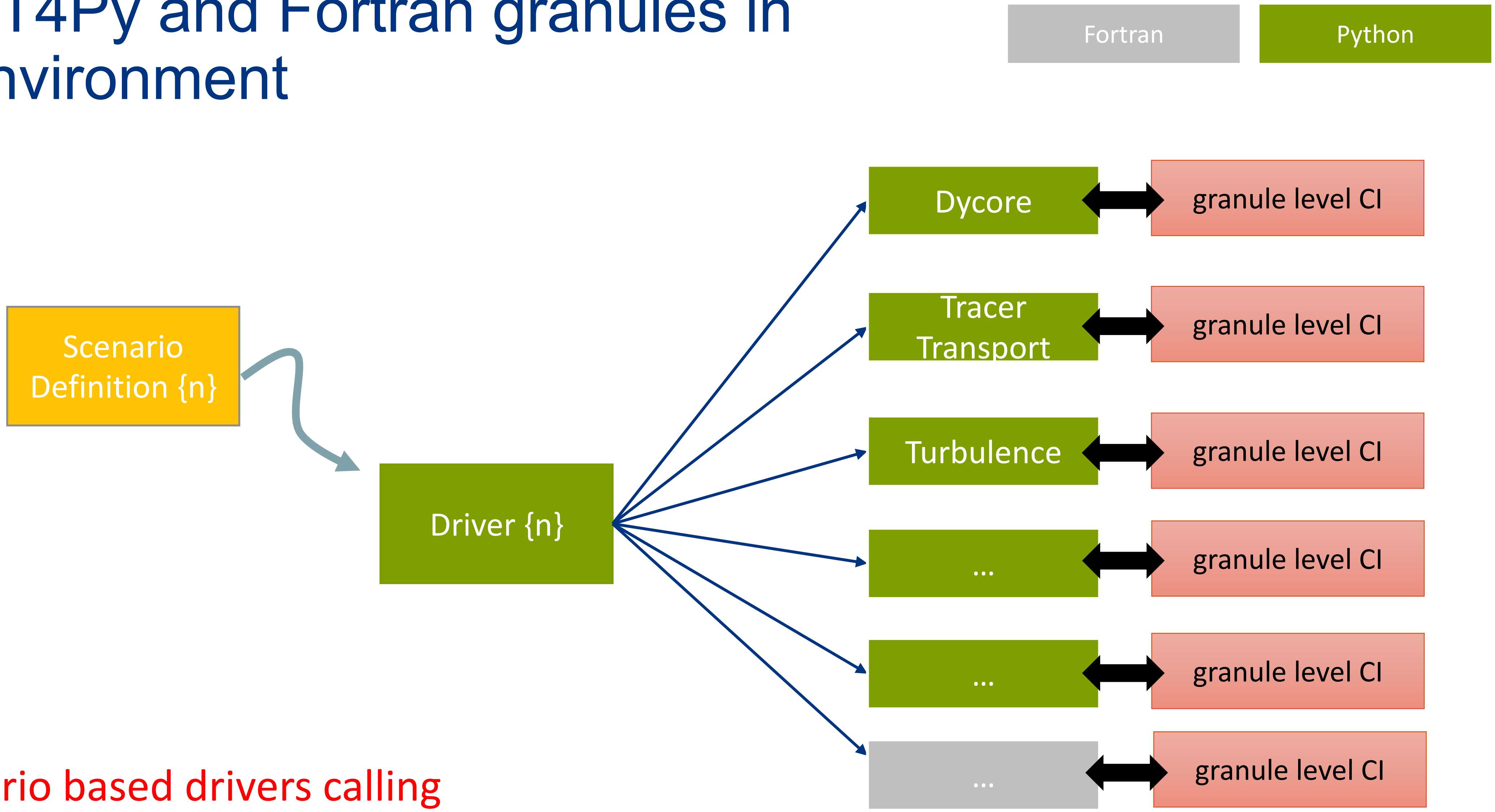
- Newer architectures are performant and (arguably) energy efficient
- Writing efficient code on them requires a good understanding of the hardware
- The resulting software is error prone

Can't blindly rely on the compiler to do the heavy lifting!

Solution

- Use a domain specific language (DSL) to hide the complexity from the user
  - We use GT4Py: a DSL embedded in Python developed at ETH
- Redesign the model into test driven granules to make it less error prone
  - Granules are independently compliable model components

