https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

Directive-based GPU acceleration of anthropogenic and biosphere flux computations in ICON-ART (v2026.04)

Arash Hamzehloo[1,2], Erik F. M. Koene[1], Nikolai Ponomarev[1,3], Michael Steiner[1,4], Sven Werchner[5], Xavier Lapillonne[6], William Sawyer[7], Michael Jähn[8], Lukas Emmenegger[1], and Dominik Brunner[1]

[1]Empa, Swiss Federal Laboratories for Materials Science and Technology, Dübendorf, Switzerland
[2]now at: ETH AI Center, ETH Zürich, Zürich, Switzerland
[3]now at: School of GeoSciences, University of Edinburgh, Edinburgh, United Kingdom
[4]now at: Environmental Defense Fund, Amsterdam, The Netherlands
[5]Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
[6]Federal Office of Meteorology and Climatology (MeteoSwiss), Zürich Airport, Switzerland
[7]Swiss National Supercomputing Center (CSCS), Lugano, Switzerland
[8]Center for Climate Systems Modeling (C2SM), ETH Zürich, Zürich, Switzerland

Correspondence: Arash Hamzehloo (arash.hamzehloo@empa.ch) and Dominik Brunner (dominik.brunner@empa.ch)

Abstract.

Atmospheric composition models require efficient computation of surface fluxes from anthropogenic emissions and terrestrial biosphere exchange. This is particularly true for applications such as ensemble-based inverse emission modeling where these calculations constitute significant performance bottlenecks. We present the directive-based GPU acceleration and comprehensive refactoring of two critical surface-flux modules within the ICON-ART atmospheric modeling framework: the Online Emission Module (OEM) for anthropogenic emissions and the Vegetation Photosynthesis and Respiration Model (VPRM) for biospheric Carbon Dioxide (CO_2) exchange. The implementation employs OpenACC compiler directives to expose inherent data parallelism while maintaining a single portable source code for both CPU and GPU architectures.

The optimization strategy addresses fundamental incompatibilities between the original CPU-oriented algorithms and GPU execution requirements through three principal modifications: (i) elimination of runtime string comparisons via pre-computed integer-indexed lookup tables constructed during initialization, (ii) temporal decomposition of computations into sections with distinct update frequencies (initialization (once), boundary conditions (periodic), temporal scaling (hourly), and emission kernels (every timestep)) to minimize redundant calculations, and (iii) reorganization of loop hierarchies to maximize memory coalescing across ICON's horizontal grid dimension. The hybrid CPU–GPU architecture executes initialization and I/O-intensive boundary updates on the host while offloading the computationally dominant temporal scaling and emission kernels to the device, with all tracer tendencies accumulated in device-resident arrays to eliminate host–device transfers within the physics integration loop.

Scientific equivalence between CPU and GPU implementations is validated using a probabilistic testing framework, which establishes tolerance envelopes from perturbed CPU ensemble runs. This is consistent with spatial comparisons of simulated CO_2 concentration fields over a European domain, which demonstrate pixel-wise agreement within $\pm 0.01\%$, corresponding to the numerical noise floor from floating-point arithmetic variations. Performance benchmarks on NVIDIA Grace–Hopper

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

(GH200) superchips of the ALPS supercomputer in Switzerland reveal speedups of ∼1.9× on a single GPU node to 5.8× on five GPU nodes relative to a 278-rank CPU baseline. Despite 54–93% higher absolute energy consumption on GPUs, the energy–delay product improves by 20–67%, demonstrating favorable computational efficiency for time-critical applications.

These advances enable ensemble-based atmospheric inversions with hundreds of tracers at kilometer-scale resolution within operationally feasible time constraints, supporting assimilation of dense satellite observations from instruments such as TROPOMI and the forthcoming CO2M mission. The GPU-enabled modules have been successfully deployed in recent ICON-ART applications for regional greenhouse-gas inversion and ensemble data assimilation. The single-source, directive-based implementation ensures long-term maintainability and portability across GPU generations while providing a reusable template for accelerating additional model components.

1 Introduction

The continuous evolution of climate, weather and atmospheric composition models toward higher resolution and more comprehensive process representation has imposed increasingly stringent demands on computational performance. One of the key components in atmospheric composition models is the computation of surface fluxes, which may become time-critical when fluxes of dozens or hundreds of tracers need to be computed. In the ICON-ART modeling framework (Rieger et al., 2015; Schröter et al., 2018; Hoshyaripour et al., 2026), which extends the ICOsahedral Nonhydrostatic (ICON) Numerical Weather Prediction (NWP) model (Zängl et al., 2015) with capabilities for simulating Aerosols and Reactive Trace (ART) gases, these fluxes are derived from a variety of modules reading fluxes from external files (e.g., emissions from wildfires) or computing them online during the simulation (e.g., sea salt and mineral dust). Here, we address two of these modules, the Online Emission Module (OEM) (Jähn et al., 2020) for anthropogenic emissions and the Vegetation Photosynthesis and Respiration Model (VPRM) (Mahadevan et al., 2008) for biospheric Carbon Dioxide (CO_2) exchange. Both modules are essential for greenhouse-gas modeling, air quality applications, and inverse modeling, yet their repeated evaluation across hundreds of thousands of grid cells and sub-hourly time steps makes them computationally demanding components of ICON-ART. This is particularly true for ensemble-based inverse modeling, where fluxes for typically a few hundred ensemble tracers need to be computed.

