Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement des Innern EDI
Bundesamt für Meteorologie und Klimatologie MeteoSchweiz

Porting the MeteoSwiss weather forecast software to GPUs

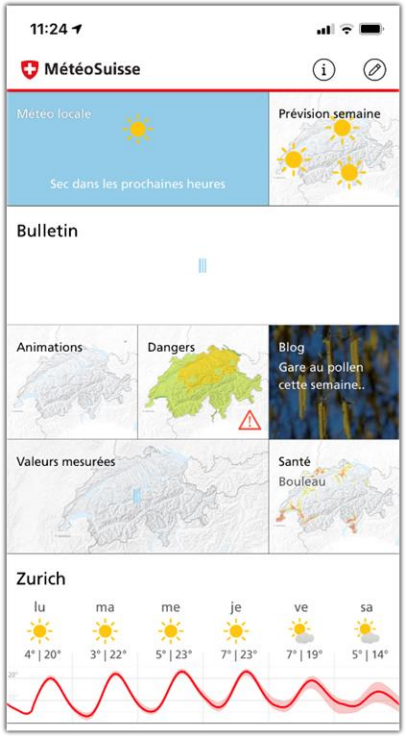
X. Lapillonne[1], C. Osuna[1], **D. Hupp[1]**, D. Alexeev[5], V. Cherkas[1], R. Dietlicher[1], E. Germann[1], F. Gessler[1], M. Jacob[4], A. Jocksch[3], J. Jucker[2], C. Müller[1], M. Röthlin[1], W. Sawyer[3], U. Schättler[4], André Walser[1]

[1]MeteoSwiss, [2]C2SM, [3]CSCS, [4]DWD, [5]Nvidia

09.11.2022, CERN, Computer Accelerator Forum

MeteoSwiss

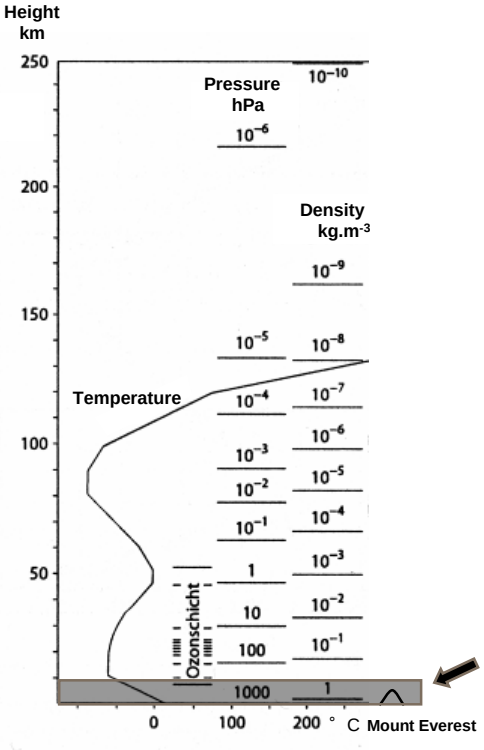
- Federal Office of Meteorology and Climatology
- Weather forecasting, warnings : population, authorities, Aviation ..
- Numerical Weather prediction
- Observation : station, radar, …
- Climatology
- Climate scenario and impact



MeteoSwiss App

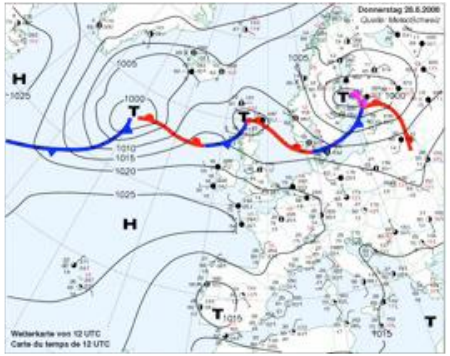
Numerical weather prediction: The atmosphere

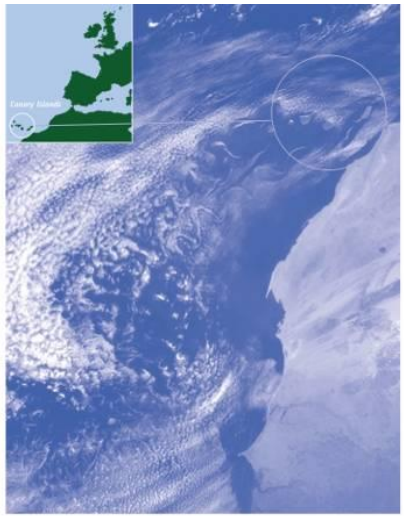
A thin gas layer around the earth



Troposphere:
in the order of 10km thickness
90% of air mass
99% of air humiduty

Numerical weather prediction: Multiscale





Meteosat-7 visible channel image, 10.00h UTC, 4 April 2001

source: meteoswiss

From 1 000 km to mm

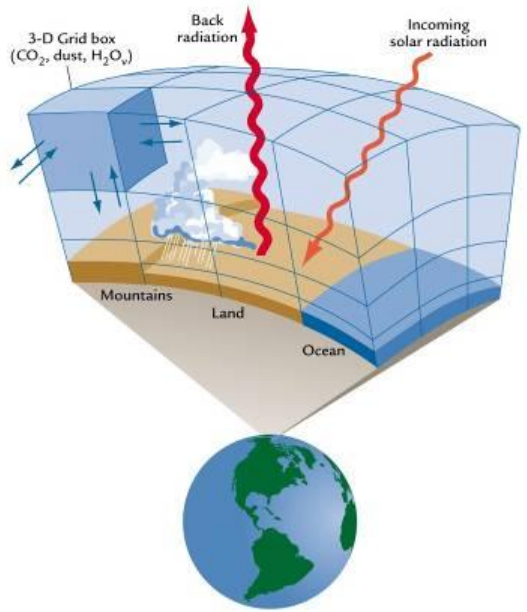
Numerical weather prediction: Atmospheric modelling

Fluid motion and thermodynamics processes
(Primitive equations : dynamics)

+

Subgrid scale processes
(physical parametrizations)

Discretised on a 3d grid



Atmospheric model (Ruddiman, 2001)

The governing equations

Momentum

$$\rho \frac{d\mathbf{v}}{dt} = -\nabla p + \rho \mathbf{g} - 2\boldsymbol{\Omega} \times (\rho \mathbf{v}) - \nabla \cdot \underline{\mathbf{t}}$$

