Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement des Innern EDI
Bundesamt für Meteorologie und Klimatologie MeteoSchweiz

C2SM Contribution to model development

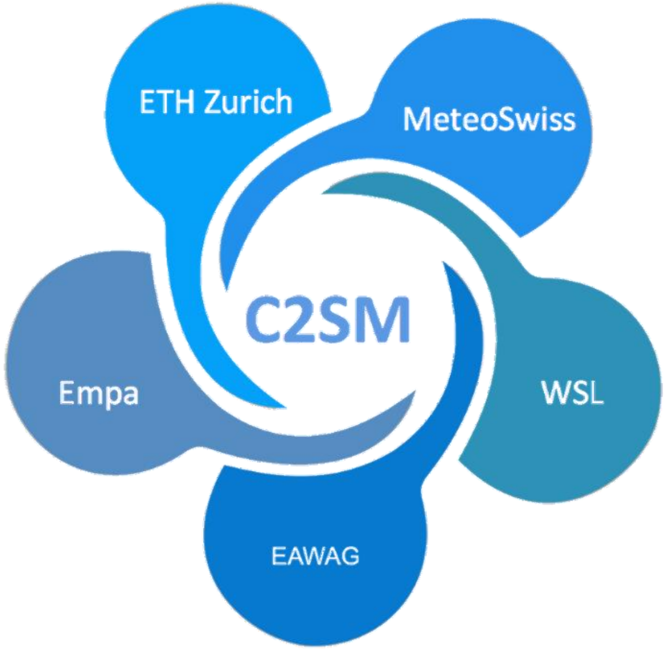
X. Lapillonne, C2SM core, EXCLAIM and ICON-22 teams

C2SM Center for Climate Systems Modeling

ETH zürich

xavier.lapillonne@meteoswiss.ch

ETH-Center for Climate Systems Modeling C2SM



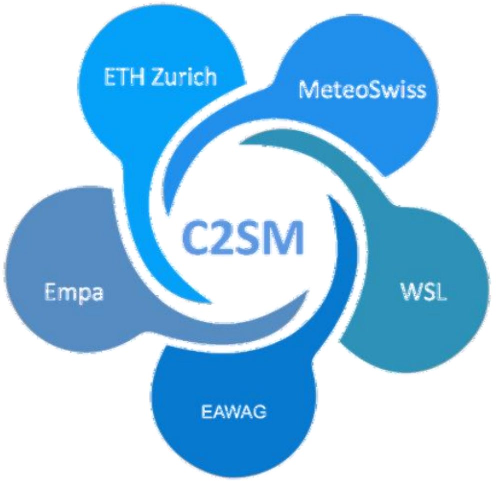
Science and technology platform of ETH Zurich

Partner institutions

Federal Office of Meteorology and Climatology MeteoSwiss
Swiss Federal Laboratories for Materials Science and Technology Empa
Swiss Federal Institute for Forest, Snow and Landscape Research WSL
Swiss Federal Institute of Aquatic Science and Technology Eawag

Enabling weather and climate science at Zurich hub

Collaborative platform for innovation



Scientific and technical support
Technical model development

Technical development of weather & climate models

High Performance Computing

Swiss Climate Change Scenarios

Climate Impacts

Organiser of technical training
Vehicle for public outreach

Center for Climate Systems Modeling C2SM

Main contributions to ICON developments from C2SM institutions:

- Lead role and coordination to port and optimize the ICON model to run on GPUs
- Explore new python-based programing paradigm to achieve performance portability
- Develop and maintain testing infrastructure and some libraries for ICON
- Develop or implement component for high resolution, or Alpine conditions, e.g. snow model, hailcast
- Investigate use of Machine Learning with ICON

A journey to GPU computing with ICON

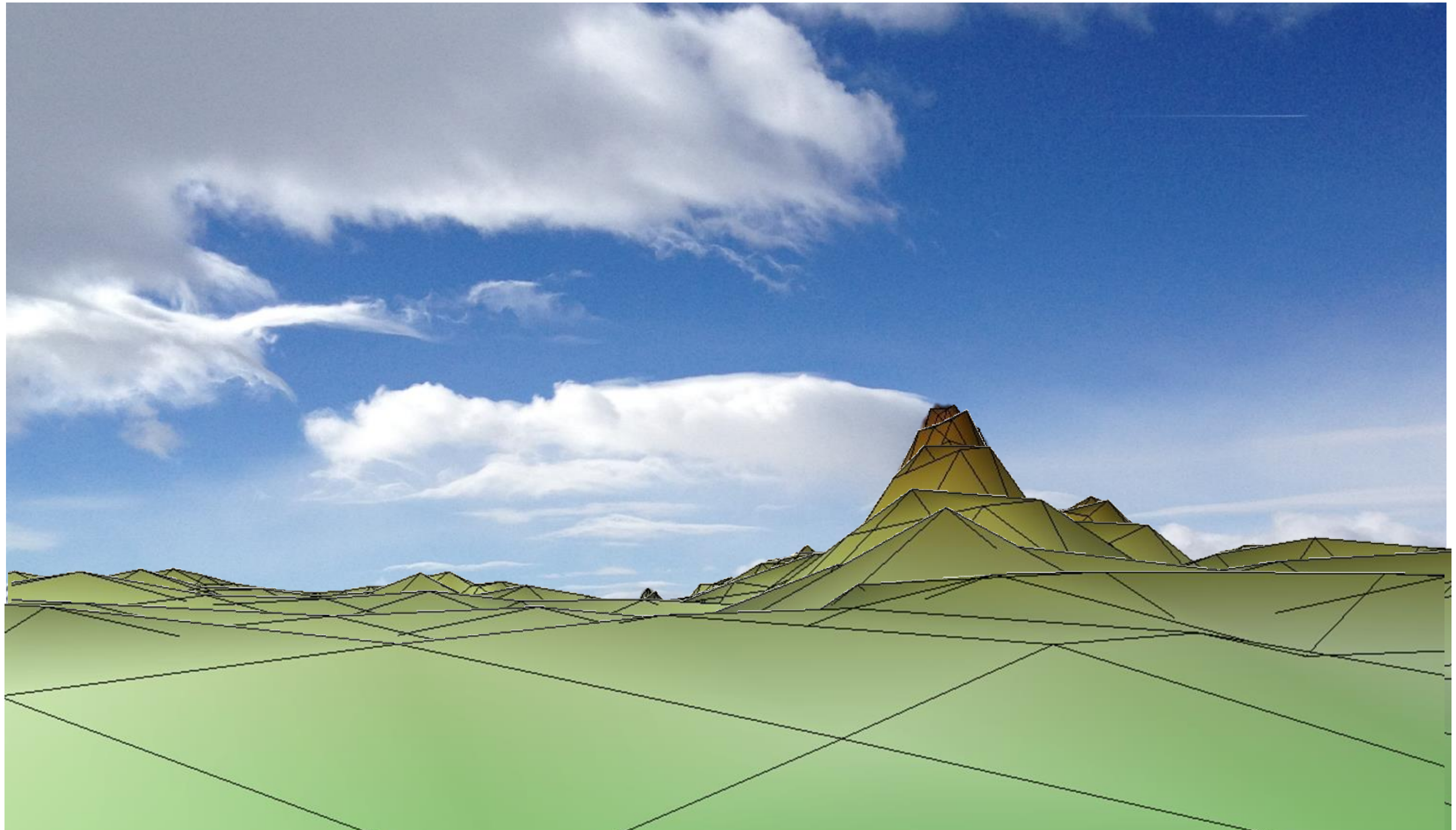
- Why using GPUs ?
- Take advantage of the high computational capacity and energy efficiency of this type of hardware
- Fastest HPC systems world wide are GPU hybrid system
- AI is driving the hardware industry and is using GPUs
- Improve model resolution, ensemble members, complexity