ICON is a state-of-the-art, nonhydrostatic atmospheric model jointly developed by the German Weather Service (DWD), the Max Planck Institute for Meteorology (MPI-M), and the Center for Climate Systems Modeling (C2SM) (Zängl et al., 2015; Lapillonne et al., 2026). It employs an icosahedral grid that provides quasi-uniform spatial resolution and supports both global and one-way and two-way nested regional configurations (Zängl et al., 2022). ICON has been designed for scalability on modern High-Performance Computing (HPC) systems, and in recent years it has been fully ported (Giorgetta et al., 2022; Lapillonne et al., 2026) to Graphics Processing Unit (GPU) architectures using OpenACC compiler directives (Wolfe et al., 2018). This port enabled MeteoSwiss to become the first national weather service to run an operational ensemble weather prediction system entirely on GPUs, demonstrating that directive-based acceleration can yield production-quality performance while maintaining code portability and scientific integrity (Lapillonne et al., 2026). Building on this foundation, ICON-ART inherits the same infrastructure but includes additional processes such as chemical reactions, aerosol dynamics, and interactive

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

emissions (Hoshyaripour et al., 2026). The increasing use of ICON-ART not only for atmospheric chemistry and aerosol applications but also for inverse modeling of greenhouse gas emissions using Ensemble Square Root Filters (Steiner et al., 2024; Thanwerdas et al., 2025) has highlighted the need to extend GPU acceleration beyond ICON's dynamical core to composition-relevant modules.

OEM was originally introduced in the COSMO-GHG and COSMO-ART models (Jähn et al., 2020) to handle emissions from anthropogenic sources dynamically during runtime. Instead of relying on preprocessed hourly emission fields, OEM reads compact inventories of individual source sectors at model initialization and applies sector-specific temporal and vertical scaling profiles within the model integration. This "online" approach drastically reduces I/O demands and allows emission strengths to respond to meteorological drivers or scenario perturbations. Within ICON-ART, OEM provides gridded emission tendencies for air pollutants, greenhouse gases or other tracers from anthropogenic sources. OEM was recently extended to support the generation of ensembles of perturbed emission fluxes, which allows ICON-ART to be coupled efficiently with well-established inversion frameworks such as the Carbon Tracker Data Assimilation Shell (CTDAS) (Van Der Laan-Luijkx et al., 2017; Steiner et al., 2024) or the Community Inversion Framework (CIF) (Berchet et al., 2021; Thanwerdas et al., 2025). The perturbed flux fields are generated during initialization from an ensemble of flux scaling factors applied in regions defined by masks, where a mask can be any combination of grid cells including an individual cell. Temporal and vertical scaling is then applied online during simulation in the same way as for regular flux fields. Since the quality of the inversion depends on ensemble size, ideally a few hundred perturbed flux fields and corresponding tracers are simulated (Thanwerdas et al., 2025), which places a high demand on the computational efficiency of OEM.

Similarly, biospheric CO_2 fluxes are simulated online through VPRM, which parametrizes Gross Primary Production (GPP) and ecosystem respiration using satellite-derived vegetation greenness and soil moisture indices, and meteorological drivers such as temperature and shortwave radiation (Mahadevan et al., 2008). Like OEM, VPRM has been augmented with the capability to generate ensembles of perturbed GPP and respiration fluxes to enable inverse modeling of anthropogenic and biospheric CO_2 fluxes as demonstrated by Ponomarev et al. (2026). Both algorithms are inherently local and highly data-parallel, making them prime candidates for acceleration on GPU architectures.

As the ICON-ART framework advances toward kilometer-scale resolution, the computational share of OEM and VPRM can reach several percent of the total model cost (Thanwerdas et al., 2025). In inverse modeling applications with large tracer ensembles, this contribution becomes significant. The current study therefore focuses on refactoring and accelerating the OEM and VPRM algorithms for execution on GPUs using OpenACC. Our objective is to achieve substantial reductions in runtime without altering the scientific results or compromising portability and maintainability of the source code.

Directive-based programming with OpenACC provides a pragmatic pathway for adapting large FORTRAN-based NWP frameworks, such as ICON-ART, to heterogeneous HPC architectures. Unlike low-level approaches such as CUDA or HIP (Herdman et al., 2012; Malenza et al., 2024), OpenACC allows incremental porting by adding compiler directives that guide the parallelization and data movement between host (CPU) and device (GPU) memories. This methodology has proven effective in the GPU port of ICON (Lapillonne et al., 2026) and ensures that the same source code remains functional on CPU-based systems. It should be noted that the CPU execution path relies on MPI parallelism alone (without additional threading via

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

OpenMP), which is the standard configuration for ICON and ICON-ART in production; the OpenACC directives are simply ignored by the compiler when building for CPU targets. The resulting code portability—a single source that compiles and runs correctly on both CPU and GPU architectures—is the primary design goal. While the present work targets NVIDIA GPUs via the NVHPC compiler, the OpenACC standard is supported by multiple compilers, including Cray/HPE and GCC, and recent progress in OpenACC-to-HIP translation layers suggests that adaptation to AMD GPU architectures is feasible without fundamental code changes. The adoption of OpenACC for ICON-ART thus aligns with the broader design philosophy of source-code portability across computing platforms used by operational and research institutions within the ICON partnership (ICON Partnership, 2026).