Vision: GT4Py and Fortran granules in python environment



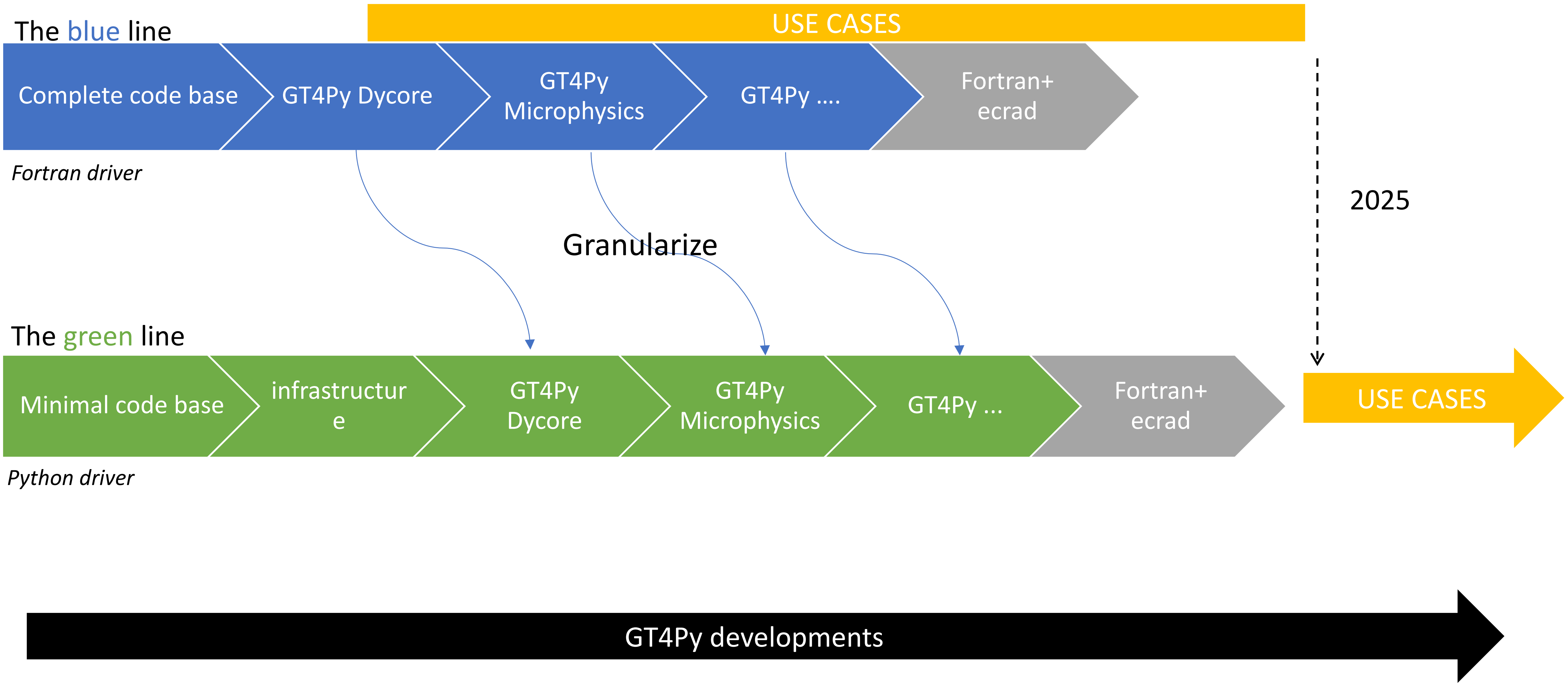
Scenario based drivers calling independently tested granules

EXCLAIM

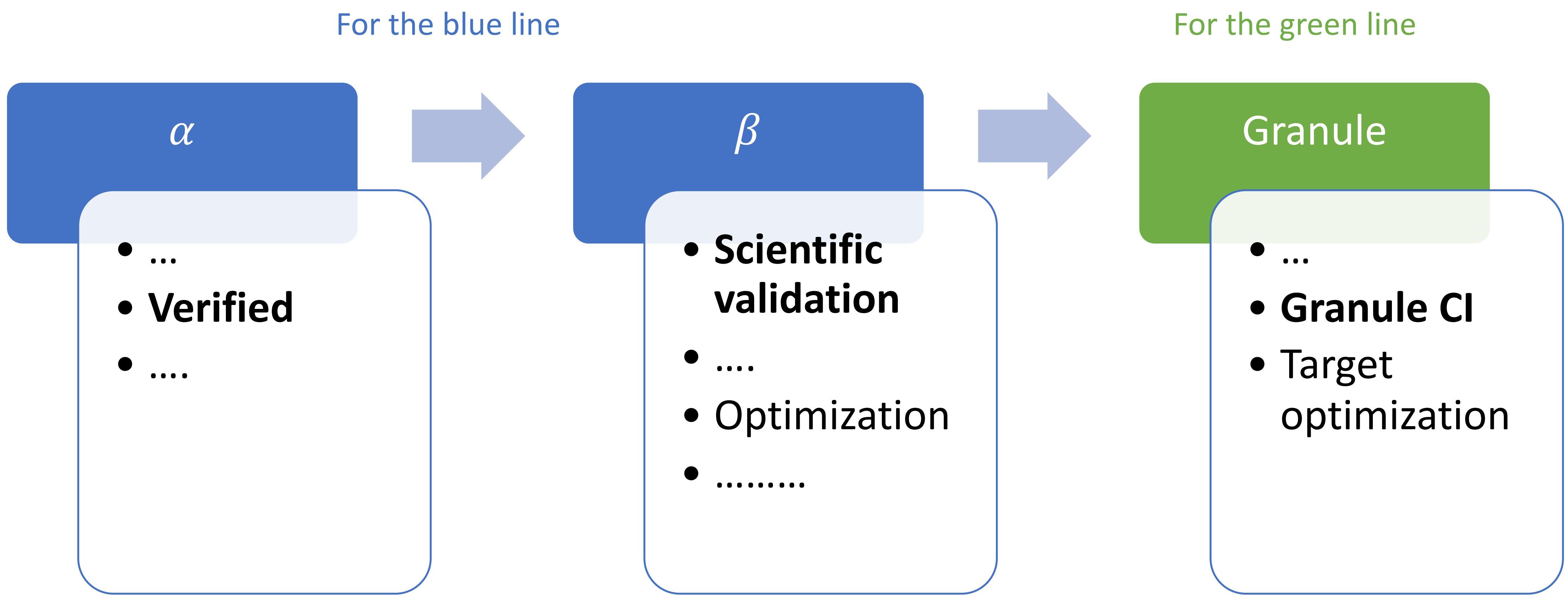
Approach

XCLAIM

Three development lines (based on ICON)



Test driven development



Status

The black line

- GT4Py v1.0.1 is available at https://pypi.org/project/gt4py/
  - Includes a stable version for Cartesian grids (module `gt4py.cartesian`)
  - Actively developed `gt4py.next` that supports both structured grids and unstructured meshes.