Alps Hybrid GPU infrastructure at CSCS

Resolution of the computational grid

$\Delta x = 2200$ m

2 x horizontal resolution ≈ 8 x computational cost

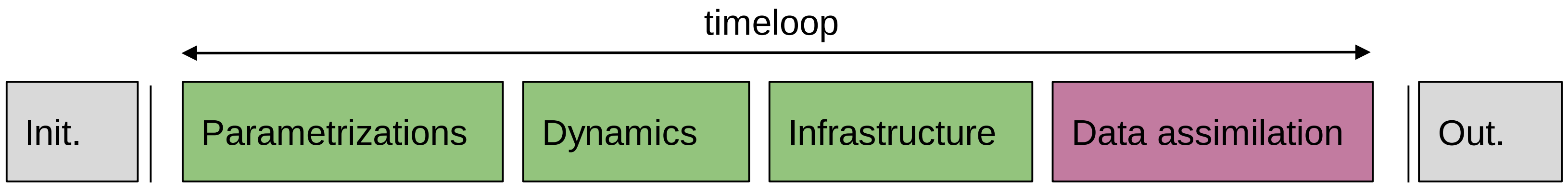


Porting ICON to GPUs

ICON is a large community model, about 1 million lines of code, porting was done in several steps, focusing on particular applications

- 2014 First start with a CSCS (ETHZ) project focusing on the dycore
- 2016 Swiss funded PASC ENIAC project involving MeteoSwiss, C2SM, CSCS and MPI-M focusing on Climate application
- 2020 MeteoSwiss ICON-22 project and COSMO IMPACT project with support from DWD focusing on NWP configurations
- 2022 EXCLAIM : Rewrite using a python-based approach, other projects from KIT, EMPA …

ICON model on GPU

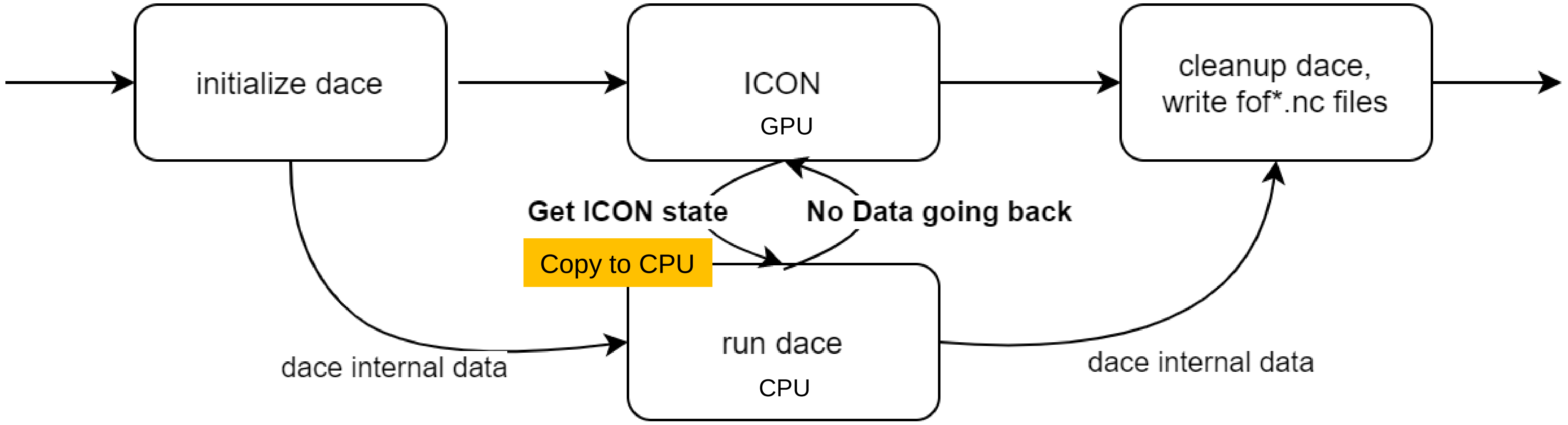


CPU - GPU - Mixed

- Full port strategy : avoid GPU-CPU transfer: all components of the time loop need to be ported to GPU
    - Exception: Data assimilation runs on partly on CPU, some diagnostics

- First GPU implementation using OpenACC compiler directives in the orignal Fortran code

Data Assimilation (DACE) on GPU strategy



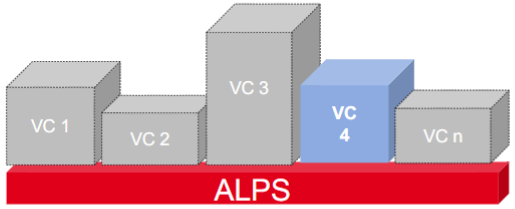
- The DACE code is kept on the CPU. Data is copied from GPU to CPU when needed.