In this work, we describe the reorganization of data structures, loop hierarchies, and memory layouts in the OEM and VPRM routines to enable efficient GPU execution while preserving compatibility with existing ICON-ART infrastructure (ICON Partnership, 2026). The modifications include persistent data regions for long-lived model state variables, optimized GPU parallelization of grid loops with OpenACC directives (the specific parallelization concepts are detailed in Sect. 4), and explicit privatization of temporary scalars to prevent race conditions. Numerical equivalence between CPU and GPU results is verified through probabilistic testing and direct field comparisons.

The remainder of this paper is organized as follows. Section 2 provides an overview of the ICON-ART framework, with emphasis on the algorithmic structure of OEM and VPRM. Section 3 details the code refactoring strategy, including the elimination of string-based lookups, temporal decomposition of computations, and the hybrid CPU–GPU architecture. Section 4 describes the GPU implementation of OEM, covering the main emission kernel, ensemble tracer handling, and memory access optimization. Section 5 presents the corresponding implementation for VPRM, including the two-stage flux computation and vegetation-class-specific ensemble scaling. Section 6 reports performance benchmarks on the ALPS supercomputer in Switzerland, analyzing strong scaling behavior, parallel efficiency, and energy consumption characteristics. Section 7 addresses GPU code evaluation. Section 8 discusses implications for high-resolution atmospheric inversion applications, demonstrated scientific use cases, and prospects for extending GPU acceleration to additional ICON-ART components. Finally, Section 9 summarizes the principal findings and outlines directions for future development.

2 Model Description

2.1 Architecture of ICON-ART

ICON-ART extends the ICON NWP framework with prognostic tracers and coupled process components for emissions, gas-phase chemistry, and aerosol microphysics (Rieger et al., 2015; Schröter et al., 2018; Hoshyaripour et al., 2026). ART is organized as an extension that communicates with ICON through explicit interfaces for tracer state, tendencies, and diagnostics (Hoshyaripour et al., 2026). Within each physics time step, advective transport and physical parameterizations of ICON are advanced first; ART processes are then invoked to compute source, sink, and transformation tendencies for all enabled tracers, which are added directly to ICON's prognostic arrays before the next dynamical substep. The configuration follows a modular design in which each process (e.g., emissions, chemistry, dry and wet deposition, sedimentation) is encapsulated

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

in its own module and can be activated and parameterized via namelists and XML configuration files. This layout facilitates performance-oriented refactoring, including the introduction of accelerator-specific data regions and kernel device offloading, while preserving a unified source base for CPU and GPU execution (Hoshyaripour et al., 2026).

The present work focuses on the surface-flux subsystem corresponding to a near-surface source term for many tracers. Anthropogenic emissions are computed online by OEM (Jähn et al., 2020), while biospheric exchange of CO_2 is represented by VPRM (Mahadevan et al., 2008). Both components operate on the native ICON grid and at the model physics cadence, thereby avoiding temporal discretization artifacts associated with pre-expanded hourly fields. Both OEM and VPRM are called from the ART emissions driver after the ICON surface and boundary-layer parameterizations have provided the required meteorological inputs (shortwave radiation and near-surface temperature for VPRM; current time for OEM). The modules return 2D surface (VPRM) or 3D volume (OEM) tendencies that are summed into the shared tracer tendency arrays and subsequently transported by ICON's diffusion and, where active, convection schemes. Their algorithmic structure is inherently data-parallel over horizontal grid points and, to a limited extent, over categories and vertical allocation indices, which makes them suitable candidates for GPU acceleration. A detailed account of the ICON-ART framework and its ART components is provided by Hoshyaripour et al. (2026).

2.2 Online Emission Module

OEM computes three-dimensional, time-dependent emissions at runtime from compact inputs read once at initialization. The inputs comprise annual or mean sectoral emission fields already mapped to the ICON grid, together with auxiliary tables defining temporal modulation, vertical allocation profiles, and, where applicable, speciation factors that map inventory families to model species (Jähn et al., 2020). Let X denote a model species and $s \in \{1, \ldots, N_s\}$ a source category. The emission tendency in model layer k at time t is

$$E_{X,k}(t) \;=\; \sum_{s=1}^{N_s} E_{X,s}\, w_{h,s}\big(h(t)\big)\, w_{d,s}\big(d(t)\big)\, w_{m,s}\big(m(t)\big)\, v_{s,k}, \tag{1}$$

where $E_{X,s}$ is the annual flux in the grid cell, $w_{h,s}$, $w_{d,s}$, and $w_{m,s}$ are unit-mean scalings for hour-of-day $h(t)$, day-of-week $d(t)$, and month-of-year $m(t)$, which are functions that extract the respective datetime components from the simulation time t, and $v_{s,k}$ is a normalized vertical profile.