Mass

$$\frac{d\rho}{dt} = -\rho \nabla \cdot \mathbf{v}$$

Heat

$$\rho c_v \frac{dT}{dt} = -p \nabla \cdot \mathbf{v} + Q_h + Q_m$$

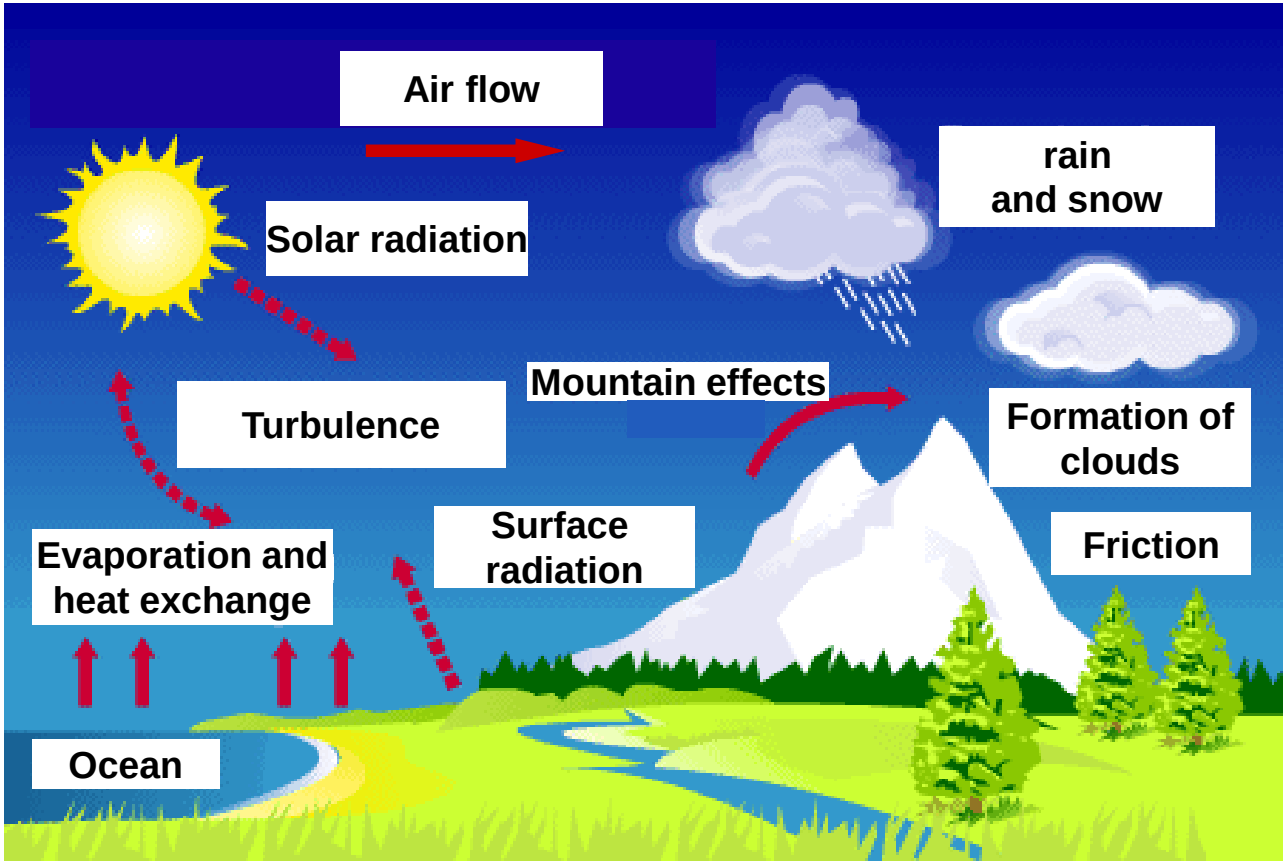
Vapor/Liquid/ solid water

$$\rho \frac{dq^x}{dt} = -\nabla \cdot \mathbf{J}^x + I^x$$

pressure

$$p = \rho R_d (1 + \alpha) T.$$

Atmospheric processes parametrizations

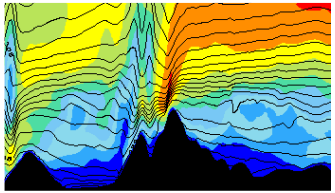


What do we need for numerical weather prediction ?

Atmospheric model = Dynamic + Parametrization

Discretization on a finite grid:

→ Numerical model



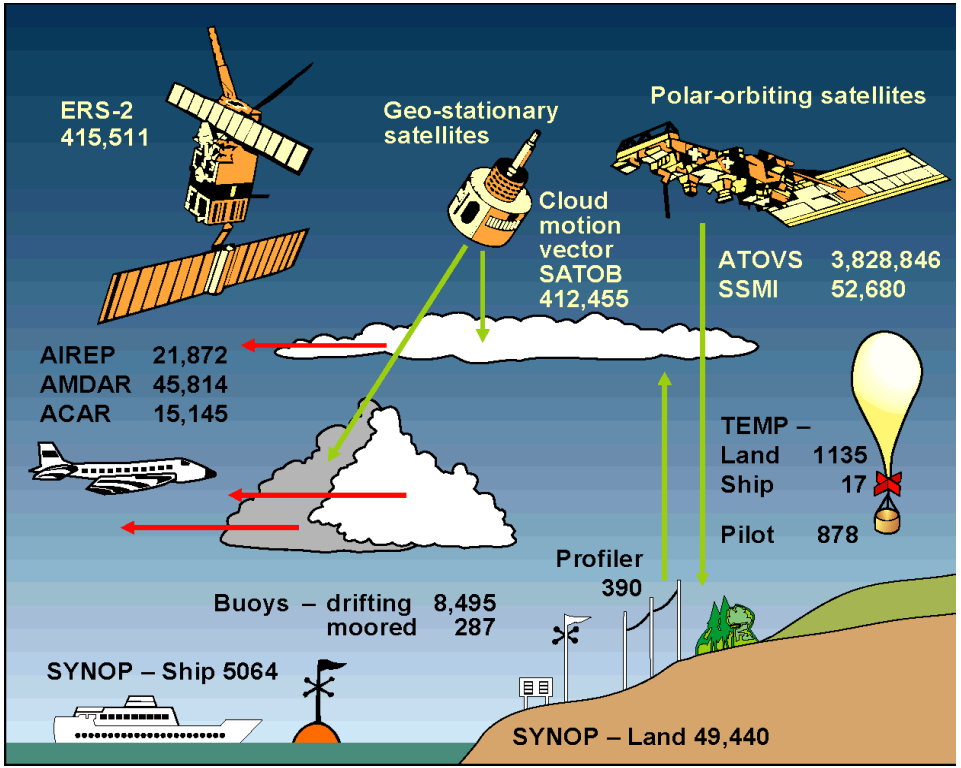
Knowledge of the initial state of the atmosphere

→ observations

The process of bringing observations to the numerical model is referred to as data assimilation

Observing the atmosphere

Initial state:
model+observation

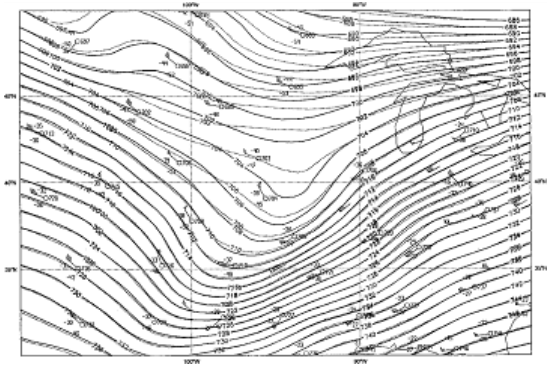


@ECMWF 2001

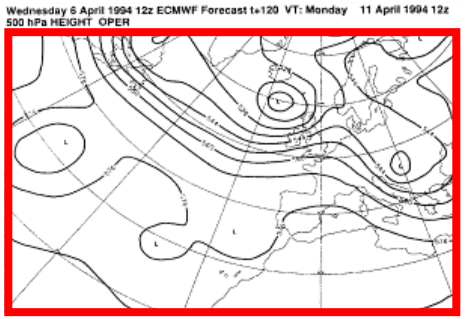
Chaos theory and predictability

The atmospheric system is a chaotic system:
i.e. high sensitivity to initial condition

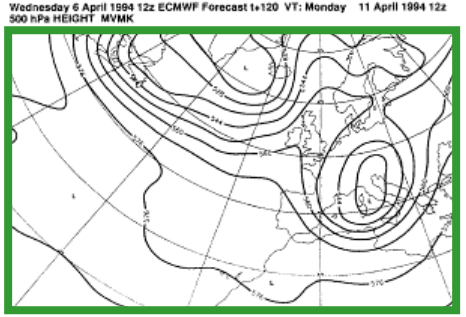
Initial conditions (dotted lines)



Initial conditions (full lines)



