**ICON-GPU for Numerical Weather Prediction**

**Marek Jacob**¹, D. Alexeev², R. Dietlicher³, F. Gessler³, D. Hupp⁴, X. Lapillonne⁵, F. Prill⁴, D. Reintert⁴, U. Schättler⁴, G. Zängl⁴, ICON-GPU Community¹⁻⁷

¹German Weather Service, ²NVIDIA Inc., ³Meteoswiss, ⁴Swiss National Supercomputing Centre, ⁵Center for Climate Systems Modeling (C2SM), ⁶Max Planck Institute for Meteorology, ⁷DKRZ

---

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 `#pragma acc` runs on GPU.
- **GPU has dedicated memory**
  - Cloned Variables
    - Double memory management
      - Only code that is explicitly decorated with `#pragma acc` runs on GPU.
- **GPU has management over CPU**
  - CPU manages GPU
    - Use a directive based approach:
      - `!$ACC`

**Ported Features**

- **Dynamics**
- **Advection (multiple schemes)**
- **Nesting**
- **Radiation**
  - RTE+RRTMGP, ecRad
- **Microphysics**
  - 1 Moment: Graupel, Cloudice schemes, 2 Moment Seifert-Beheng
- **Surface**
  - TERRA, fLack, Jüles, Linux Workstations
- **Ensemble perturbation**
  - SPPT
- **Snow model**
  - Snowpolino
- **Pollen emission, transport**
  - ICON’s block structure (see box to the left)
- **Latent-heat-nudging**
- **Grid scale orographic drag**
  - ICEM (life cycle analysis)
- **Sedimentation**
  - ICON-ART
- **ECMWF diagnostics**
- **Energy efficiency**

**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 straightforward GPU parallelization.
- Novel `#pragma` pragmas have to be explained to scientific model developers for better acceptance and sustainability.
- Uniform look and feel of `#pragma` 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 (~1.6 x).
- Asynchronous GPU execution (~1.6 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.

**Program Flow and Data Transfers for ICON-GPU**

**Performance Comparison Chip-to-Chip (current state)**

**Energy efficiency**

Throughput and Energy used for time loop. R2B6 Experiment. Values as reported by nvprof (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.

---

**References**

1. G. Zängl
2. NVIDIA Inc.
3. Meteoswiss
4. Marek.Jacob@dwd.de
5. Business Unit for Research and Development, Numerical Models (FE 13)

---

**Memory and Parallelization Layout**

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