In ICON-ART, OEM is invoked every physics time step. The implementation proceeds through initialization of persistent working arrays, update of lateral boundary tendencies in limited-area mode, evaluation of temporal profiles for the current time, interpolation to the sub-hourly physics cadence if enabled, and assembly of three-dimensional tendencies that are added to the tracer fields. It is worth noting that updating the tendencies at the lateral boundaries enables scaling of the tracer ensembles at those boundaries. This, in turn, allows optimization not only of emissions but also of background concentrations, which are determined by inflow from such boundaries.

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

2.3 Vegetation Photosynthesis and Respiration Model (VPRM)

VPRM provides the biospheric contribution to the surface CO_2 exchange as the difference between temperature-driven ecosystem respiration R and light-dependent GPP (Mahadevan et al., 2008),

$$\mathrm{NEE} = \mathrm{R} - \mathrm{GPP}, \tag{2}$$

with positive values denoting net release to the atmosphere and negative values denoting net uptake (all fluxes in μmol CO_2 m^{-2} s^{-1}). NEE denotes the Net Ecosystem Exchange. GPP is formulated as a light-use efficiency scheme modulated by environmental scalars (temperature and short wave radiation) and satellite indices of vegetation greenness (Enhanced Vegetation Index, EVI) and soil moisture (Land Surface Water Index, LSWI). Respiration is approximated as a linear function of near-surface air temperature T (in °C) with vegetation class-specific coefficients (a_ℓ, b_ℓ) calibrated against eddy-covariance observations (Mahadevan et al., 2008). Each ICON grid cell contains fractional areas of a small set of vegetation classes. The model computes class-wise GPP and $\mathrm{R}_\ell = a_\ell T + b_\ell$ for all classes ℓ represented in the cell and forms a weighted sum using the class fractions to obtain the net ecosystem exchange.

3 Code Architecture and GPU Optimization Strategy

The OEM and VPRM modules within ICON-ART were originally developed for CPU-based execution with a focus on scientific accuracy and flexibility. However, the computational demands of high-resolution simulations with large ensemble configurations necessitate a comprehensive refactoring and optimization effort to enable efficient execution on GPU-accelerated HPC systems. This section details the systematic approach undertaken to port these modules to GPUs using OpenACC directives while maintaining full backward compatibility with CPU execution.

To facilitate the subsequent technical discussion, Table 1 summarizes the main loop indices and array dimensions used throughout the ICON-ART emission modules. In ICON, the horizontal domain is partitioned into blocks of cells, with each block containing up to `nproma` cells. This blocking structure enables efficient vectorization on CPUs and coalesced memory access on GPUs.

3.1 Original Code Structure and Identified Bottlenecks

Prior to optimization, a thorough profiling analysis was conducted to identify computational bottlenecks in both modules. The original implementations exhibited several characteristics that severely impacted GPU performance potential.

3.1.1 String-Based Index Lookups

The original implementations relied extensively on string comparisons within performance-critical loops to resolve category and tracer indices. For instance in OEM, for each grid cell at every timestep, multiple string comparison operations were invoked:

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

Table 1. Summary of main loop indices and array dimensions used in the ICON-ART emission modules.

Symbol	Description	Typical Range
Loop indices		
`jc`	Horizontal cell index within a block	1 to `nproma`
`jb`	Block index	`i_startblk` to `i_endblk`
`k`	Vertical level index	1 to `nlev`
`nt`	Tracer (species) index	1 to `emis_tracer`
`nc`	Emission category index	1 to `ncat_max`
Array dimensions		
`nproma`	Block length (cells per block)	8 (CPU), $\mathcal{O}(10^4)$ (GPU)
`nblk`	Number of blocks	`nproma`-dependent (CPU), 1 (GPU)
`nlev`	Number of vertical levels	60–90
`emis_tracer`	Number of emission tracers	1–200
`ncat_max`	Maximum emission categories	2–20 (inverse), 50–100 (air quality)
`n_ens_max`	Maximum ensemble members	up to 500

Listing 1. Original string-based index lookup pattern in OEM (simplified). The configuration arrays `ycatl_l`, `yvpl_l`, and `ytpl_l` store category names, vertical profile identifiers, and temporal profile identifiers, respectively, for each tracer-category combination.

```
DO nt = 1, emis_tracer
  DO nc = 1, ncat_max
    IF (ycatl_l(nt,nc) /= "") THEN
      ! String comparison: O(n) search through array
      grd_index = get_gridded_emissions_idx(ycatl_l(nt,nc))
      vp_cat_idx = get_category_idx(yvpl_l(nt,nc), 'vp')
      tp_cat_idx = get_category_idx(ytpl_l(nt,nc), 'tp')

      DO jb = i_startblk, i_endblk
        DO jc = is, ie
          ! Another string-based linear search per cell
          tp_country_idx = get_country_idx(country_ids(jc,jb))
          ...
        END DO
      END DO
    END IF
  END DO
END DO
```

These string operations are fundamentally incompatible with GPU architectures, which lack efficient hardware support for variable-length string comparisons and exhibit severe performance degradation due to thread divergence caused by conditional branching within string comparison routines.