- `gt4py.next` next is used in EXCLAIM

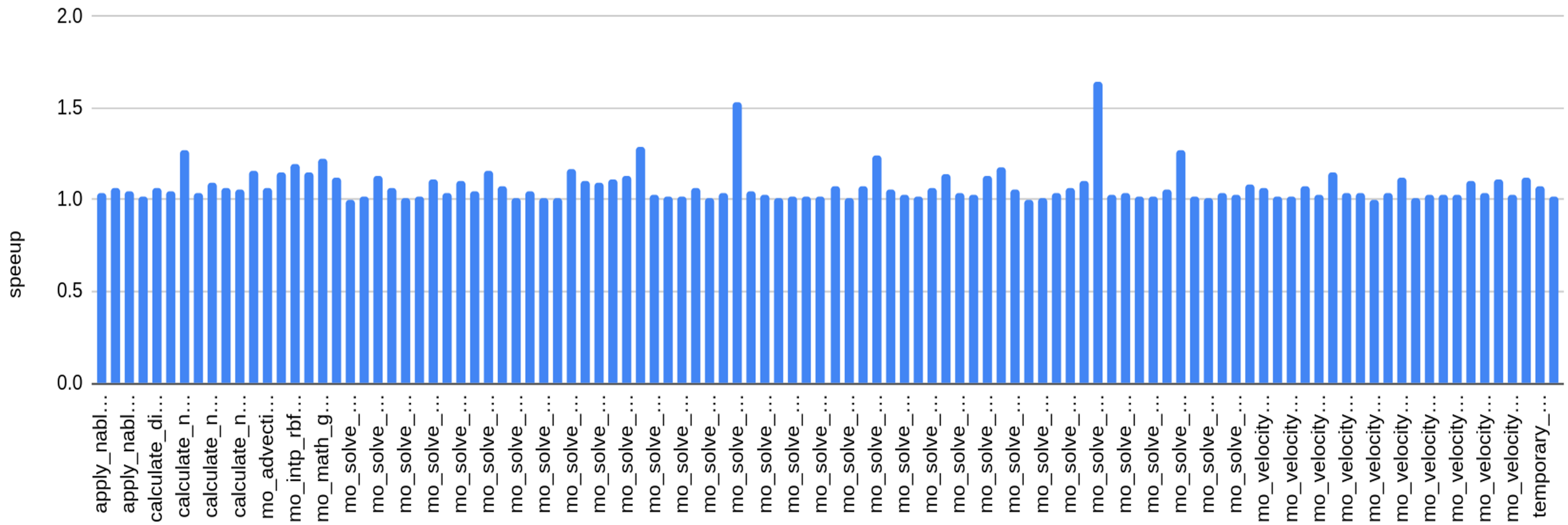
Status

- Completed
  - Declarative NumPy-like frontend
  - Feature complete for majority of ICON
  - Execution (of the optimized code) from Python

- In progress
  - Optimization
  - Debuggability

Performance (as of Jan 2023)

Dycore stencil performance at par with OpenACC on Nvdia A100 (cuda 11.8)



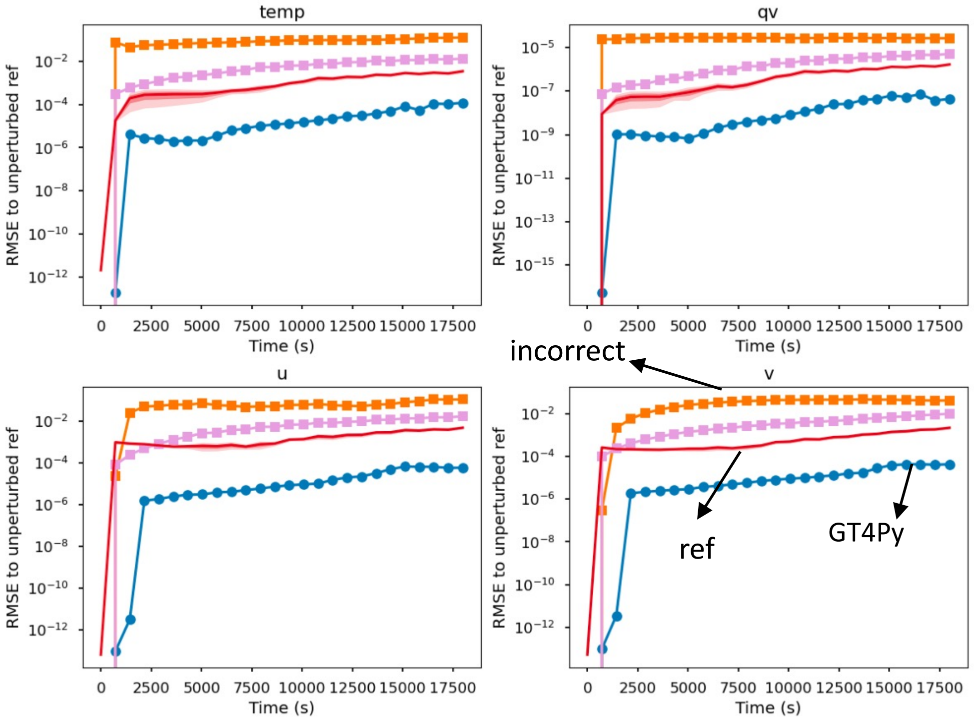
The blue line

Components	GT4Py port	Optimize	verification and validation
Dycore			
Microphysics			
Tracer advection			
…			