MeteoSwiss system at CSCS

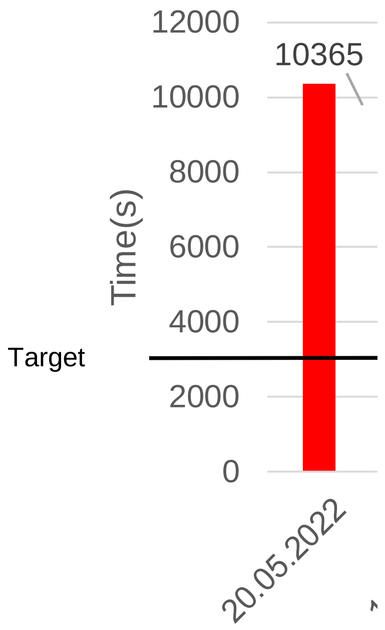
MeteoSwiss system HPC Computing Services on Alps Plattform 2 Virtual Clusters (VC)

- Production: 42 GPU / 15 CPU Nodes
- R&D: 30-50 GPU / ~15 CPU Nodes (elastic)

- GPU nodes:
  - 4 x NVIDIA A100
  - 1 x AMD Epyc 64-cores CPU
- CPU nodes:
  - 2 x AMD Epyc 64-cores CPUs
- Not dedicated : part of the large system :
  - R&D partition can be extended
  - new challenges : maintenance, testing …



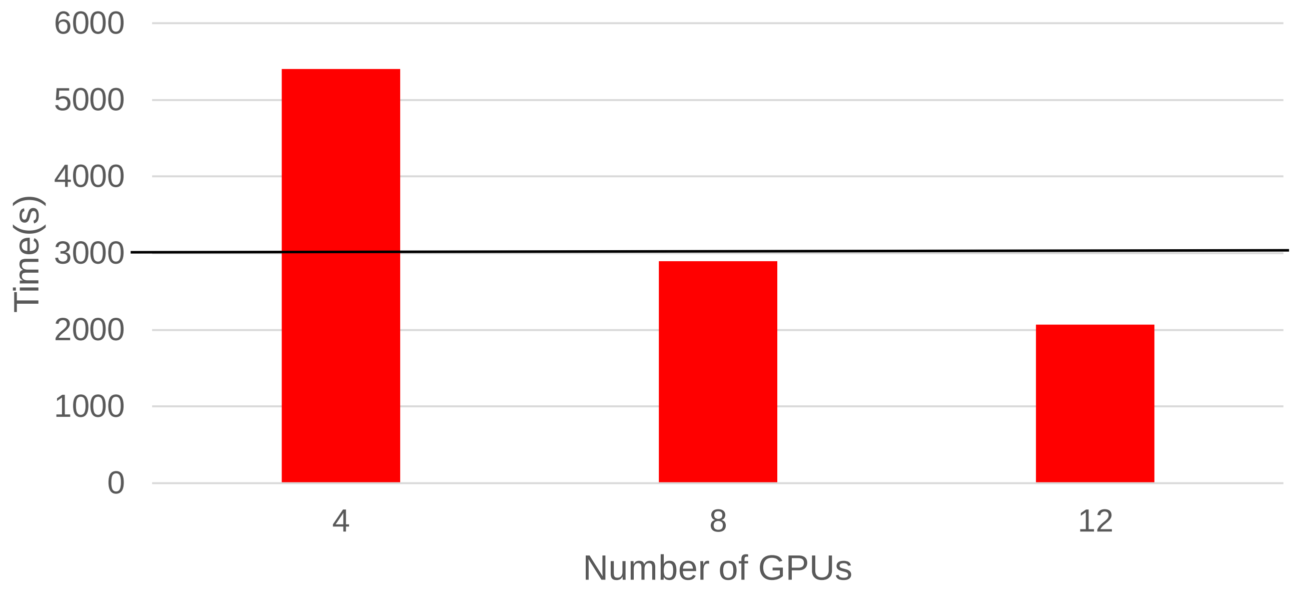
ICON-GPU MeteoSwiss : Performance optimization



🟥 ICON (A100 GPUs) ■ COSMO (V100 GPUs)

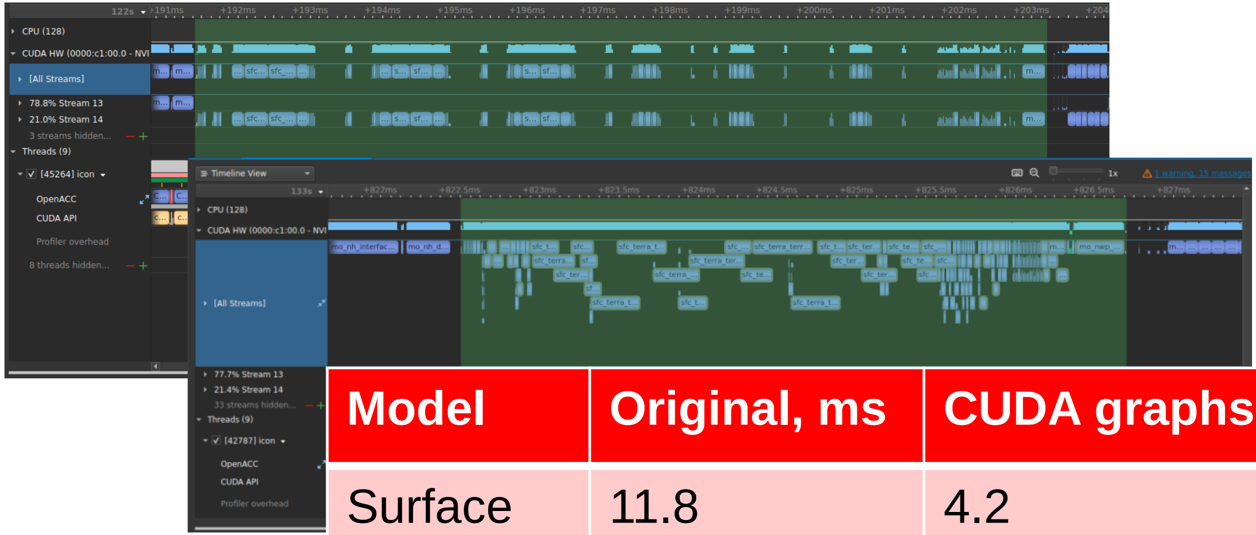
- ICON-CH1-EPS on Alps, 33h on 8 A100 Nvidia GPUs, i.e. 2 nodes.
- Optimization reduced time to solution to target performance