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

3.1.2 Suboptimal Memory Access Patterns

The original loop nesting ordering prioritized the tracer and category dimensions as outermost loops, resulting in non-coalesced memory access patterns on GPUs. For GPU architectures, optimal memory coalescing requires that consecutive threads access consecutive memory locations. The original ordering:

$$\text{Loop order: } \underbrace{\text{nt}}_{\text{outer}} \rightarrow \underbrace{\text{nc}} \rightarrow \underbrace{\text{jb}} \rightarrow \underbrace{\text{jc}} \rightarrow \underbrace{\text{k}}_{\text{inner}} \tag{3}$$

caused strided memory access with stride lengths proportional to the problem dimensions (i.e., the number of tracers and categories), resulting in poor cache utilization and memory bandwidth efficiency. The optimized loop ordering places the block loop (`jb`) as the outermost sequential loop, with the horizontal cell (`jc`) and vertical level (`k`) loops parallelized and collapsed for GPU execution, while the tracer (`nt`) and category (`nc`) loops remain sequential in the innermost positions:

$$\text{Optimized loop order: } \underbrace{\text{jb}}_{\text{outer (seq)}} \rightarrow \underbrace{\text{jc}}_{\text{parallel}} \rightarrow \underbrace{\text{k}}_{\text{collapsed}} \rightarrow \underbrace{\text{nt}}_{\text{seq}} \rightarrow \underbrace{\text{nc}}_{\text{inner (seq)}} \tag{4}$$

3.1.3 Redundant Computations

Several quantities that remain constant throughout the simulation or change only at specific intervals were recomputed at every timestep: the product of gridded emissions and vertical scaling factors (constant throughout simulation), temporal scaling factors (constant within each hour), and country-to-temporal-profile mapping (constant throughout simulation).

Table 2 summarizes the identified bottlenecks and their estimated performance impact.

Table 2. Summary of identified performance bottlenecks in the original OEM and VPRM implementations. Impact severity is categorized as Critical (preventing GPU execution), High (>50% performance penalty), Medium (10–50% penalty), or Low (<10% penalty).

Bottleneck	Module	Severity	Root Cause
String comparison in inner loop	OEM	Critical	`get_gridded_emissions_idx()`, `get_category_idx()`
Linear search per cell	OEM	High	`get_country_idx()`
String comparison for flux type	VPRM	High	`vprm_flux_type == 'resp'/'gpp'`
Non-coalesced memory access	OEM, VPRM	Medium	Loop order: nt→nc→jb→jc
Redundant temporal scaling	OEM	Medium	Full recomputation every timestep
Branch divergence in vegetation loop	VPRM	Medium	Conditional `IF (l == i_vprm_lc_*)`
Repeated weight computation	OEM	Low	dtime × emissions × vert_scale

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

4 Implementation of GPU-Optimized Online Emission Module

This section provides detailed documentation of the GPU-optimized OEM implementation, including the data structures, OpenACC directives, and algorithmic modifications.

4.1 GPU Optimization Strategy

The optimization strategy follows four fundamental principles: (1) elimination of string operations from GPU kernels through pre-computation, (2) restructuring of computational phases to minimize redundant calculations, (3) reorganization of loop structures to maximize memory coalescing, and (4) separation of distinct computational pathways into dedicated GPU kernels to minimize thread divergence. An overarching design goal is to keep all performance-critical computations on the GPU so that host–device data transfers are limited to the infrequent initialization and boundary-update phases (Sects. 4.1.1 and 4.1.2).

4.1.1 Pre-computation and Lookup Table Generation

The cornerstone of the GPU optimization is the systematic replacement of all string-based operations with integer-indexed array accesses through pre-computed lookup tables. This transformation is performed once during the initialization phase (first timestep) on the CPU, with the resulting tables transferred to GPU memory for subsequent access. We introduce the concept of "active category pairs": combinations of tracer index n_t and category index n_c for which emissions are defined (i.e., `ycatl_l(nt,nc)` $\neq$ `""`). Rather than iterating over all possible (n_t, n_c) combinations and checking for validity, we pre-enumerate only the active pairs,

$$\mathcal{A} = \{(n_t, n_c) : \texttt{ycatl\_l}(n_t, n_c) \neq \texttt{""},\ n_t \in [1, N_{\text{tracer}}],\ n_c \in [1, N_{\text{cat}}]\}. \tag{5}$$