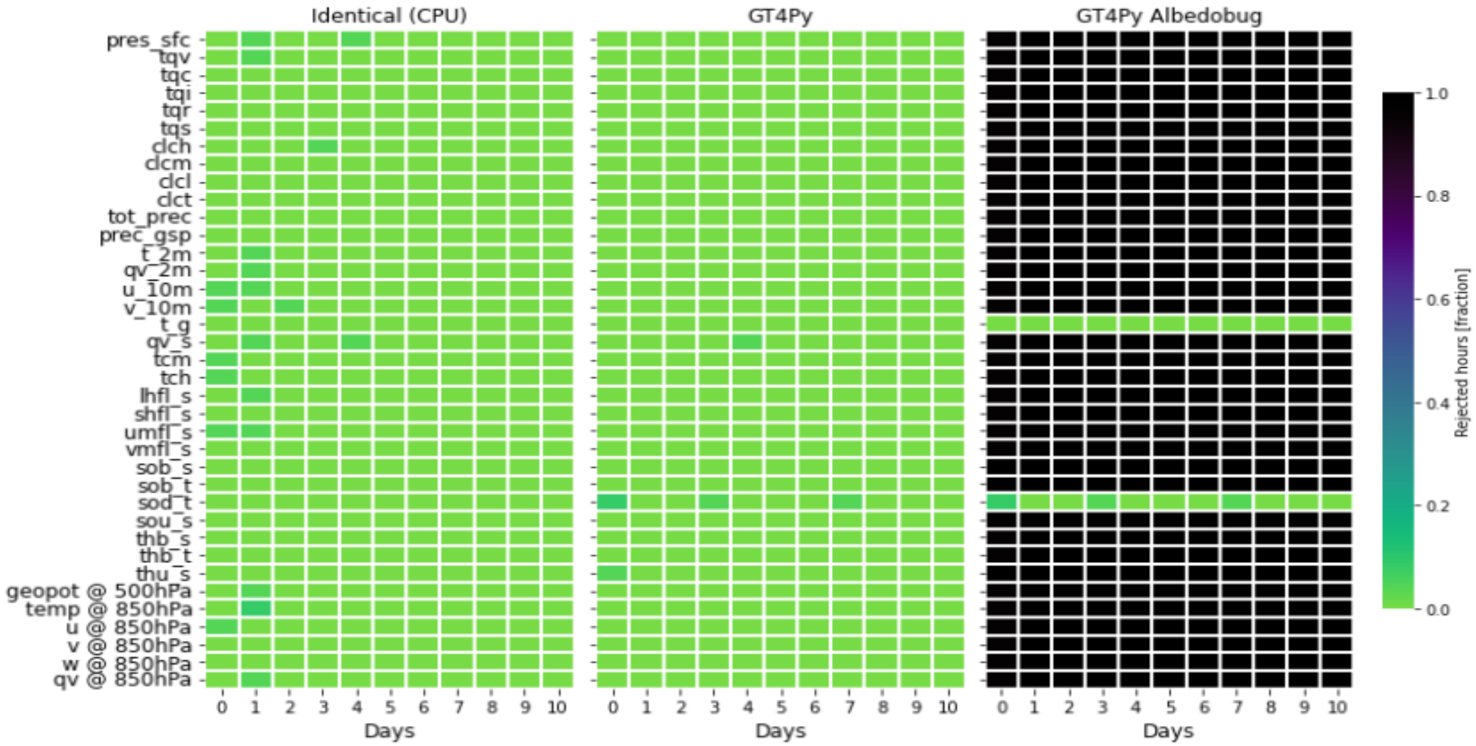
```
@scan_operator(
    axis=KDim,
    forward=True,
    init=(0.0,0.0,0.0,0),
) # inital values of state_kup variables
def _graupel_scan(
    state_kup: tuple[float64,float64,float64,int32],
    k_end: int32,
    dt: float64,
    temperature: float64,
    rho: float64,
    qc: float64,
    qr: float64,
    qnc: float64,
    prr_gsp: float64
    ) -> tuple[float64,float64,float64,int32]:

    (qr_kminus1,prr_gsp_kminus1,Vnew_r,k_lev) = state_kup

    Vnew_r, rhoqrV = velocity_precFlux(Vnew_r,rho,qr,qr_kminus1,...)

    Scaut_c2r, Scacr_c2r = autoconversion_raincollection(temperature,qc,qr,qnc)

    Cqrt = Scaut_c2r + Scacr_c2r + ...
    qr_intermediate = TV(Vnew_r,...)
    qr = maximum( 0.0 , qr_intermediate + Cqrt * dt )

    if ( k_lev == k_end ):
        # Precipitation fluxes at the ground
        prr_gsp = 0.5 * (qr * rho * Vnew_r + rhoqrV)
    else:
        prr_gsp = 0.0

    k_lev = k_lev + int32(1)

    return (qr,prr_gsp,Vnew_r,k_lev)
```

Verifying Dynamical Core

Probest (works on global averages)