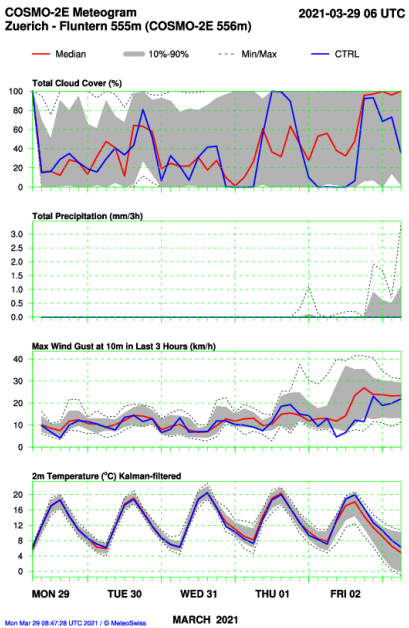
Operational 120 h forecast

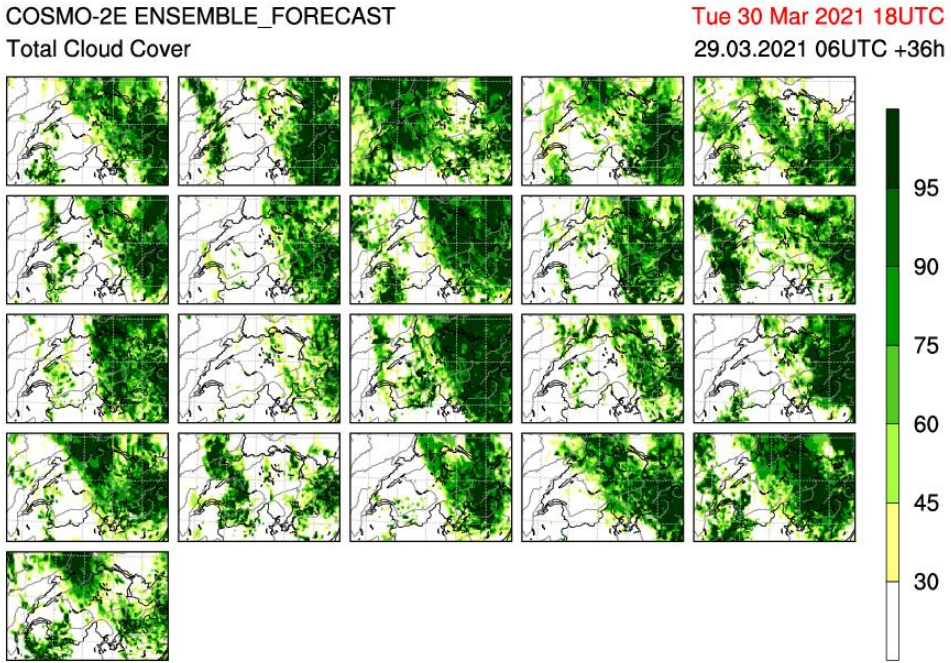


Modified forecast

Ensemble prediction

- Run multiple simulations with different initial condition and with perturbations





High Perormance Computing (HPC)

- NWP is a time critical problem
- Large computing requirement, grid, ensembles …
- Need to adapt to latest advances in computer architecture
- Usually runs on dedicated HPC systems

Cray XC-30 Hybrid Piz Daint at CSCS (5,272 GPUs)

GPU computing at MeteoSwiss

- GPU : Graphical Processing Units
- Most of the fastest supercomputers (Top500) are based on accelerator/many-core technologies
- Moore’s law is ending, what do we do ? => Increase on chip parallelism
- Take advantage of the high computational capacity and energy efficiency of GPUs
- MeteoSwiss was the first National Weather Service to use GPUs for production in 2016

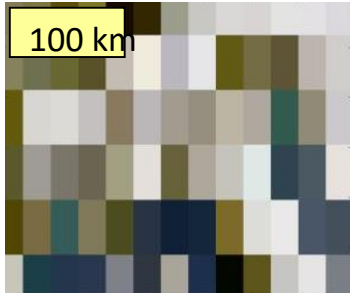
GPU

CPU

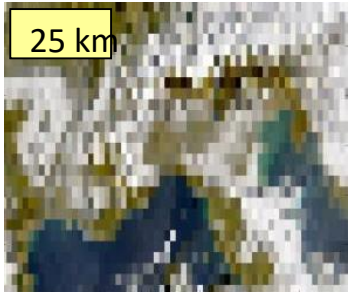
MeteoSwiss Cray CS-Storm
Hybrid GPU system at CSCS

What can we do with more computing power ?

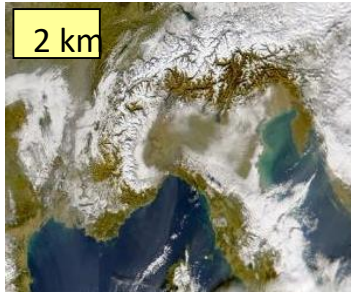
- Improve model representation and physical processes