For each active pair $i \in \mathcal{A}$, we store the pre-resolved integer indices,

$$\texttt{active\_nt}(i) = n_t \tag{6}$$

$$\texttt{active\_nc}(i) = n_c \tag{7}$$

$$\texttt{active\_grd\_idx}(i) = \text{index into gridded emissions array} \tag{8}$$

$$\texttt{active\_vp\_idx}(i) = \text{index into vertical profile array} \tag{9}$$

$$\texttt{active\_tp\_idx}(i) = \text{index into temporal profile array} \tag{10}$$

$$\texttt{active\_trcr\_idx}(i) = \text{index into tracer array} \tag{11}$$

Similarly, for ensemble simulations, we construct lookup tables that map ensemble member indices directly to their corresponding tracer indices, eliminating the need for iterative string matching,

$$\texttt{ens\_lookup\_int\_idx}(n_{\text{ens}}, n_t) = \text{tracer index for ensemble member } n_{\text{ens}} \text{ of tracer } n_t \tag{12}$$

$$\texttt{ens\_lookup\_table\_nr}(n_{\text{ens}}, n_t) = \text{ensemble table entry number (or } -1 \text{ if not found)} \tag{13}$$

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

The country-to-temporal-profile mapping, originally requiring a linear search through all country identifiers for each grid cell, is similarly pre-computed into a direct-access array,

$$\texttt{tp\_country\_idx}(j_c, j_b) = \ell : \texttt{country\_ids}(j_c, j_b) = \texttt{tp\_countryid}(\ell) \tag{14}$$

4.1.2 Temporal Decomposition of Computations

The refactored OEM code architecture employs a sectioned approach that separates computations based on their temporal update frequency. This design minimizes redundant calculations while ensuring data consistency across the simulation:

1. Section 1 (Initialization, $t = 0$): Executes once on the CPU at simulation startup ($t = 0$). This section constructs the lookup tables, index arrays, and pre-computed emission weights (gridded and vertical profiles) that eliminate the need for string comparisons and repeated calculations during the main simulation loop. Upon completion, the resulting data structures are transferred to GPU device memory using `!$ACC ENTER DATA COPYIN` directives, where they remain resident for the duration of the simulation. This one-time transfer cost is amortized over potentially millions of subsequent timesteps.

2. Section 2 (Boundary Update, Δt_{BC}): Executes on the CPU at the boundary-forcing interval Δt_{BC} (typically 3–6 hours). This section reads updated lateral boundary conditions for tracer tendencies from external forcing files and performs the necessary interpolation and unit conversions. The updated boundary fields are then copied to GPU memory using `!$ACC UPDATE DEVICE` directives. Keeping this section on the CPU leverages existing I/O infrastructure and avoids the complexity of GPU-based file reading, while the relatively infrequent execution (every few hours, depending on the frequency at which lateral boundary conditions are supplied) results in negligible host-to-device transfer overhead.

3. Section 3 (Hourly Update, $\Delta t = 1\,\mathrm{h}$): Executes entirely on the GPU at hourly intervals. This section computes the temporal scaling factors that modulate emissions based on diurnal, weekly, and seasonal cycles. By pre-computing values for both the current hour ($\mathrm{tsf}_{\mathrm{now}}$) and the following hour ($\mathrm{tsf}_{\mathrm{next}}$), the implementation enables sub-hourly linear interpolation within Section 4 without requiring additional kernel launches or host–device synchronization during each timestep.

4. Section 4 (Every Timestep, Δt): The main computational core of OEM, executing on the GPU at every physics timestep Δt (typically 5–120 seconds). This kernel combines the pre-computed emission weights with interpolated temporal scaling factors to update tracer mixing ratio tendencies. Because all required data (lookup tables, weights, temporal factors, and tracer arrays) reside in GPU device memory, Section 4 executes without any host–device data transfers, maximizing kernel throughput.

5. Section 5 (Restart, once): Initialization of tracer concentrations from restart files when applicable.

Figure 1 illustrates this OEM sectioned architecture and the data flow between components. The key advantage of this hybrid CPU–GPU architecture is that it places the most frequently executed computations (Sections 3 and 4) on the GPU

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

while retaining I/O-intensive operations (Sections 1 and 2) on the CPU where they can leverage existing infrastructure. This distribution ensures that the vast majority of computational work occurs on the GPU without host–device communication overhead.