Scaling of ICON-CH1-EPS



- Fit within time to solution on 8 GPUs (=2 nodes), still good scaling when using 12 GPUs (= 3 nodes)

Cuda-graph Optimization



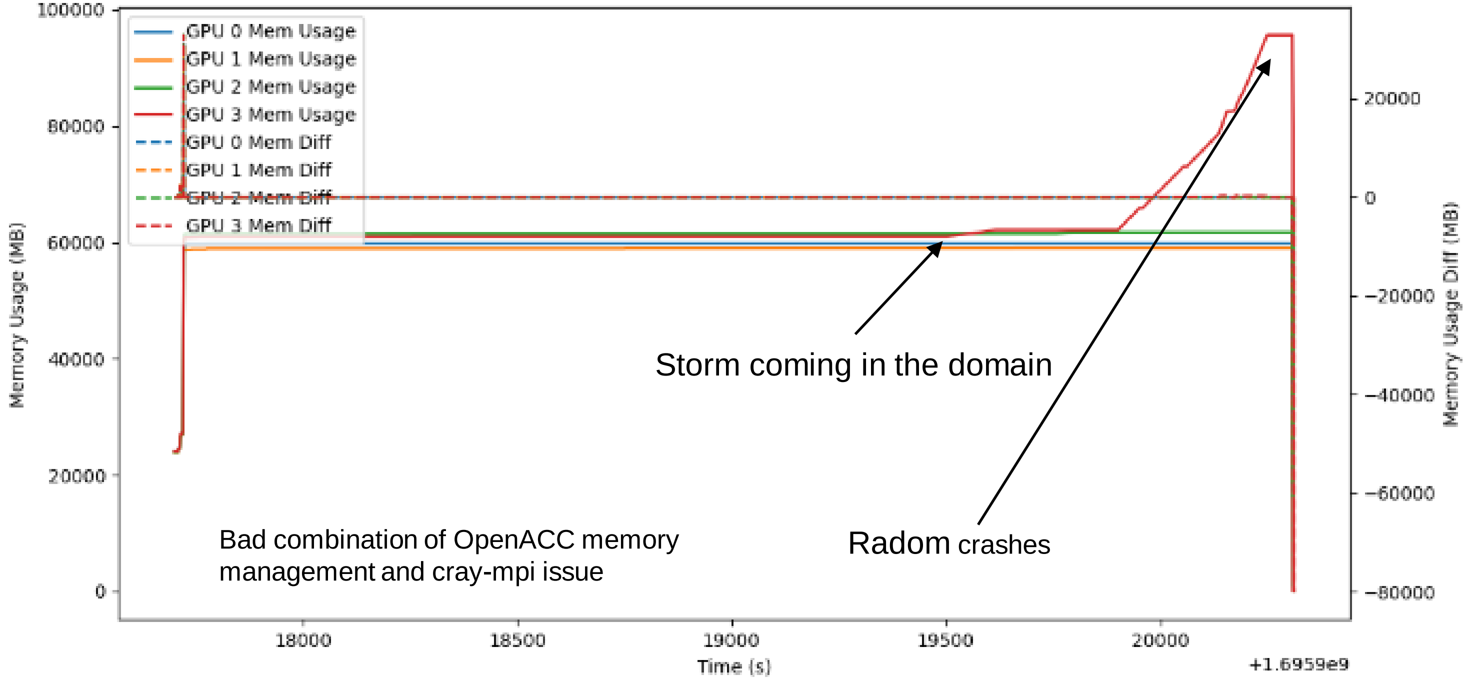
Beneficial for components with many small kernels

Model	Original, ms	CUDA graphs, ms	Speedup
Surface	11.8	4.2	2.8x
Turbtrans	10.4	2.0	5.1x

Source: Dmitry Alexeev, NVIDIA

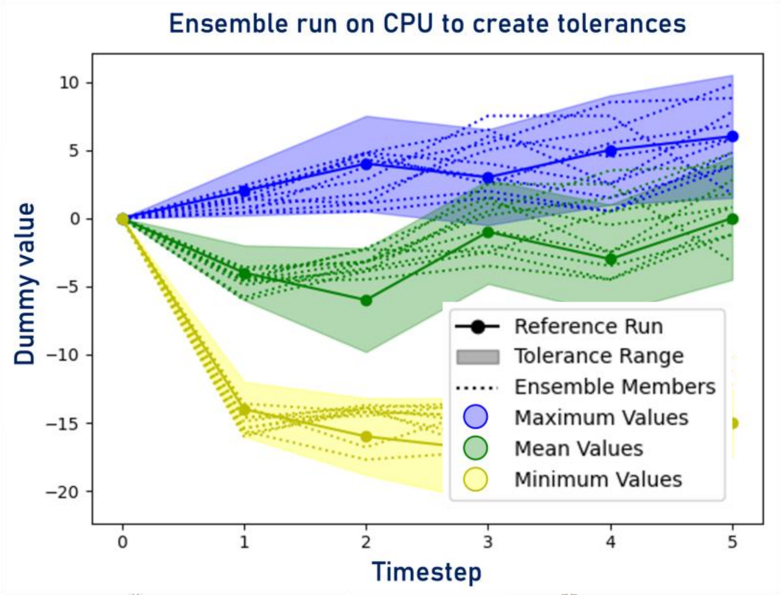
GPU development in ICON

- Many issues on our way OpenACC, mpi, competing changes in other MR …
- 9 days before pre-operation: random crash for some weather situation