Global climate model



Global weather model



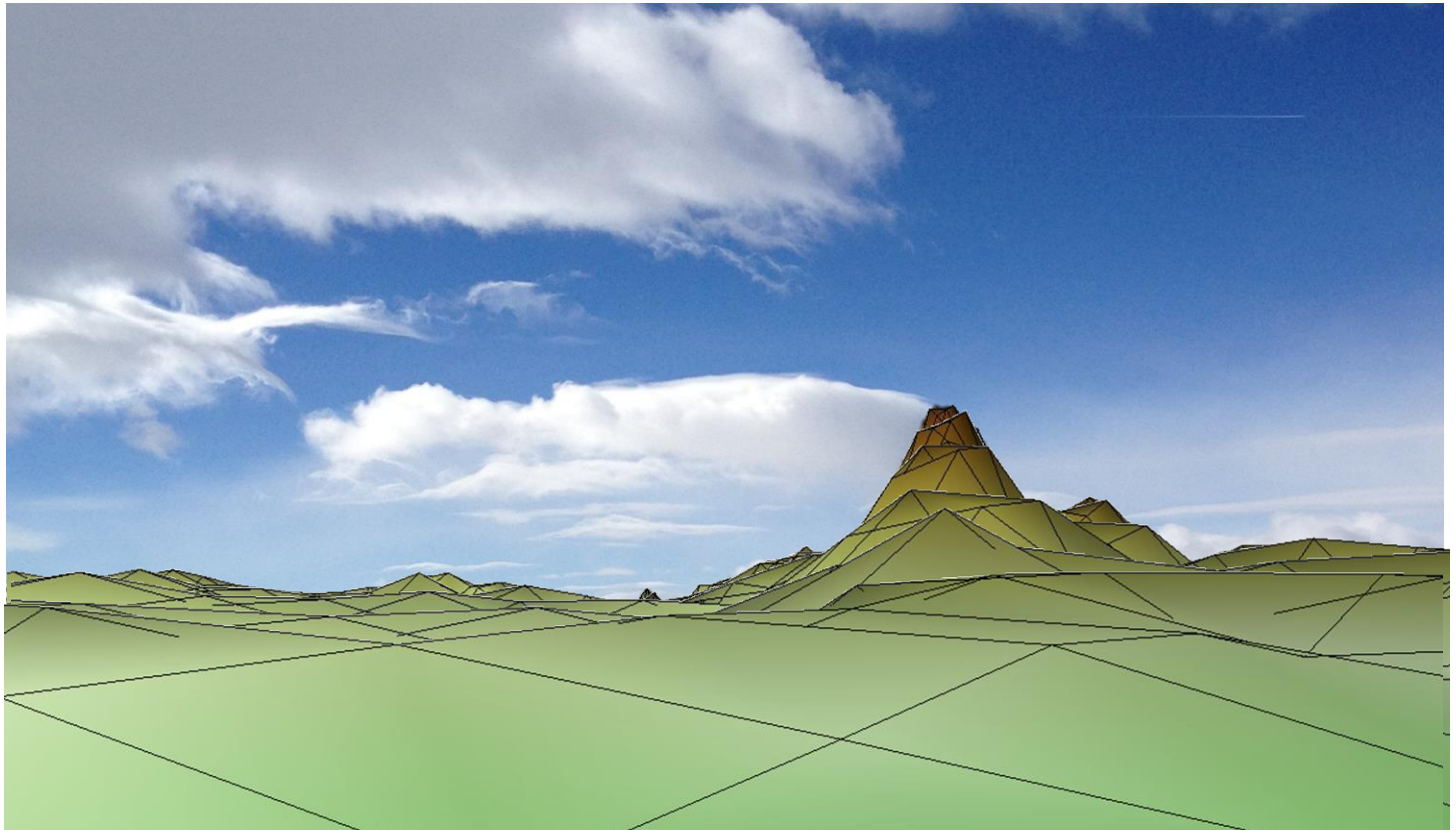
Regional weather model

2 x horizontal resolution ≈ 8 x computational cost

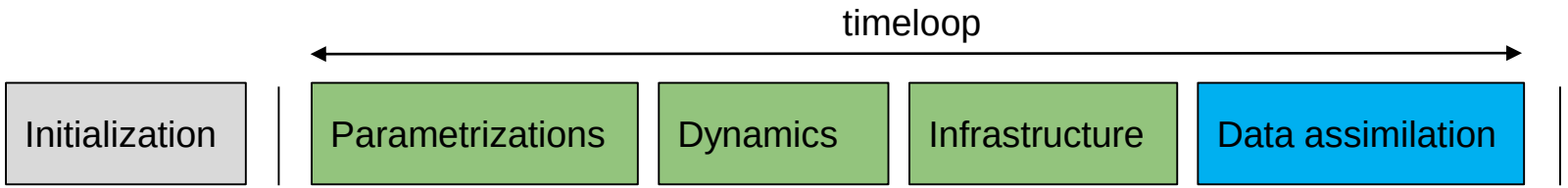
- Increase number of ensemble: better evaluation of predictability

Resolution of the computational grid

$\Delta x = 2200$ m



Numerical weather prediction using OpenACC



- Most components for NWP Regional and global application ported, optimization work ongoing
- Support for both double and mixed precision
- Regular testing on builbot infrastructure

Turbulence

Optimizations applied

- Apply ASYNC clause as much as possible
- Collapse purely nested k- and i-loops
- Fuse i-loops inside k-loop
- Replace !$ACC DATA PRESENT with DEFAULT(PRESENT) at kernel-level
- Applying GANG(STATIC:1)

Source: Jonas Jucker

Problem of many small GPU Kernels

```
…
!$ACC PARALLEL LOOP GANG VECTOR ASYNC(1) DEFAULT(NONE) IF( lzacc )
DO i=ivstart, ivend
   umfl_s(i)=rhon(i,ke1)*tkvm(i,ke1)*zvari(i,ke1,u_m)
END DO

!$ACC PARALLEL LOOP GANG VECTOR ASYNC(1) DEFAULT(NONE) IF( lzacc )
DO i=ivstart, ivend
   vmfl_s(i)=rhon(i,ke1)*tkvm(i,ke1)*zvari(i,ke1,v_m)
END DO

!$ACC PARALLEL LOOP GANG VECTOR ASYNC(1) DEFAULT(NONE) IF( lzacc )
DO i=ivstart, ivend
   edr(i,ke1)=tke(i,ke1,ntur)**3/(d_m*l_tur_z0(i))
END DO
```

These lead to 3 individual kernels with the overhead of launching 3 kernels, for a very small compute time.

Source: Victoria Cherkas

Problem of many small GPU Kernels

The problem we need to solve within turbulence.

Source: Victoria Cherkas

Solution: Bundling loops into single GPU kernel

```
!$ACC PARALLEL ASYNC(1) DEFAULT(NONE) IF( lzacc )
…
!$ACC LOOP GANG(STATIC:1) VECTOR
DO i=ivstart, ivend
   umfl_s(i)=rhon(i,ke1)*tkvm(i,ke1)*zvari(i,ke1,u_m)
END DO

!$ACC LOOP GANG(STATIC:1) VECTOR
DO i=ivstart, ivend
   vmfl_s(i)=rhon(i,ke1)*tkvm(i,ke1)*zvari(i,ke1,v_m)
END DO

!$ACC LOOP GANG(STATIC:1) VECTOR
DO i=ivstart, ivend
   edr(i,ke1)=tke(i,ke1,ntur)**3/(d_m*l_tur_z0(i))
END DO

!$ACC END PARALLEL !!
```

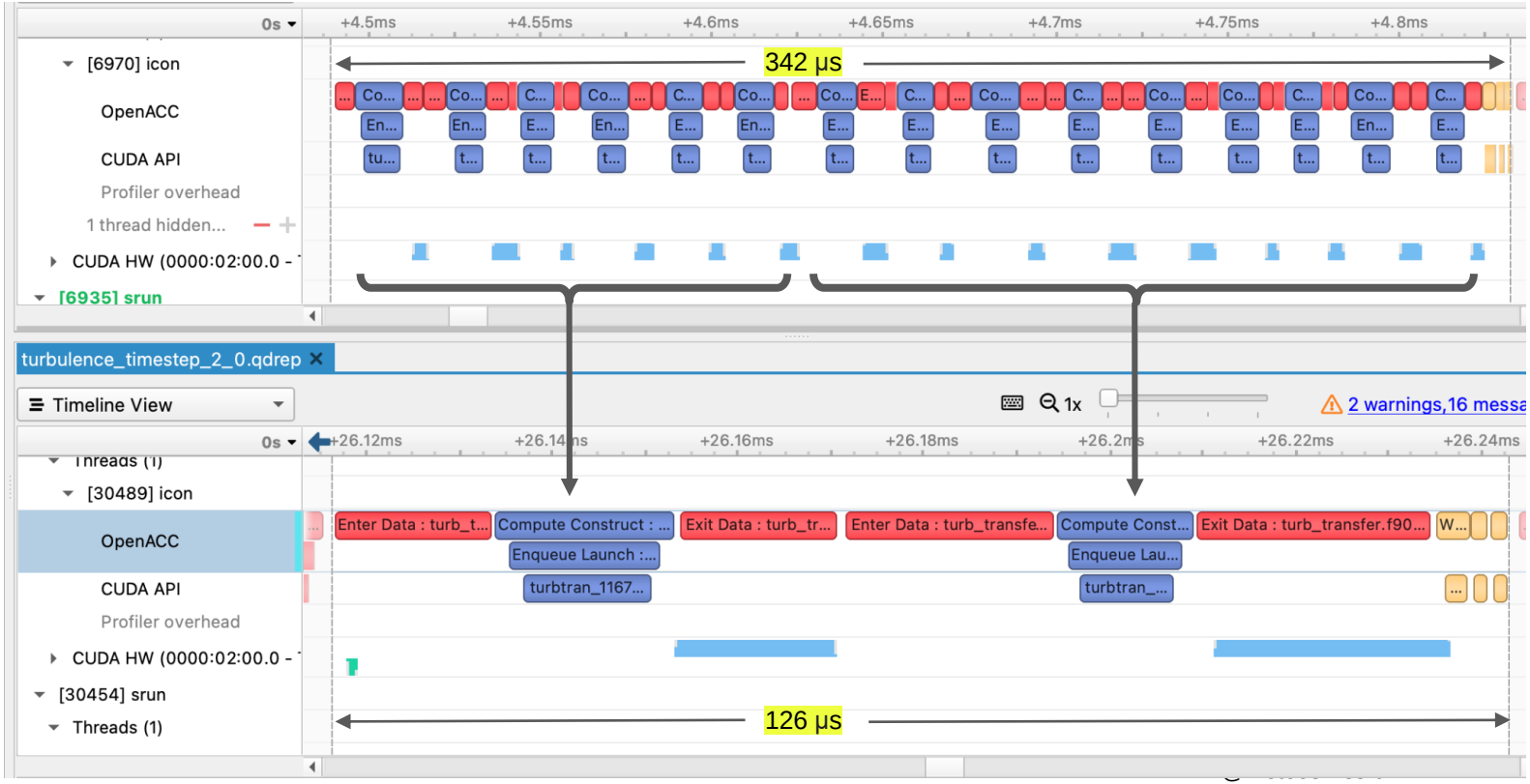
Using gang(static:1) and a single parallel region, we effectively have **one fused kernel** without the breaks in between.

