PASC23
26.06.2023 28.06.2023
Davos, Switzerland

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

ICON-GPU for Numerical Weather Prediction

Marek Jacob[1], D. Alexeev[2], R. Dietlicher[3], F. Gessler[3], D. Hupp[3], X. Lapillonne[3], F. Prill[1], D. Reinert[1], U. Schättler[1], G. Zängl[1], ICON-GPU Community[1-7]
[1]German Weather Service, [2]NVIDIA Inc., [3]Meteoswiss, [4]Swiss National Supercomputing Centre, [5]Center for Climate Systems Modeling (C2SM), [6]Max Planck Institute for Meteorology, [7]DKRZ

An image of "The Next Generation Supercomputer for the German Weather Service" generated with the deep learning model DALL·E 2 by OpenAI and the additional prompts "Add a logo with a white swirl on blue swirl" and "Print the letters DWD".

The ever doubling of scalar CPU performance has slowed down – known as the "End of Moore's law". Graphical Processing Units (GPU) provide massively parallel computing power. Do they become the future hardware for Numerical Weather Prediction?
The ICON model has been adapted to work with GPU systems through a multi-institute effort over the past years. Here, we present the current state.

How to Program a GPU for ICON?

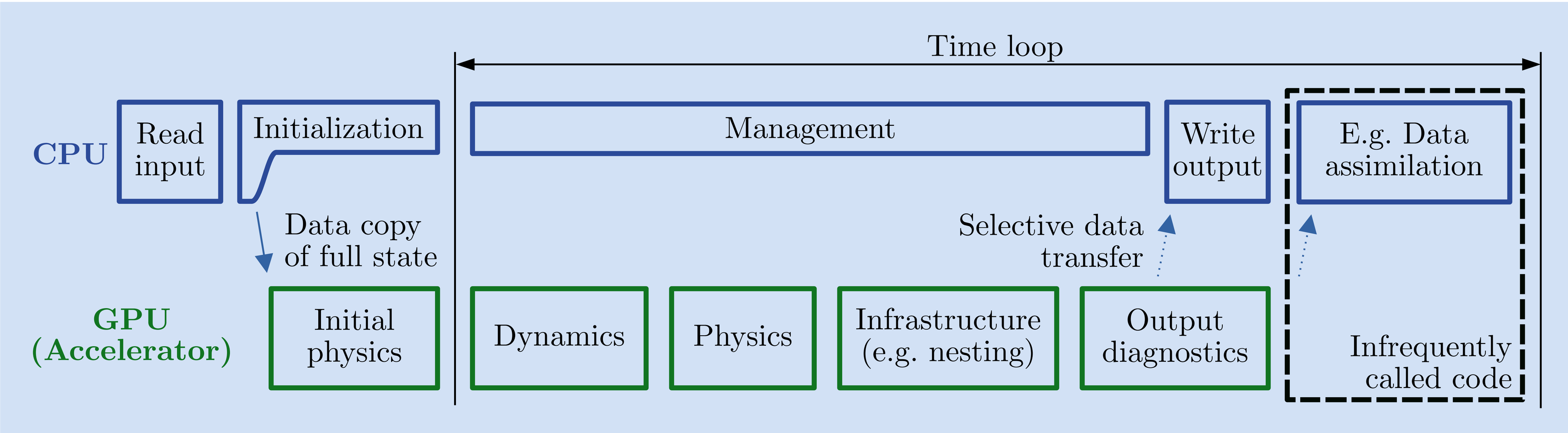
- ICON is written in FORTRAN
  - ➔ Use a directive based approach: OpenACC 2.6
- GPU Machine: CPU + GPU
  - ➔ CPU manages GPU
  - ➔ Only code that is explicitly decorated with `!$ACC` runs on GPU.
- GPU has dedicated memory
  - ➔ Double memory management
  - ➔ Cloned Variables
- CPU ⇄ GPU bandwidth is low.
  - ➔ Time loop completely on GPU.

The directive based OpenACC approach allows the ICON code to be ported incrementally. This keeps ICON-GPU inherently up-to-date with the CPU code.