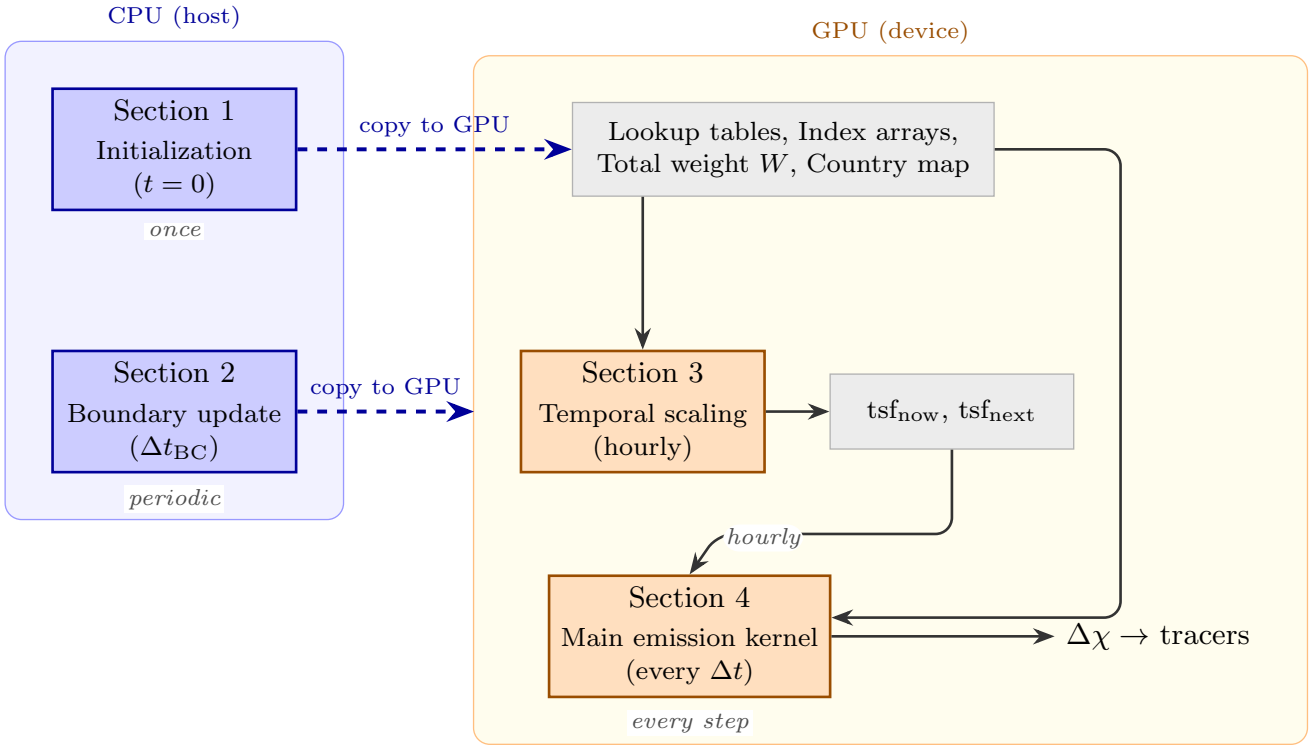

Figure 1. Schematic of the GPU-optimized OEM architecture illustrating the temporal decomposition of computations and the distribution between CPU and GPU. Sections 1 and 2 (blue, left) execute on the CPU: Section 1 performs one-time initialization at $t = 0$, constructing lookup tables, index arrays, total emission weights, and country mappings, which are then transferred to GPU device memory via `!$ACC ENTER DATA COPYIN`. Section 2 updates lateral boundary conditions at interval Δt_{BC} and copies the updated fields to the GPU. Sections 3 and 4 (orange, right) execute entirely on the GPU: Section 3 computes temporal scaling factors hourly, storing values for the current and next hour (tsf_{now}, tsf_{next}) to enable sub-hourly linear interpolation; Section 4 is the main emission kernel executing at every physics timestep Δt, combining pre-computed weights with interpolated temporal factors to update tracer mixing ratio tendencies $\Delta\chi$. Dashed blue arrows indicate host-to-device data transfers; solid black arrows indicate data flow within the GPU.

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

4.1.3 Pre-computed Emission Weights

A significant optimization is the pre-computation of the combined emission weight factor, which remains constant throughout the simulation. In the original implementation, the tracer mixing ratio χ was updated at every timestep as:

$$\chi_{j_c,k,j_b,\tau}(t+\Delta t) = \chi_{j_c,k,j_b,\tau}(t) + \frac{\Delta t \cdot \tilde{E}_{j_c,j_b,s} \cdot v_{k,s} \cdot w_s(t)}{\rho_{\text{air}}(j_c,k,j_b,t)}, \tag{15}$$

where χ denotes the tracer mixing ratio, τ is the tracer index, $\tilde{E}$ is the gridded annual mean emission, v is the vertical scaling factor, $w(t)$ is the temporal scaling factor, and ρ_{air} is the air mass per grid cell layer (kg m^{-2}, corresponding to ICON's `airmass_now` variable). We observe that the product $\Delta t \cdot \tilde{E} \cdot v$ is time-invariant and can be pre-computed once. We therefore define the total emission weight:

$$W_{j_c,k,n_t,n_c,j_b} = \Delta t \cdot \tilde{E}_{j_c,j_b,g(n_t,n_c)} \cdot v_{j_c,k,j_b,p(n_t,n_c)}, \tag{16}$$

where $g(n_t,n_c)$ and $p(n_t,n_c)$ are the pre-computed gridded emission and vertical profile indices, respectively. The main kernel then simply computes:

$$\chi_{j_c,k,j_b,\tau}(t+\Delta t) = \chi_{j_c,k,j_b,\tau}(t) + \frac{W_{j_c,k,n_t,n_c,j_b} \cdot w_s(t)}{\rho_{\text{air}}(j_c,k,j_b,t)}, \tag{17}$$

which separates the time-invariant weight W (computed once during initialization) from the time-varying components $w_s(t)$ and $\rho_{\text{air}}(t)$ that must be evaluated at each timestep.