- Improves cache reuse
- Reduces launch overhead

Notice loop bounds are same for the 3 loops.

Source: Victoria Cherkas

Before and after profile comparison



Source: Victoria Cherkas

Porting and optimization challenges

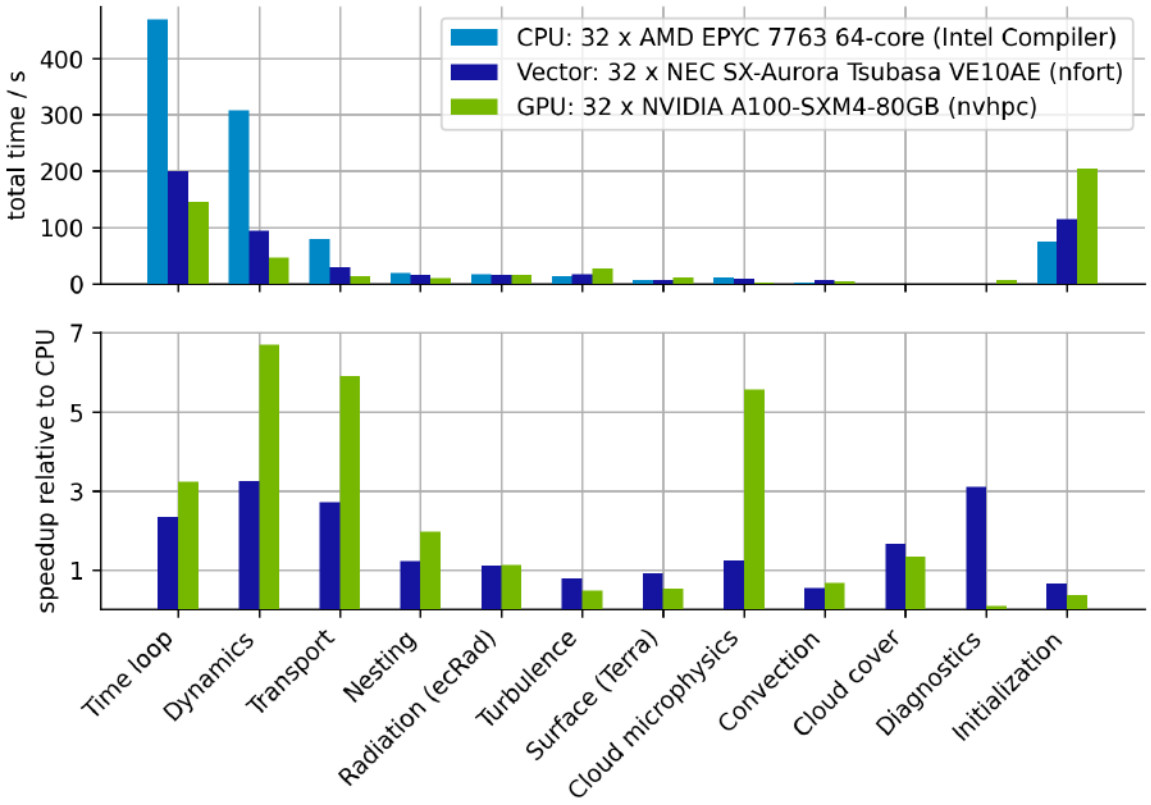
OpenACC optimizations

- GPU and CPU working asynchronously
  - Reduces launch overhead
- Bundling similar loop constructs into single GPU kernels
  - Improves cache reuse
  - Reduces launch overhead
- Compiler assisted / manual inlining of function calls
  - Required for complex (deep call-trees) GPU kernels
  - Enables optimizations above

Conceptual challenges

- Code management
  - Disruptive code changes are challenging
  - ecrad: juggling diverse Institutes

Current performance



CPU vs GPU vs NEC
Experiment:

- ~DWD deterministic forecast Global R2B**8**
- 5 242 880 cells (10 km) + Nested grid over Europe 845 340 cells (5 km)
- Output disabled

Performance and energy

- GPU 3x faster than CPU
- 32 GPUs: 663 s/day 268 Wh/day
- 32 VEs: 914 s/day 292 Wh/day

Source: Marek Jacob, DWD, using ICON branch with ecRad on GPU

Why using Domain Specific Language (DSL) in weather and climate ?

- DSL : computer language restricted to a particular domain
- We need performance to reach time to solution
- Separation of concern between domain and computer scientist
- Single source code for multiple target architectures
- Possible to write a new backend when a new technology emerged
- Allow aggressive optimization without degrading readability of user code
- Allow optimization across components – data centric optimization

High level DSL for ICON

- Need to support unstructured grid, such as ICON grid
- New abstraction (e.g. neighbors operations)
- Focus on usability, productivity. Should be usable for domain scientist
- High level python dsl (gt4py)
- Development work in the EXCALIM project
  - High resolution use cases
  - Re-write code components using python DSL framework

Motivation

$$\nabla_{\underline{n}}\psi(e) = \frac{\psi(c_1(e)) - \psi(c_0(e))}{\hat{l}}$$

```
#ifdef _OMP
!$OMP ....
#else
!$ACC ....
#endif
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 #ifdef __LOOP_EXCHANGE
 DO je = i_startidx, i_endidx
   DO jk = slev, elev
 #else
   DO jk = slev, elev
     DO je = i_startidx, i_endidx
 #endif
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
#ifdef _OMP
!$OMP ...
#else
!$ACC ...
#endif
```

Motivation

$$\nabla_{\underline{n}}\psi(e) = \frac{\psi(c_1(e)) - \psi(c_0(e))}{\hat{l}}$$

```
grad_norm_psi_e = neighbor_sum(
        psi_c,
        axis=E2CDim,
        weights=[-1, 1]
)/lhat
```

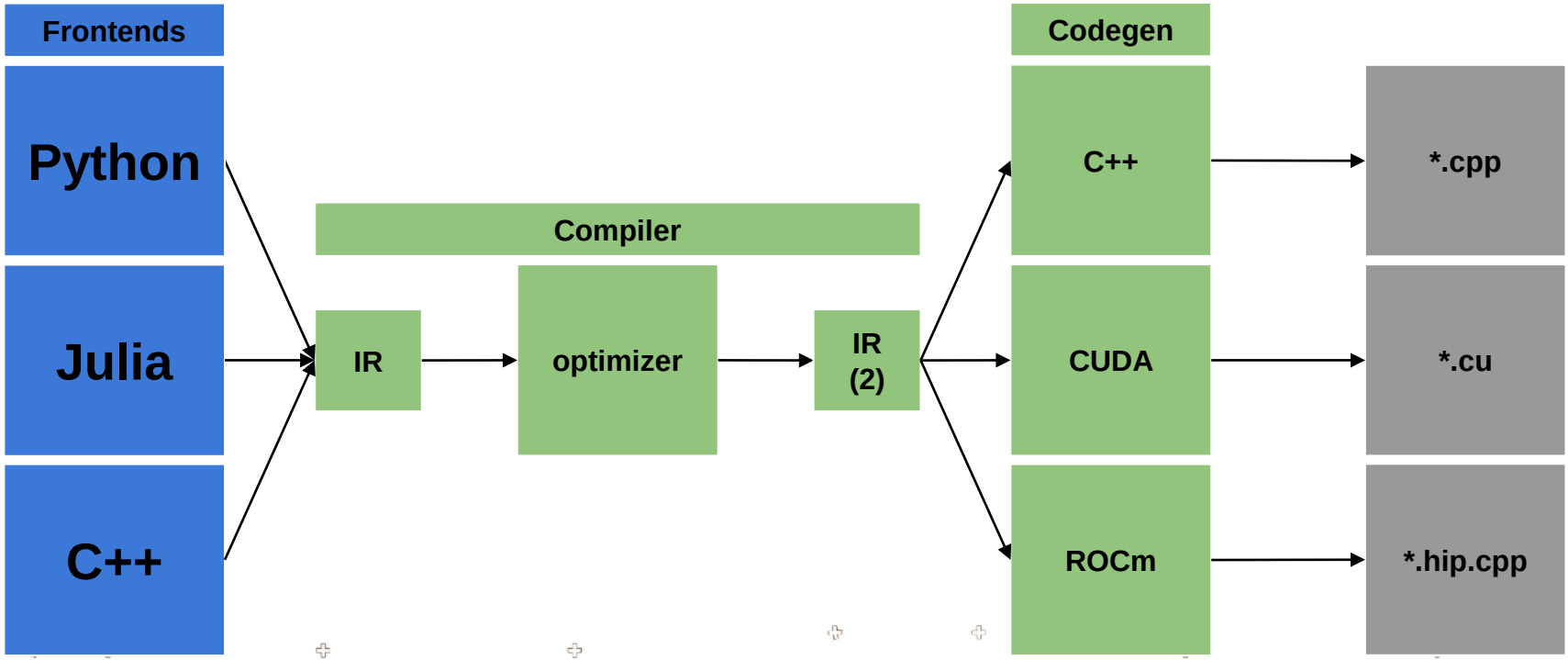
```
#ifdef _OMP
!$OMP ....
#else
!$ACC ....
#endif
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 #ifdef __LOOP_EXCHANGE
 DO je = i_startidx, i_endidx
   DO jk = slev, elev
 #else
   DO jk = slev, elev
     DO je = i_startidx, i_endidx
 #endif
      grad_norm_psi_e(je,jk,jb) =  &
        ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
          psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
        / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
#ifdef _OMP
!$OMP ...
#else
!$ACC ...
#endif
```