Supported Platforms

NVIDIA (NVHPC SDK ≥ 21.2)
Piz Daint, Balfrin, Levante, JULES, Linux Workstations
AMD Cray/HPE compiler for LUMI: work in progress
AMD compiler: to be explored

Program Flow and Data Transfers for ICON-GPU



- *Infrequently called code runs on CPU. (E.g. initialization (configuration, reading), data assimilation couplers).*
- *Frequent and computational heavy code runs on GPU. Data remains on GPU. (CPU becomes outdated.)*
- *Some routines (e.g. MPI communication) must support CPU and GPU data.*

Memory and Parallelization Layout

Name list

```
DO jb = 1, n_blocks            ! horizontal
    DO jk = 1, n_nlevels       ! vertical
        DO jc = 1, nproma      ! horizontal
            var(jc, jk, jb) = ...
        END DO
    END DO
END DO
```

```
DO jb = 1, n_blocks
    CALL physics_1(var(:, :, jb))
END DO

SUBROUTINE physics_1(var(:, :)
    DO jk = 1, n_nlevels   ! vertical
      !$ACC PARALLEL LOOP
      DO jc = 1, nproma    ! horizontal
          var(jc, jk) = ...
      END DO
      !$ACC END PARALLEL LOOP
    END DO
```

CPU: 32
GPU: big!

Horizontal blocking and OpenACC parallelization of the innermost nproma-sized cell loop. The arrays of the parallelized loop (should) have unit stride.

Ported Features

- Dynamics
- Advection (multiple schemes)
- Nesting
- Radiation (RTE+RRTMGP, ecRad, radiative heating, reduced grid)
- Microphysics (1 Moment: Graupel, Cloudice schemes, 2 Moment Seifert-Beheng)
- Surface (TERRA, flake, JSBACH, sea ice)
- Turbulence (Raschendorfer, Vdiff)
- Convection (Tiedtke-Bechtold)
- Non-orographic gravity wave drag, sub-grid scale orographic drag
- Latent-heat-nudging
- Ensemble perturbation, SPPT
- Multiple output diagnostics
- Restart
- Output for DACE

Work in progress:
- Snow model (Snowpolino)
- Pollen emission, transport, sedimentation (ICON-ART)
- Compatibility to RTTOV/synsat and EMVORADO

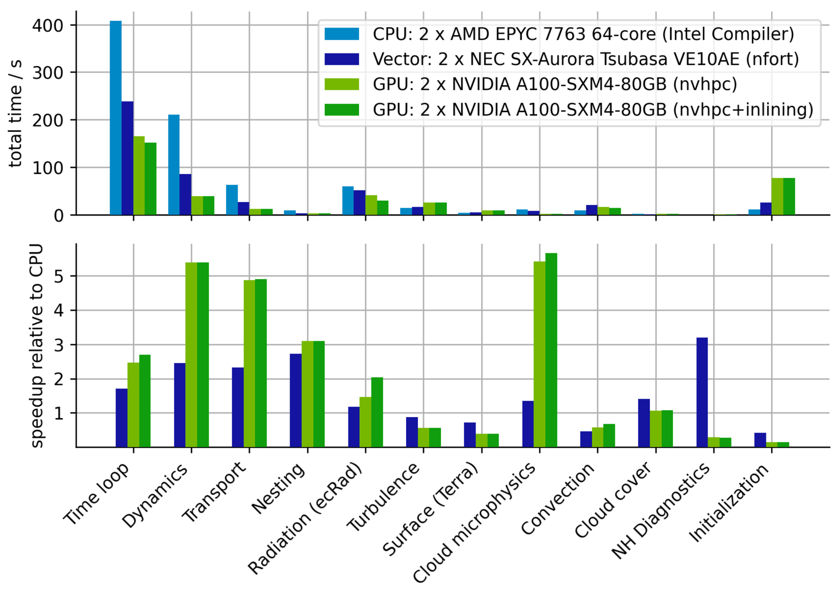
Lessons learned

- CPU - GPU equivalence breaks easily.
- Every ported feature must be covered by a test. The tests compare GPU results against CPU reference plus a small tolerance.
- Care should be taken to add `STOP` calls to unported code branches.
- ICON's block structure (see box to the left) is well suited for straight forward GPU parallelization.
- Novel `!$ACC` pragmas have to be explained to scientific model developers for better acceptance and sustainability.
- Uniform look and feel of `!$ACC` improves readability.
- Close collaborations with compiler and vendor teams are very helpful.