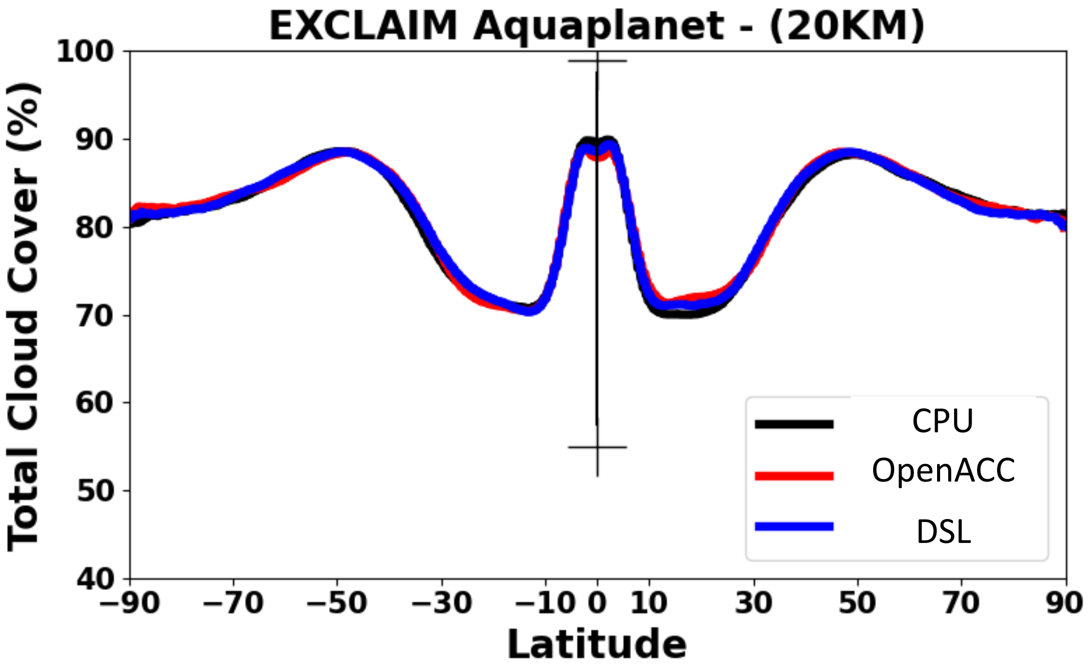


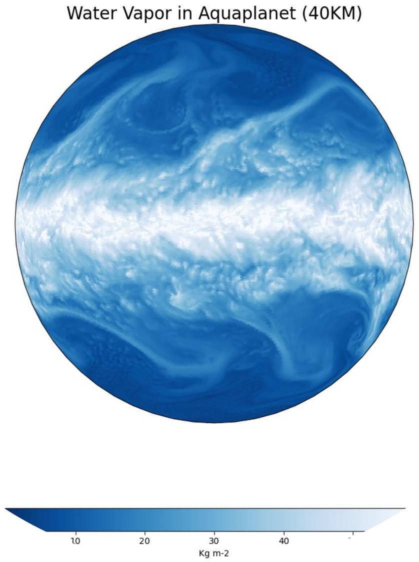
Zeman's test (works on individual grid cells)

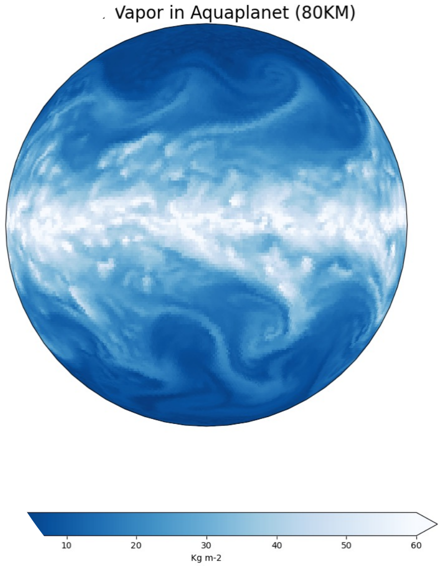


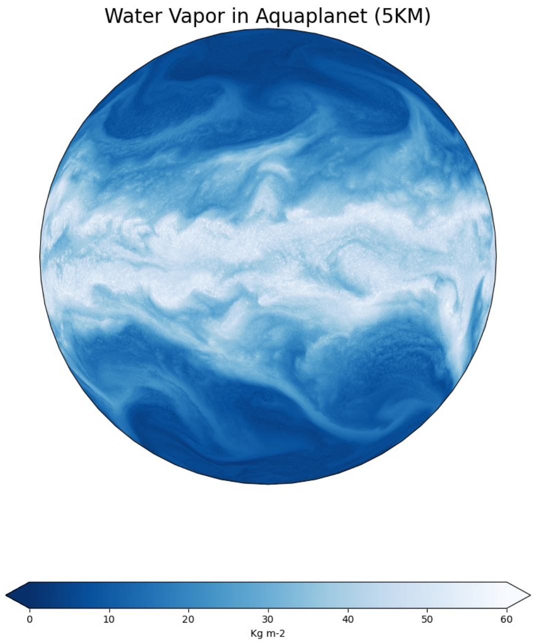
Zeman & Schär (2022)