High level DSL for ICON

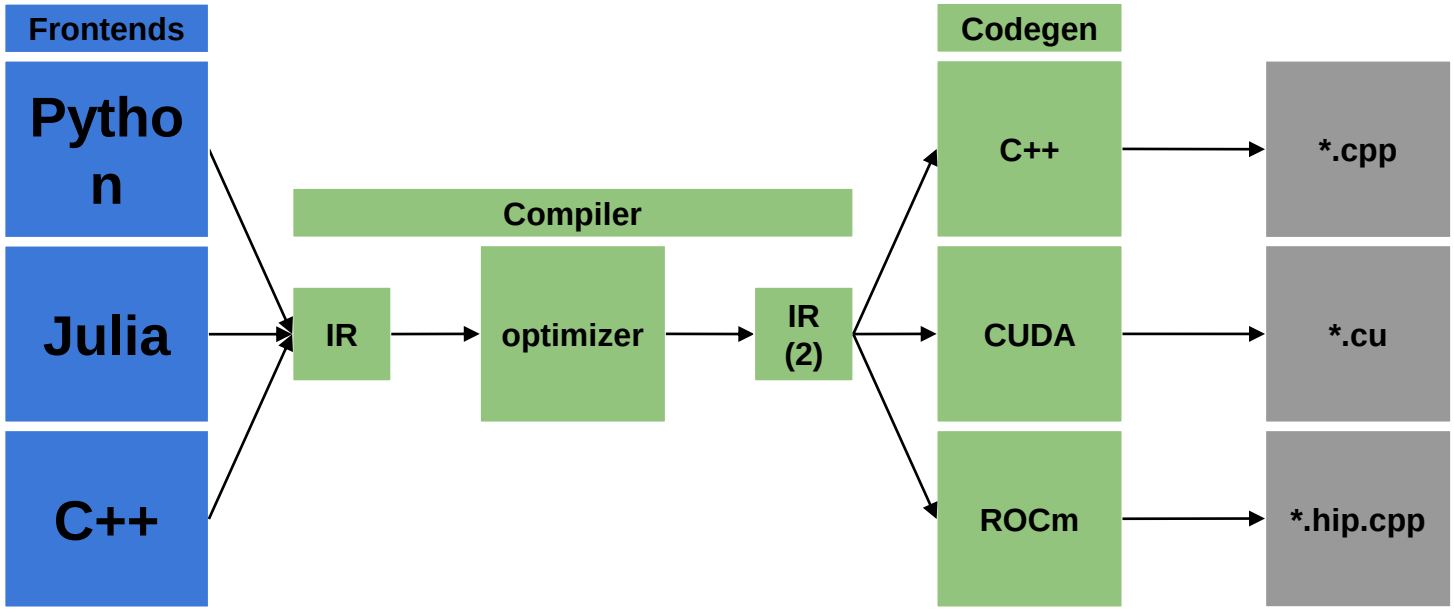
- Need to support unstructured grid, such as ICON grid
- New abstraction (e.g. neighbors operations)
- Focus on usability, productivity. Should be usable for domain scientist
- High level python dsl (gt4py)
- Development work in several projects, ESCAPE, EXCLAIM(ETHZ)

- Performance :
  - CUDA code generation for GPU : allow more optimization than OpenACC
  - long term perspective - data centric optimization across components, e.g. fusion …