Testing and validation

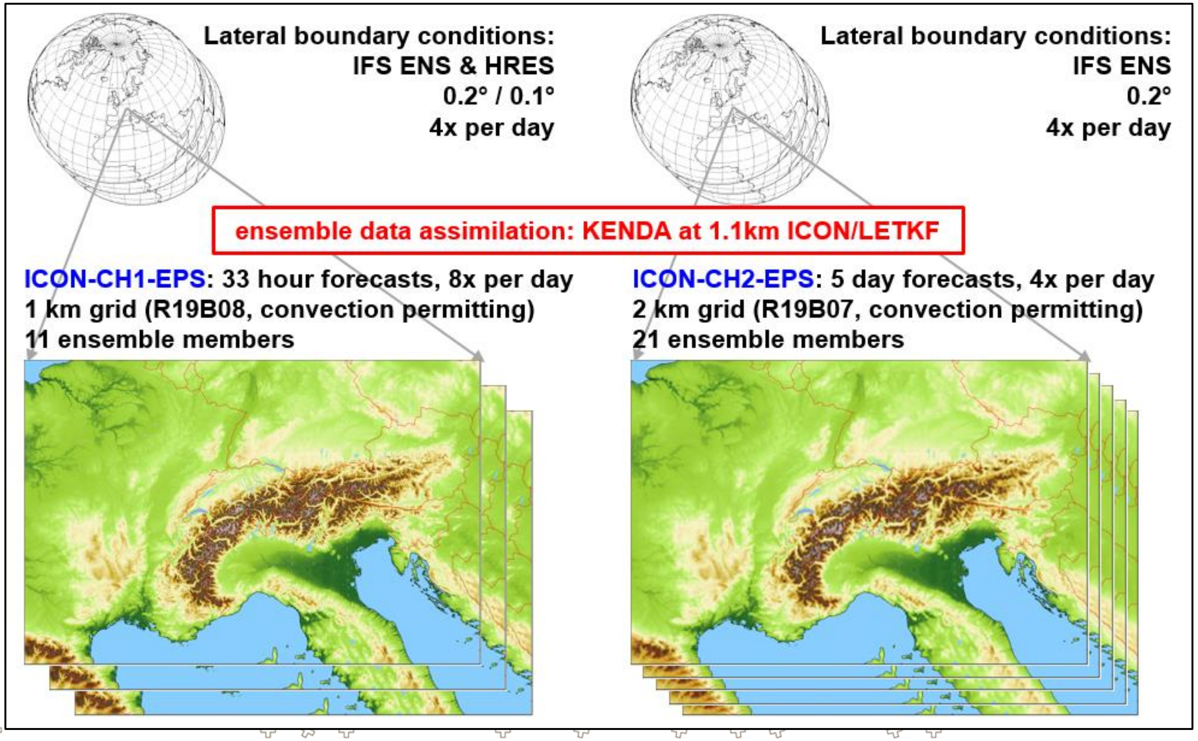
- Short regression tests validating that GPU is within a perturbed CPU ensemble : probtest automatised testing framework
- Running on many systems of the ICON community, integrated with gitlab
- Long validation (multiple seasons) against observations and CPU executable



MeteoSwiss ensemble system

- Pre-operational phase started October 2nd 2023
- Not using all optimization yet (no cuda-graph)

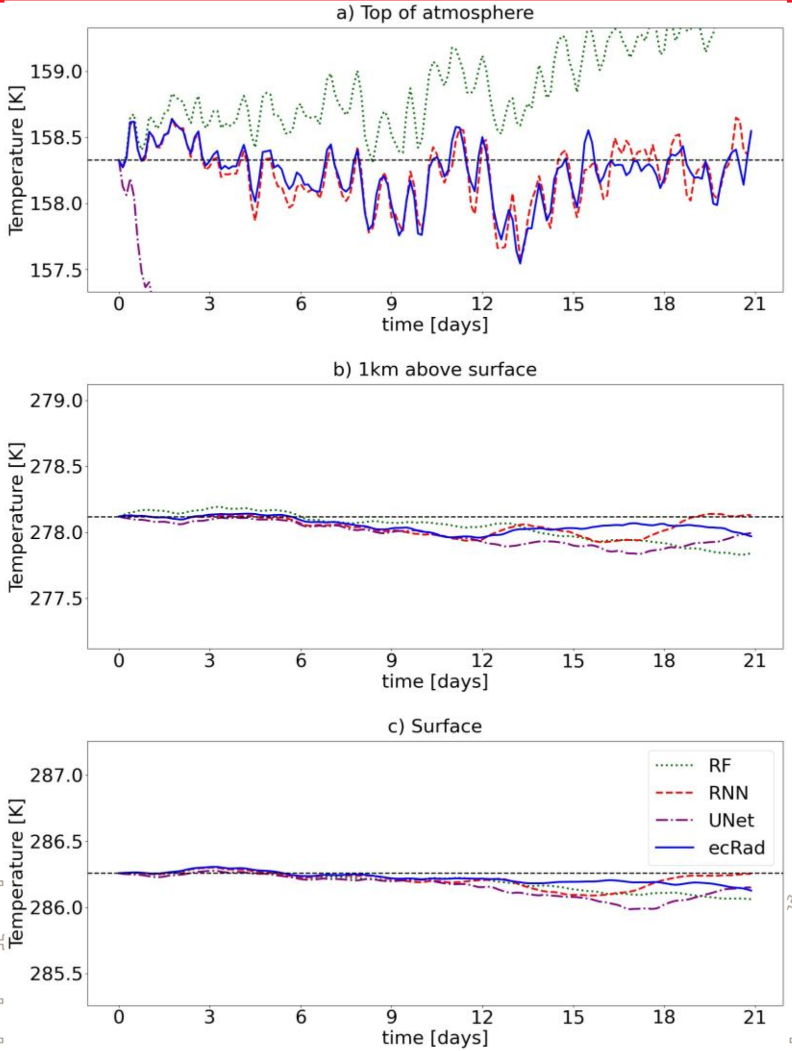
- Full schedule on Tasna (Alps)
- ICON-CH1-EPS : 3 nodes/member
- ICON-CH2-EPS : 1 node/member
- KENDA-CH1 : 1 node/member


Lateral boundary conditions:
IFS ENS & HRES
0.2° / 0.1°
4x per day
Lateral boundary conditions:
IFS ENS
0.2°
4x per day
ensemble data assimilation: KENDA at 1.1km ICON/LETKF
ICON-CH1-EPS: 33 hour forecasts, 8x per day
1 km grid (R19B08, convection permitting)
11 ensemble members
ICON-CH2-EPS: 5 day forecasts, 4x per day
2 km grid (R19B07, convection permitting)
21 ensemble members

ML based radiation

Courtesy Guillaume Bertolli

- ETH project in Sebastian Schemm's group to replace the ecRad radiation scheme in ICON with a ML based model
- Having ICON running on GPU allows potentially to run both ICON and the radiation ML inference on the GPU