Validating through global aquaplanet



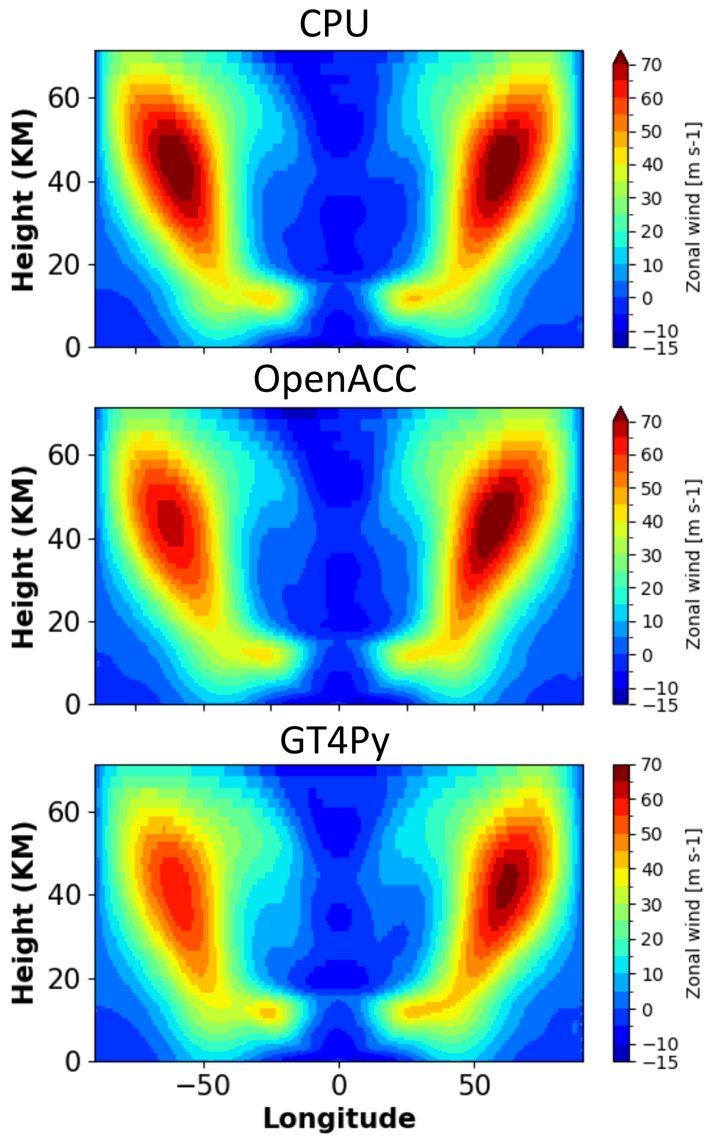






challenges …

- Several technical issues 10 km -> 5 km
- Unexplained model crash after 9 months of simulation @ 5km
- Unexplained assymetry



The green line

- The green line is all about a slim code and a python driver
- Need (optimized) granules to be called from the driver

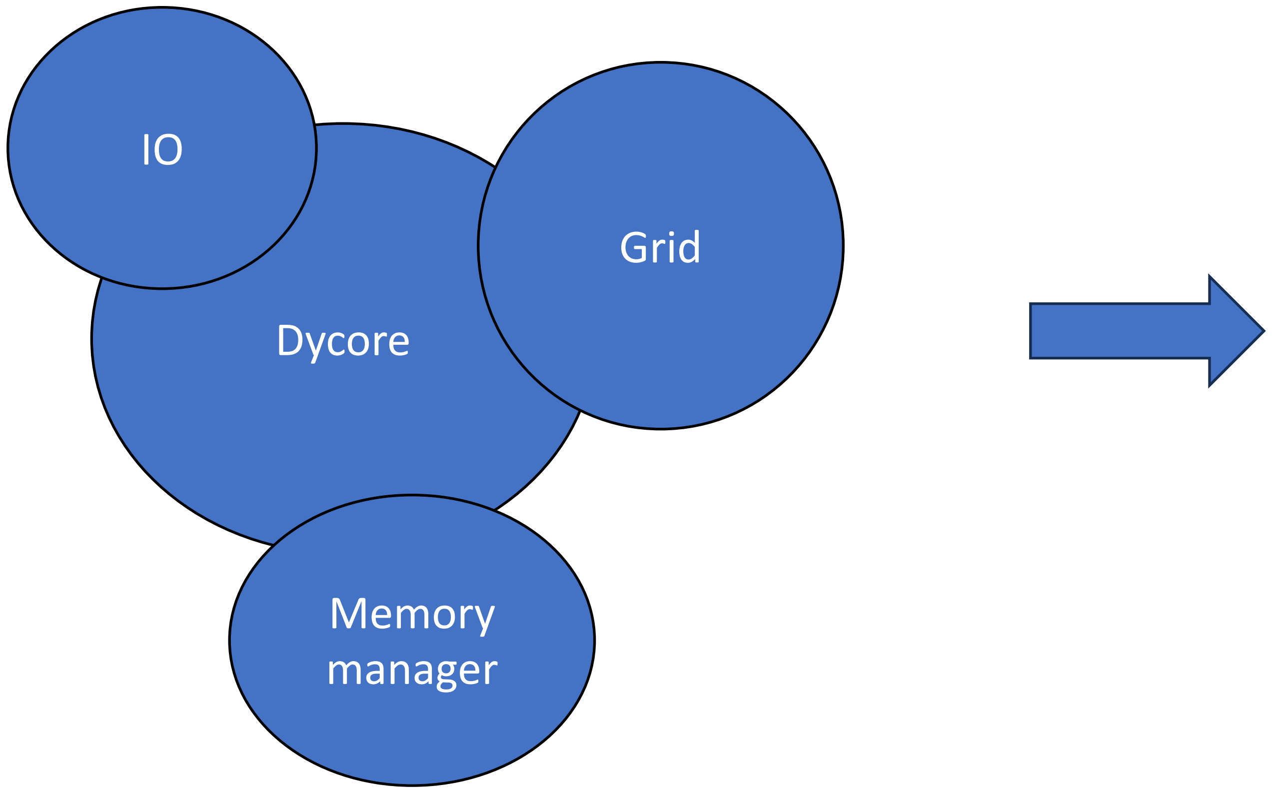
➢ As a first step, granularize components needed to drive dynamics alone on a single node (~ Q1 2024)