Optimizations

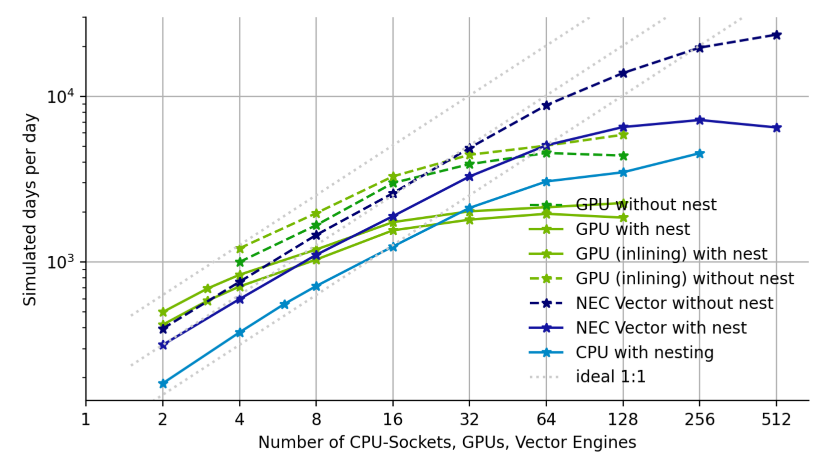
- Almost no CPU-GPU data transfers in the time loop.
- Direct GPU-to-GPU communication (1.3…1.6 x).
- Asynchronous GPU execution (1.16 x).
- Compiler inlining for modularized code (~1.1 x).
- Fuse kernels using `GANG(STATIC:1)`.
- Upcoming: Reduction of kernel launch overhead using CUDA Graphs (~1.2 x, requires to-be-released Nvidia SDK).

Performance Comparison Chip-to-Chip (current state)



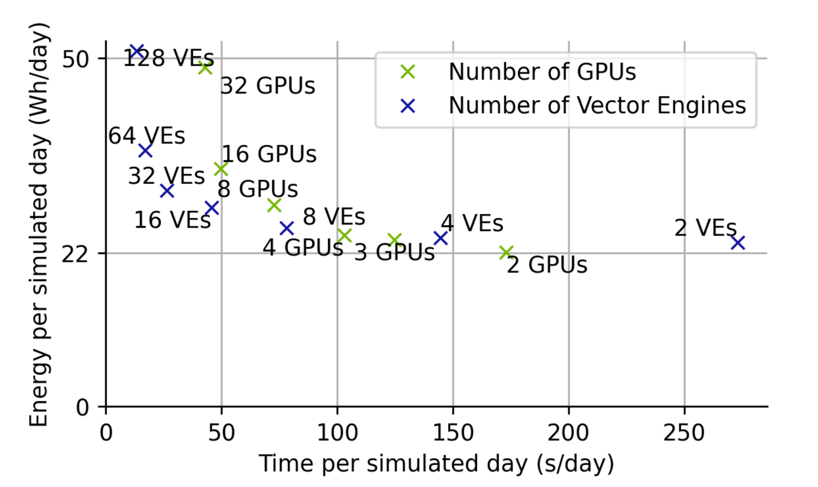
Comparison of different timers reported by ICON. The experimental setup is similar to DWD global + ICON-EU ensemble forecast (in 2021). Resolution: Global R2B6 (~ 328 000 cells, 40 km) with nested grid over Europe (~ 49 000 cells, 20 km). Output: disabled. Timestep: 360 s, radiation on reduced grid every 36 min.

Strong Scaling



Speed up by hardware doubling. Strong scaling saturates when number of cells (in nest) is on the order of hardware vector length. NEC vector length: 256. Number of A100 cores (double precision): 3456.

Energy efficiency



Throughput and Energy used for time loop. R2B6 Experiment. Values as reported by nvidia-smi (NVIDIA) and veda-smi (NEC), excluding the host CPU.

Conclusions

- ICON-GPU is almost ready for operational service at MeteoSwiss and is almost feature complete for DWD's operational setups.
- NWP-relevant configurations (rather few VEs or GPUs) become faster and more energy efficient on GPUs than on DWD's current vector machine. Further optimizations are work-in-progress.
- Non-unified GPU and CPU memory makes debugging very laborious.

Marek.Jacob@dwd.de
Business Unit for Research and Development, Numerical Models (FE 13)