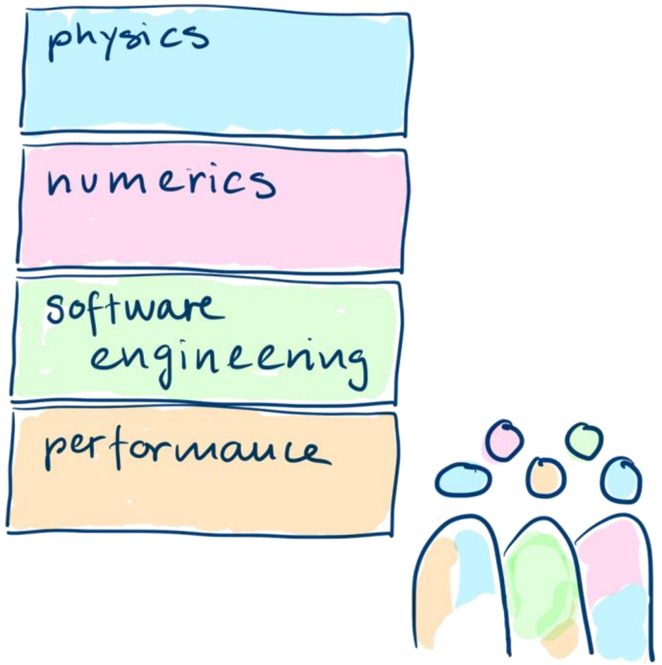
MeteoSchweiz

EXCLAIM: Exploring new python-based programing paradigm to achieve performance portability

Courtesy Magdalena Luz and the EXCLAIM team

EXCLAIM

Complexity



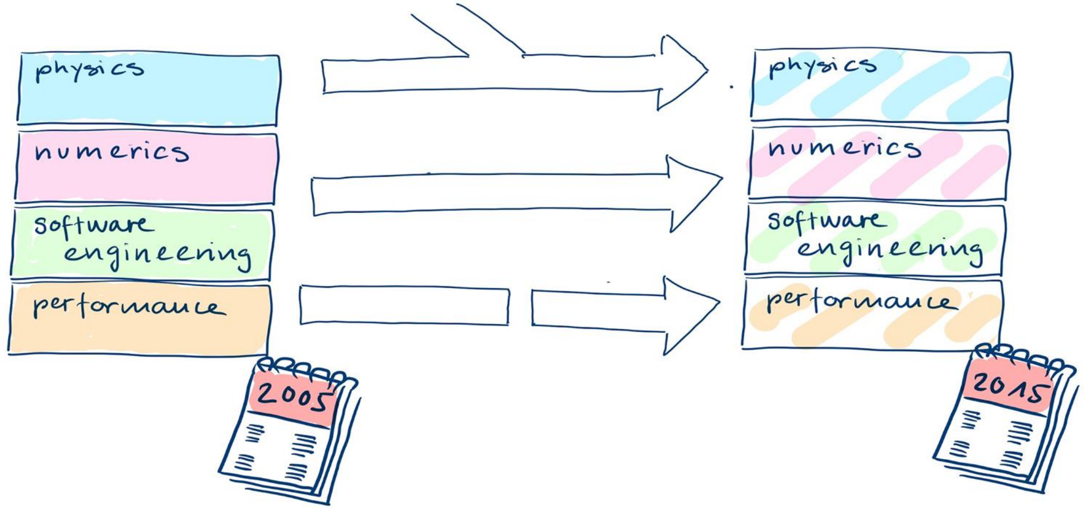
Model development takes a lot of

knowledge…

No one can know this all, at least not well enough…

because we need to be on the top of each of them.

Time



Common to all large software systems:

If you rest, they rot.

Symptoms in ICON

- all aspects are mixed up and “on the same page”:
  - physics, numerics implementation
  - hardware specific instructions, (openacc, compiler ifdef)
  - performance tweeks
- it uses a old (fashioned) language : Fortran
- low abstraction level - the code is not expressive, lot of duplication.

Why: Community focus is on

- **physics**, **numerics** - of course, main developers are domain scientists!
- **stability** - NWP production forecasts!
- performance - to lower extend due to knowhow

Other aspects

- **hard to read** and understand even if you have some basic knowhow on the field
- hard to **interface** with new developments, for ex. AI model
- difficult to recruit and onboard new people
- **Current OpenACC is working,** but mostly for Nvidia hardware, AMD only with Cray system. Will OpenACC + Fortran still available in 5 years from now ?
- **performance**, can we improve by using more sophisticated methods?

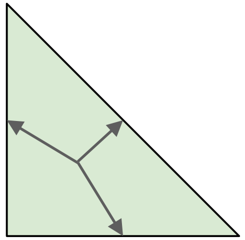
A solution?

Approach taken by EXCLAIM

EXCLAIM

Domain Specific Language

- makes it easier to express your domain
  - simplifies code by adapting the abstraction layer
  - separate what you do from where you run and how you do it



Example: Divergence

```
div = neighbor_sum(vn(C2E) * geofac_div, axis=C2EDim)
```