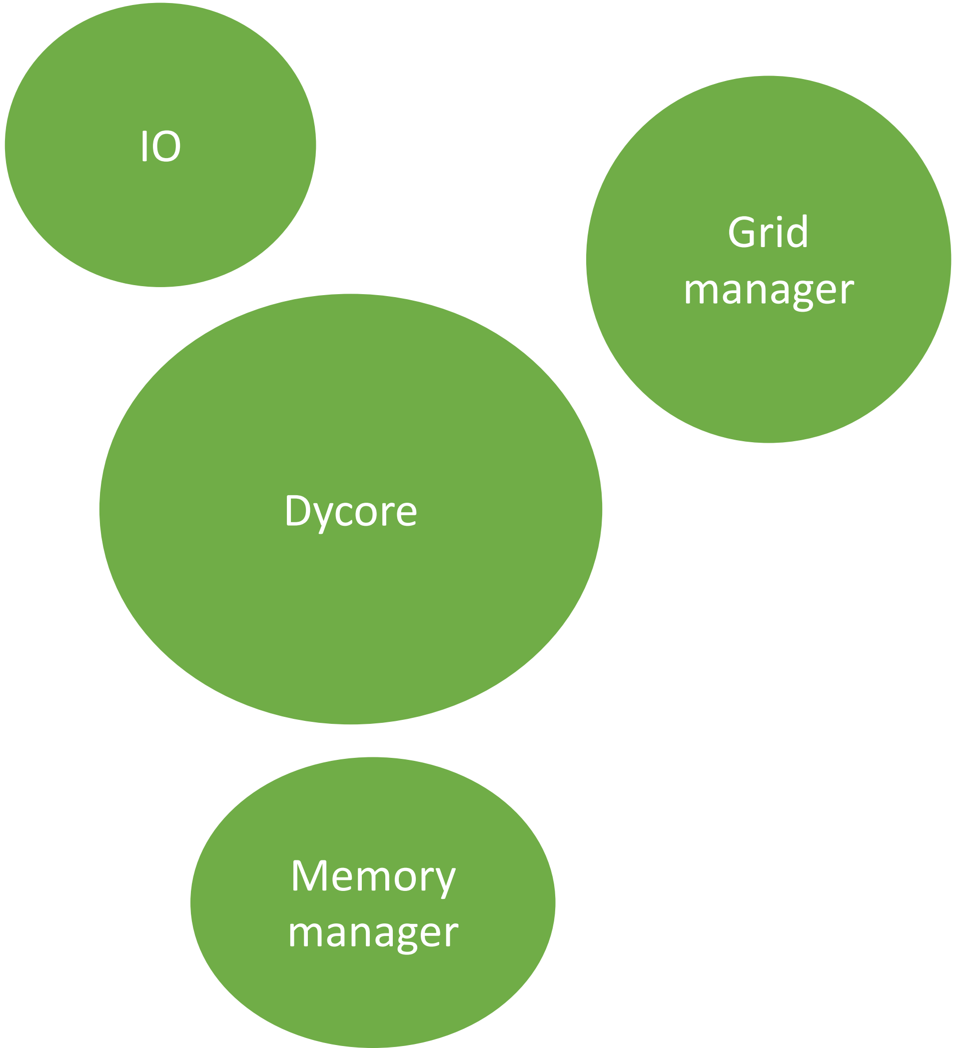
Driving dynamical core in Python

$$\frac{\partial v_n}{\partial t} + \frac{\partial K_h}{\partial n} + (\zeta + f)v_t + w\frac{\partial v_n}{\partial z} = -c_{pd}\theta_v \frac{\partial \pi}{\partial n}$$

$$\frac{\partial w}{\partial t} + \mathbf{v_h} \cdot \nabla w + w\frac{\partial w}{\partial z} = -c_{pd}\theta_v \frac{\partial \pi}{\partial z} - g,$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\mathbf{v}\rho) = 0,$$





Dycores are intrinsically tied to the model infrastructure

The green line : Status

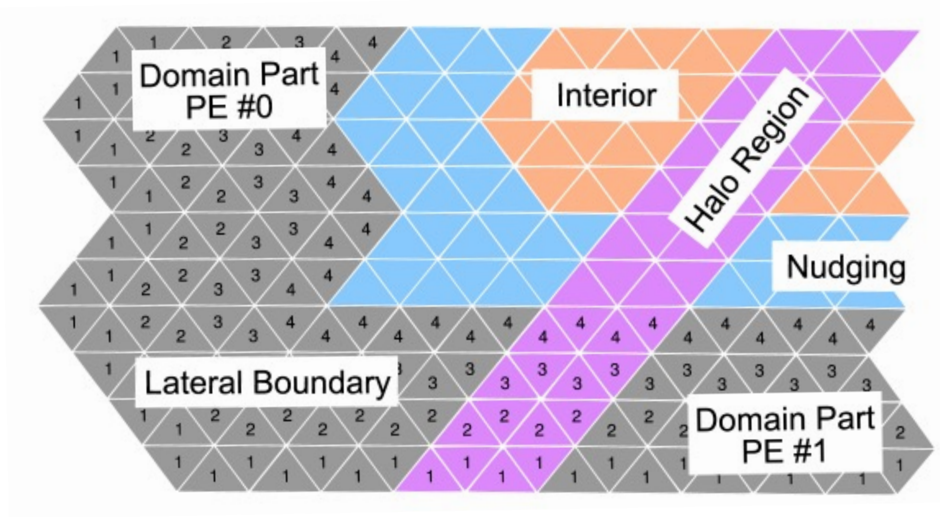
Components	Granularize	Optimize	Granule CI
infrastructure			
Dycore (+diffusion)			
Microphysics			
Tracer advection			
…			

Challenges

ICON carries its grid connectivity (including decomposition and communicators) in a complex derived data type `t_patch` down to innermost loops. Standalone granules (dycore, diffusion, advection) need this.