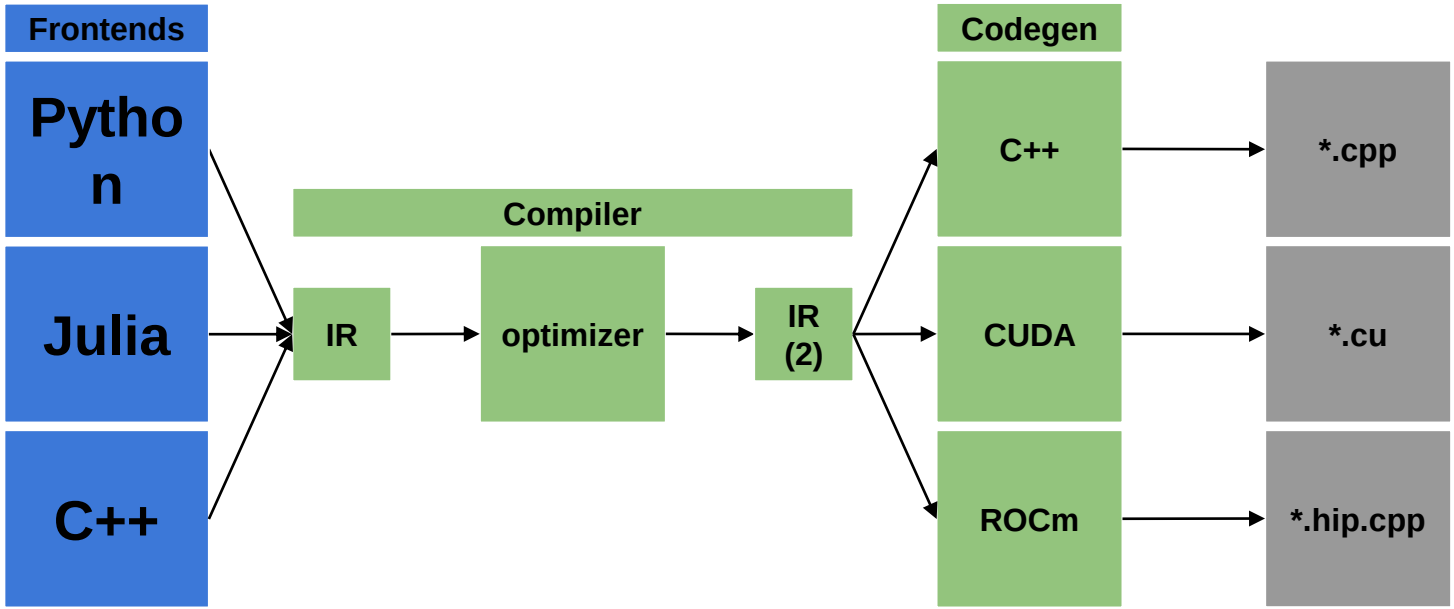
General DSL toolchain



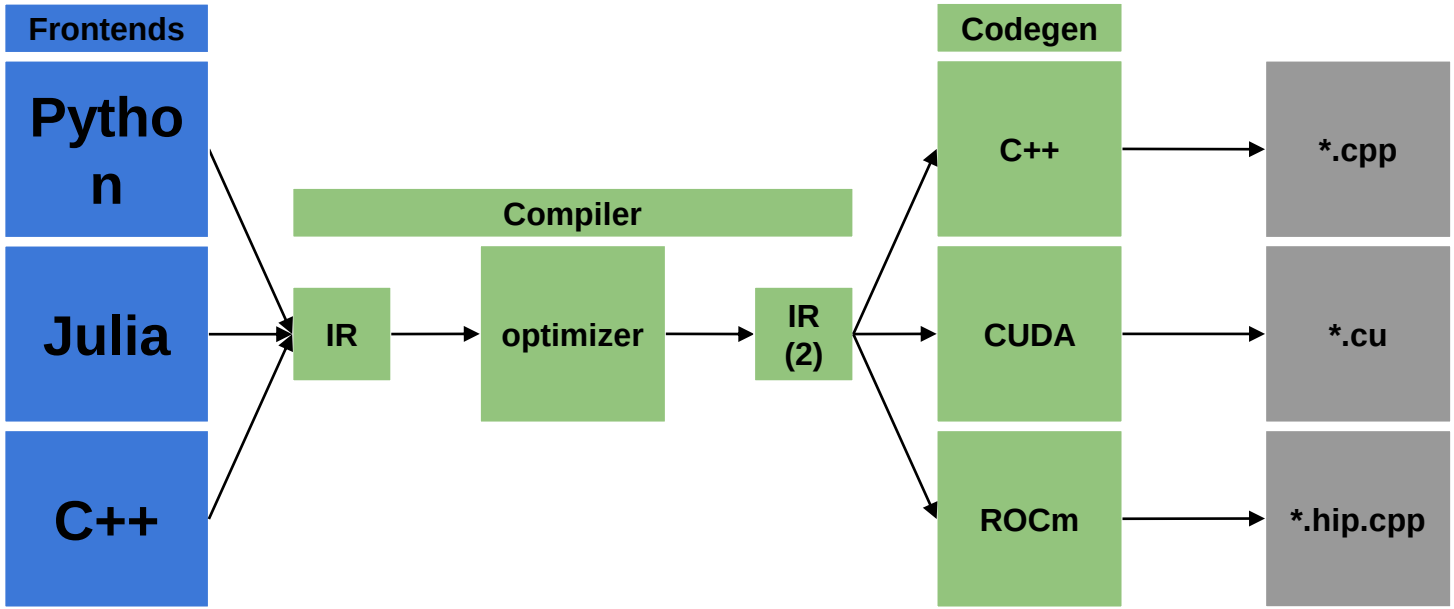
Task 1: Higher order stencil



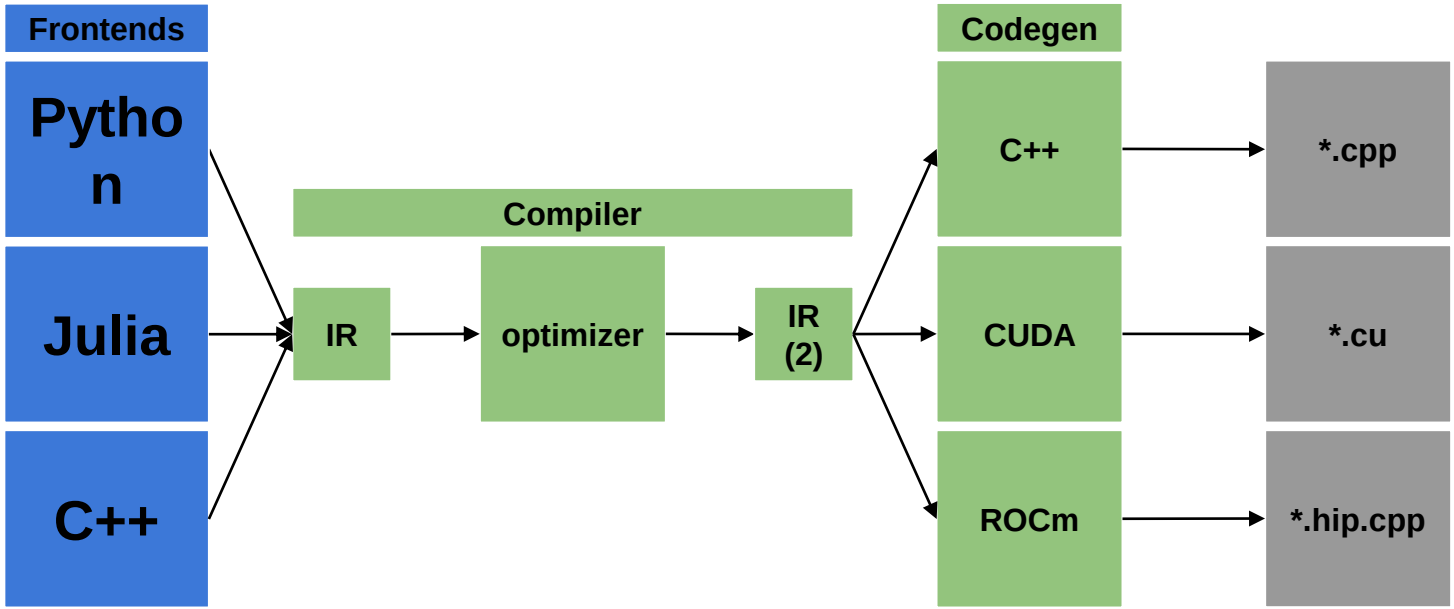
Task 2: Another architecture



Task 3: Substitute mesh library



Task 4: Fusing stencils



GT4Py Code Sample

- Definition of operators isolated
  - Can be composed from existing ones
- Programs use the operators
  - And compose them
- Infrastructure is user defined
  - Mesh, locations and connectivities

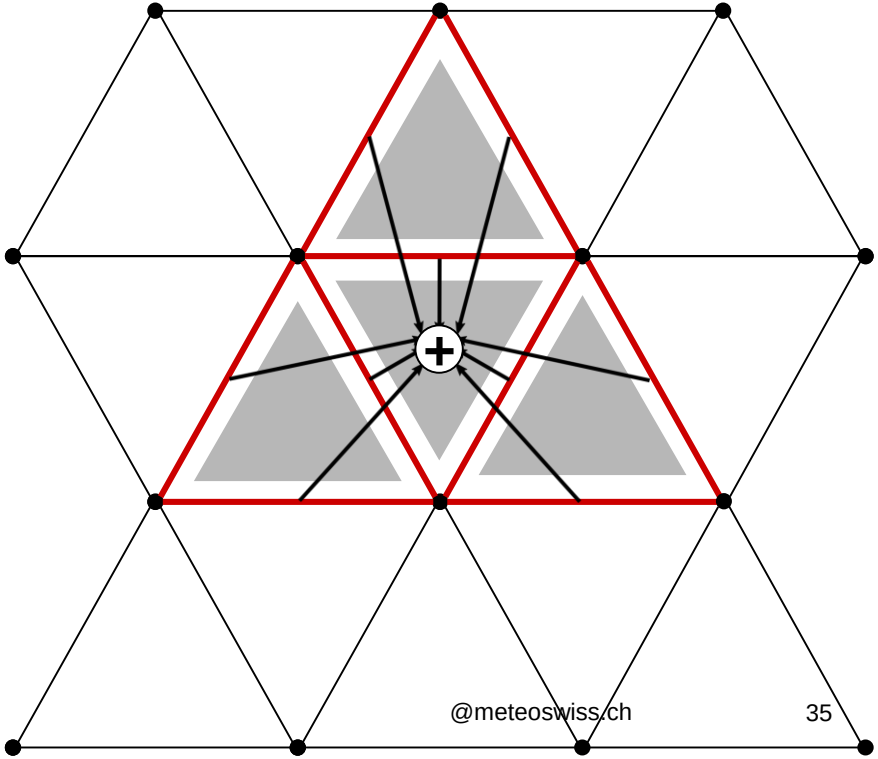
```python
from icon4py.common.dimension import (
    V2C,
    CellDim,
    KDim,
    V2CDim,
    VertexDim,
)
```

```python
@field_operator
def op_interpolation_cells2verts(
    p_cell_in: Field[[CellDim, KDim], float],
    c_intp: Field[[VertexDim, V2CDim], float],
) -> Field[[VertexDim, KDim], float]:
    p_vert_out = neighbor_sum(c_intp * p_cell_in(V2C), axis=V2CDim)
    return p_vert_out
```