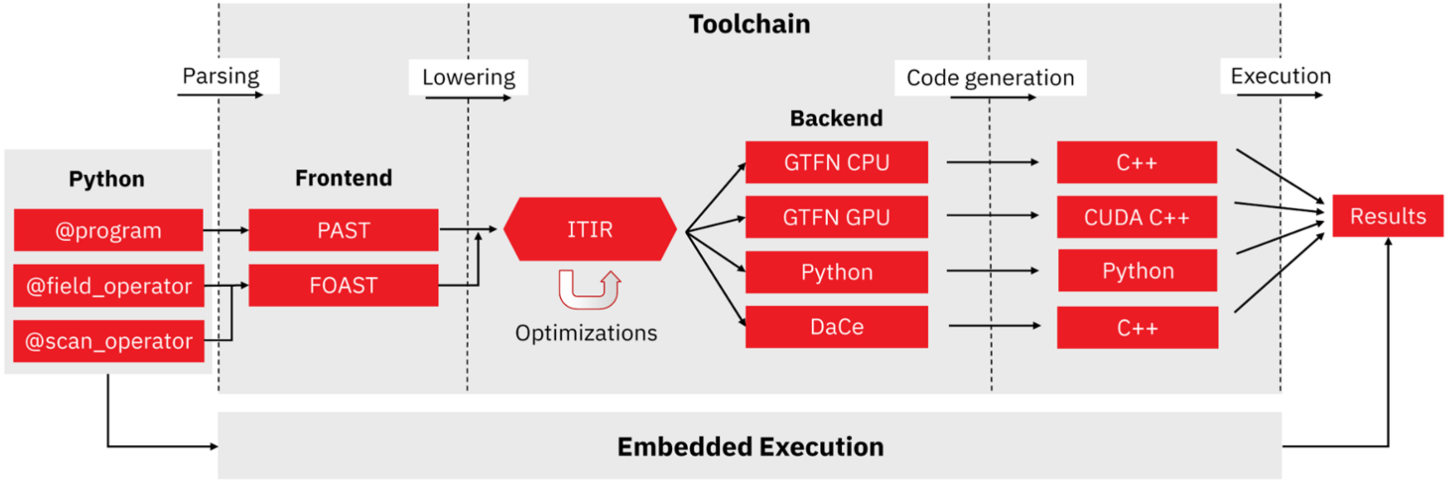
```
DO jb = i_startblk,i_endblk

  CALL get_indices_c(p_patch, jb, i_startblk, i_endblk, &
    i_startidx, i_endidx, rl_start, rl_end)

  DO jk = 1, nlev
    DO jc = i_startidx, i_endidx

      div(jc,jk) = p_nh_prog%vn(ieidx(jc,jb,1),jk,ieblk(jc,jb,1))*p_int%geofac_div(jc,1,jb) + &
                   p_nh_prog%vn(ieidx(jc,jb,2),jk,ieblk(jc,jb,2))*p_int%geofac_div(jc,2,jb) + &
                   p_nh_prog%vn(ieidx(jc,jb,3),jk,ieblk(jc,jb,3))*p_int%geofac_div(jc,3,jb)
    ENDDO
  ENDDO
ENDDO
```

```
@field_operator
def _divergence(vn: Field[[EdgeDim, KDim], float],
                geofac_div: Field[[CellDim, C2EDim], float]) -> Field[[CellDim, KDim], float]:
    return neighbor_sum(vn(C2E) * geofac_div, axis=C2EDim)


@program(backend=...)
def divergence(vn: Field[[EdgeDim, KDim], float],
               geofac_div: Field[[CellDim, C2EDim], float],
               div_vn: Field[[CellDim, KDim], float):
    _divergence(
        vn, geofac_div, out=div_n
    )


_divergence(vn, geofac_div, div_n, offset_provider = {"C2E": c2e_offset_provider})
```

XCLAIM

GT4Py

GridTools4Python: embedded in Python, uses GridTools in its toolchain.



Ongoing gt4py investigation : ICON (C2SM), FVM (ECMWF), FV3 (NASA)

XCLAIM

2 development streams in EXCLAIM

1. GT4Py in ICON : python stencils in Fortran code

2. Python Port of ICON : full re-write in Python

1. Fortran driven ICON with GT4Py stencils

GT4Py in ICON

Performance:

Target a 1.3x speed improvement, matching MeteoSwiss's Dusk/Dawn benchmarks (See Christoph's Poster).

Maintainability:

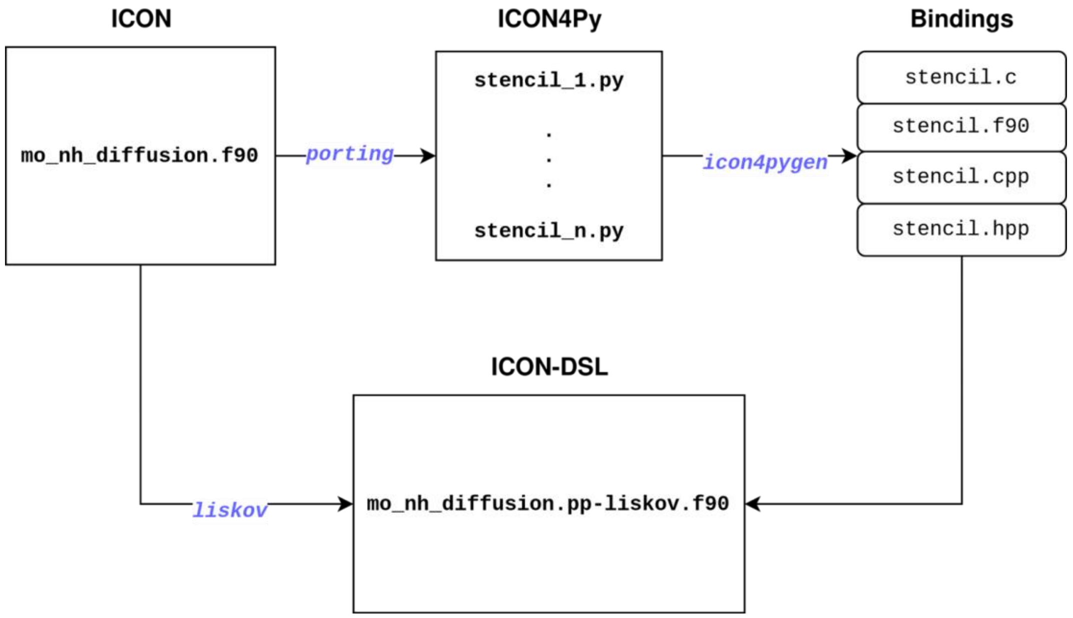
Make stencil updates and critical code optimization easier to do since we’re using Python.

Portability:

Achieve "write once, run anywhere" with adaptable backend code for various hardware architectures.

Integrating Python in a Fortran environment

1. **Porting:** Convert Fortran stencils to GT4Py stencils written in Python.
2. **Code generation:** Utilize GT4Py to produce efficient C++ code.
3. **Bindings generation:** Create Fortran bindings to enable calling the generated C++ code.
4. **Preprocess:** Modify the ICON codebase to automatically invoke the C++ code.