Solution: *Grid Manager* library

- Reads and Distributes basic key grid information (e.g., vertex positions)
- Calculates derived grid information (e.g., neighbors) on the fly
- Provides multi-language accessor functions (Fortran, C, Python)
- The application can manage fields, e.g., with a data dictionary, avoiding frequent recalculation



Current status: adopted a Grid Manager module from the DWD discontinuous Galerkin “BRIDGE” code

- Introduced interface to specify *predefined decomposition* (e.g., from ICON, MeTiS or another tool)
- Minimal grid connectivity information *extended for ICON diffusion granule requirements*
- Interfacing the drivrer: *Python → thin C layer → Fortran BIND(C) → Fortran 2008 Grid Manager*

Summary

Global cloud resolving simulations in a python framework

Performance

User experience

Overall model performance

- A throughput of ~ 0.4 Simulation Year Per Day at @ 5 km on 1000 Piz Daint nodes

- Optimization in progress

- Grand challenge for 2024: Global APE @ 1 km for > 2 years using the blue line

User experience

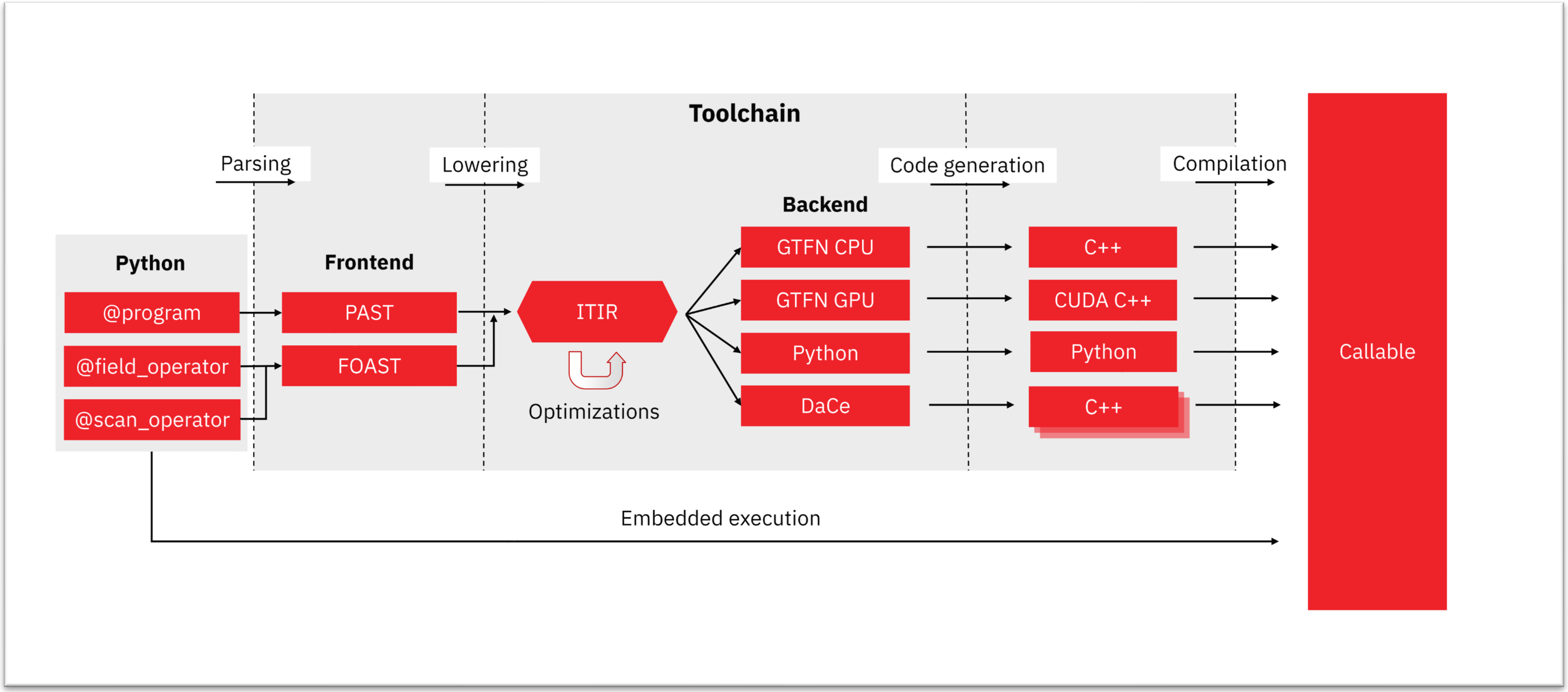
Chia Rui Ong

- Long experience in writing Fortran codes
- Tasked to port the microphysics in GT4Py
- 3 weeks later: its smooth!

Thank you!

A note about GT4Py: a python framework to develop weather/climate models

GridTools 4python



ECMWF

Ai2