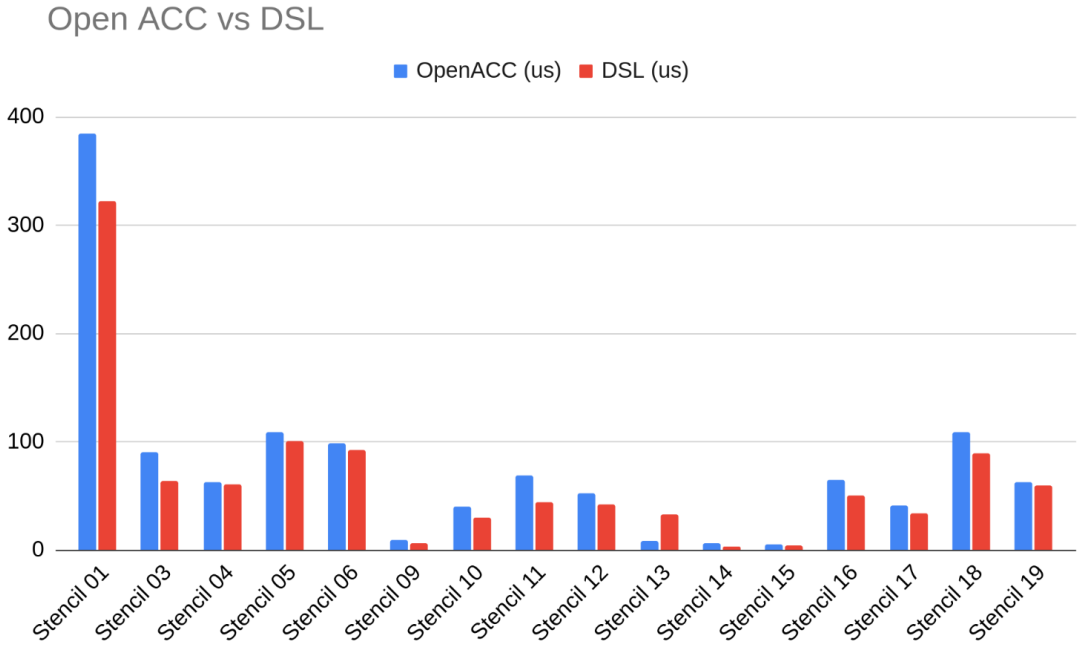
```python
@program
def interpolation_cells2verts(
    p_cell_in: Field[[CellDim, KDim], float],
    c_intp: Field[[VertexDim, V2CDim], float],
    p_vert_out: Field[[VertexDim, KDim], float],
):
    op_interpolation_cells2verts(
        p_cell_in, c_intp, out=p_vert_out
    )
```

Python DSL notation example (gt4py) : Neighbor Chains

```python
@field_operator
def intp(
    fe: Field[[EdgeDim], float],
    w: Field[[CellDim, C2E2C2EDim], float],
) -> Field[[CellDim], float]:
    f_c = neighbor_sum(w * f_e(C2E2C2E), axis=C2E2C2EDim)
    return f_c
```



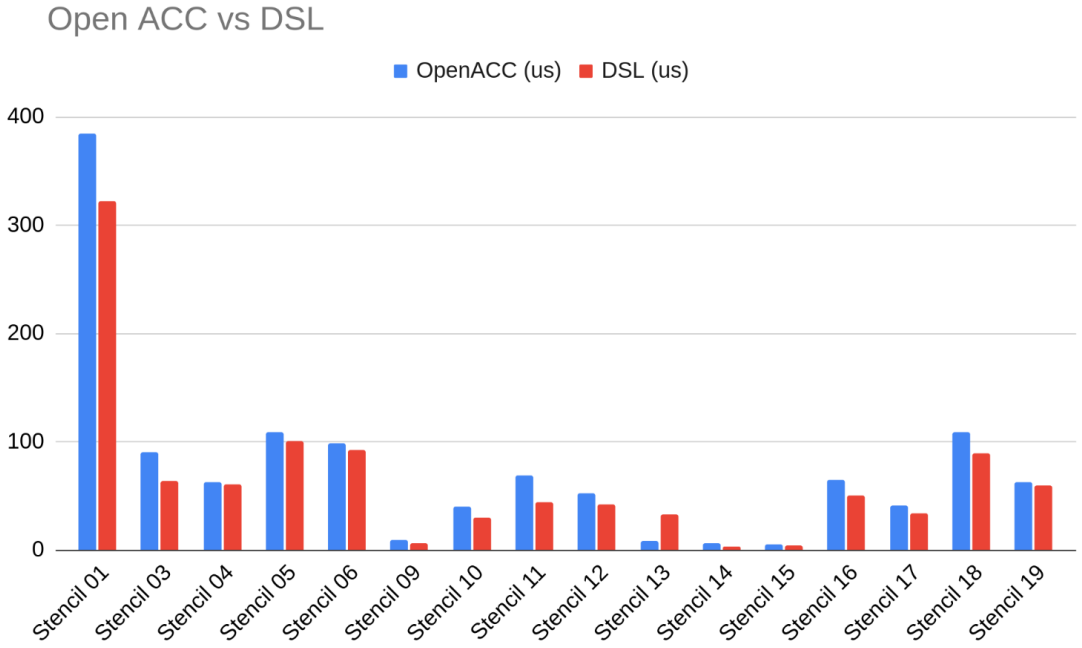
Performance of ICON dycore (DSL) prototype



DSL : Dusk/ Dawn

- Stencil by stencil translation 10-20% faster DSL compared to OpenACC.
- Prototype DSL dycore about 40% (1.4x) faster then OpenACC - not fully optimized. Dry dycore only.

Performance of ICON dycore (DSL) prototype



DSL : Dusk/ Dawn

- Stencil by stencil translation 10-20% faster DSL compared to OpenACC.
- Prototype DSL dycore about 40% (1.4x) faster then OpenACC - not fully optimized. Dry dycore only.

Conclusions

- First version of ICON model ported to GPU using OpenACC compiler directives
- Most components for global and regional NWP ported, and shall be soon available
  - basic version: Q4 2022
  - complete version: std NWP configurations tested with buildbot: Q4 2023
- Reasonable first performance, 3x faster than CPU, but still potential for optimization as still 2x time slower than COSMO
- Training of the ICON developers to work with OpenACC will be organized
- NWP application can likely benefit from DSL development in EXCLAIM, in particular for the dynamical core which shall be used once available

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

MeteoSchweiz
Operation Center 1
CH-8058 Zürich-Flughafen
T +41 58 460 91 11
www.meteoschweiz.ch

MeteoSvizzera
Via ai Monti 146
CH-6605 Locarno-Monti
T +41 58 460 92 22
www.meteosvizzera.ch

MétéoSuisse
7bis, av. de la Paix
CH-1211 Genève 2
T +41 58 460 98 88
www.meteosuisse.ch

MétéoSuisse
Chemin de l'Aérologie
CH-1530 Payerne
T +41 58 460 94 44
www.meteosuisse.ch