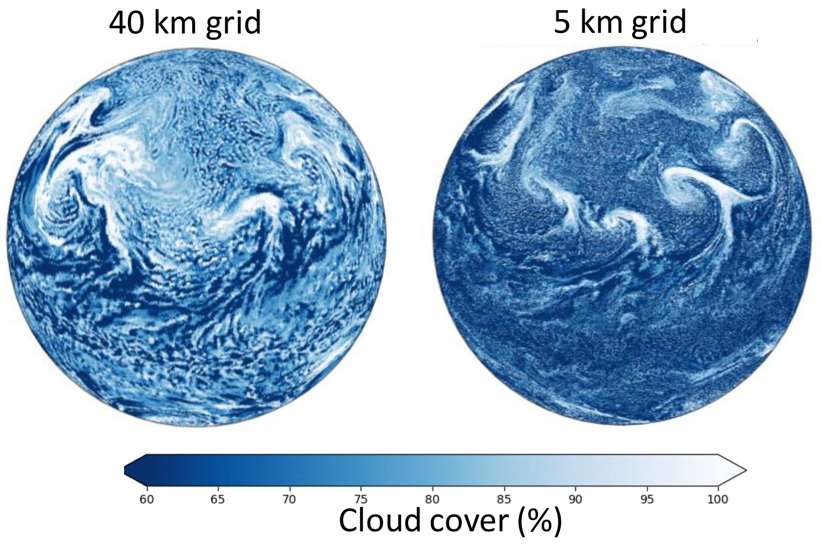


Where we are

- Dycore stencils in python integrated and validated in the Fortran code
- On par with OpenACC, working on optimization

Low level clouds

Simulations using Python dycore



Courtesy: Praveen Kumar Pothapakula

EXCLAIM

2. Python Port of ICON

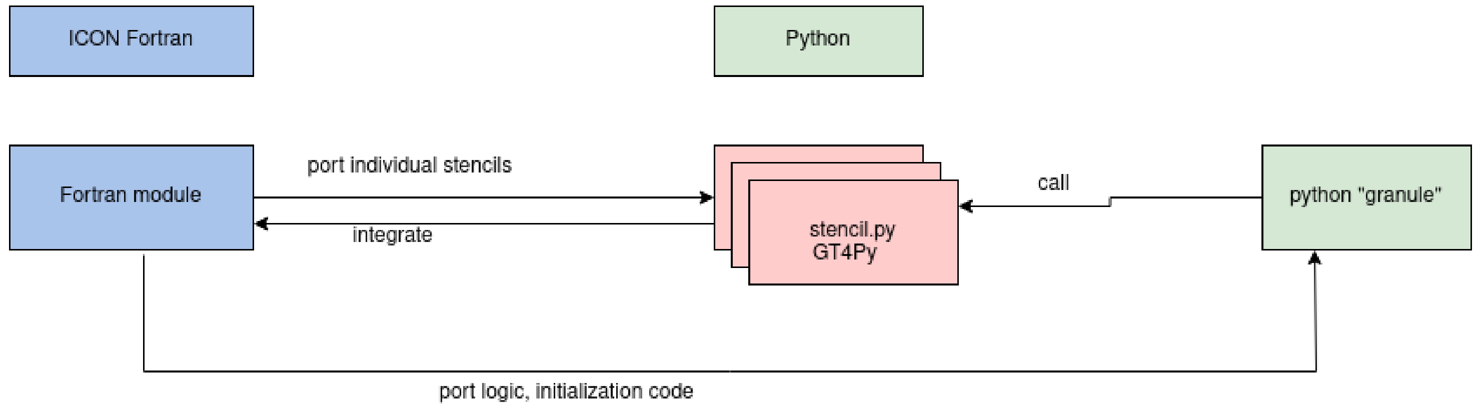
GT4Py is embedded in Python and needs Python to do its magic.

So why not use Python to drive the entire model?

We aim at:

- reach a more **modular model** architecture
- use a rich ecosystem that is actively developed
- easier onboarding of new people
- make it easier to adapt and change
- interface with new paradigms

Approach



- rewrite driving logic
- rewrite initialization code
- reuse stencils

Current status: dycore only test cases running

XCLAIM

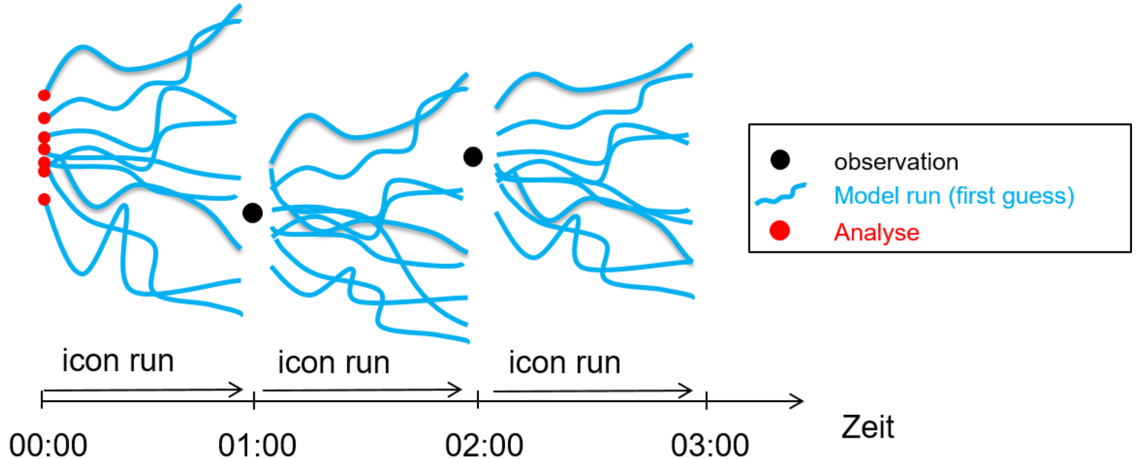
Conclusions

- Among various contributions, C2SM has played a leading role to port the ICON model to GPUs
- ICON is now one of the few model world wide to be able to run GPUs for applications ranging from Climate to operational NWP production
- Benefit from hardware development in particular taking advantage of the strong drive from Machine Learning
- Going beyond the current implementation, exploration are ongoing to use a python based framework (gt4py) to make the model future proof and improve performance portability

Backup slides

ICON with Data Assimilation on GPU

- Kilometer-Scale Ensemble Data Assimilation (KENDA). Calculations in ensemble space spanned by the ensemble members, using observations to compute analysis
- Assimilation component takes the model and compares them with observations (DACE) – write feedback files (fof)


observation
Model run (first guess)
Analyse
icon run
icon run
icon run
Zeit
00:00
01:00
02:00
03:00