4.2 Data Structures for GPU Optimization

The GPU-optimized OEM introduces several module-level arrays to store the pre-computed lookup tables and index mappings described conceptually in Sect. 4.1.1. This section documents their concrete FORTRAN declarations. These arrays are declared with the `ALLOCATABLE` attribute to allow dynamic sizing based on the simulation configuration:

Listing 2. GPU optimization data structures in OEM.

```
! Active category pairs (nt, nc) where ycatl_1(nt,nc) /= ""
INTEGER :: n_active_cats = 0
INTEGER, ALLOCATABLE :: active_nt(:), active_nc(:)
INTEGER, ALLOCATABLE :: active_grd_idx(:), active_vp_idx(:), active_tp_idx(:)
INTEGER, ALLOCATABLE :: active_trcr_idx(:)

! Flags for ensemble and diagnostic tracers (pre-computed on CPU)
LOGICAL, ALLOCATABLE :: is_ensemble_tracer(:)
INTEGER, ALLOCATABLE :: ens_count_map(:)  ! Maps nt to ens_count
LOGICAL, ALLOCATABLE :: is_diag_output(:)

! Pre-computed ensemble lookup table: (nens, nt) -> indices
INTEGER, ALLOCATABLE :: ens_lookup_table_nr(:,:)
INTEGER, ALLOCATABLE :: ens_lookup_int_idx(:,:)

! Category lambda indices for ensemble
```

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026


```
INTEGER, ALLOCATABLE :: nc_ensemble_map(:)

! Pre-computed temporal scaling factors and total weights
REAL(KIND=wp), ALLOCATABLE :: tsf_now(:,:,:,:), tsf_next(:,:,:,:)
REAL(KIND=wp), ALLOCATABLE :: total_weight(:,:,:,:,:)

! Country index mapping (pre-computed)
INTEGER, ALLOCATABLE :: tp_country_idx(:,:)
```

The dimensionality of these arrays reflects the problem structure:

- `active_*`: dimension (N_{active}) where $N_{\text{active}} = |\mathcal{A}|$
- `is_ensemble_tracer`, `is_diag_output`: dimension (N_{tracer})
- `ens_lookup_*`: dimension $(N_{\text{ens,max}}, N_{\text{tracer}})$.
- `tsf_now`, `tsf_next`: dimension $(\texttt{nproma}, N_{\text{tracer}}, N_{\text{cat}}, N_{\text{blk}})$ for sectoral temporal scaling factors
- `total_weight`: dimension $(\texttt{nproma}, N_{\text{lev}}, N_{\text{tracer}}, N_{\text{cat}}, N_{\text{blk}})$
- `tp_country_idx`: dimension $(\texttt{nproma}, N_{\text{blk}})$

It is worth noting that `nproma` in ICON defines the block length of data rows (arrays), chosen to optimally exploit the memory hierarchy of HPC systems and thereby improve computational efficiency. Its value is defined differently for CPU and GPU executions (Lapillonne et al., 2026), as will be discussed later in this paper.

The data structures listed in Listing 2 belong to the unified single-source base of the module: they are allocated and populated identically on CPU and GPU, and the only architecture-specific elements are the `!$ACC` directives that move them to device memory, which the compiler silently ignores for CPU targets. We did not introduce a code bifurcation. In our experience, the refactor also makes the CPU code path easier to read than the original implementation: the per-cell string comparisons and linear searches in the inner loops have been replaced by a small set of integer-indexed lookup arrays whose meaning is documented in their declarations, and the inner kernels (Sects. 4.1.1–4.1.3) now contain only arithmetic on those pre-resolved indices. The additional module-level allocations introduce a modest declaration overhead but reduce the inner-loop logic, so the net effect on CPU code clarity is positive.

4.3 Initialization Phase (Section 1)

The initialization phase executes once at the first timestep and performs all pre-computations required for efficient GPU execution. Algorithm 1 presents the pseudocode for this phase.

Algorithm 1 OEM Initialization Phase (Section 1)

Require: Configuration data, sectoral emission fields

Ensure: Pre-computed lookup tables on GPU

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

// Step 1: Normalize names to lowercase
for $n_t = 1$ **to** N_{tracer} **do**
  $\texttt{emis\_name}(n_t) \leftarrow \texttt{tolower}(\texttt{emis\_name}(n_t))$
end for
// Step 2: Count and enumerate active categories
$N_{\text{active}} \leftarrow 0$
for $n_t = 1$ **to** N_{tracer} **do**
  for $n_c = 1$ **to** N_{cat} **do**
    if $\texttt{ycatl\_l}(n_t, n_c) \neq$ "" **then**
      $N_{\text{active}} \leftarrow N_{\text{active}} + 1$
    end if
  end for
end for
Allocate arrays of size N_{active}
// Step 3: Populate active category arrays with resolved indices
$i_{\text{cat}} \leftarrow 0$
for $n_t = 1$ **to** N_{tracer} **do**
  for $n_c = 1$ **to** N_{cat} **do**
    if $\texttt{ycatl\_l}(n_t, n_c) \neq$ "" **then**
      $i_{\text{cat}} \leftarrow i_{\text{cat}} + 1$
      Resolve and store all integer indices for pair (n_t, n_c)
    end if
  end for
end for
// Step 4: Pre-compute ensemble lookup tables
if $N_{\text{ens}} > 0$ **then**
  Build $\texttt{ens\_lookup\_int\_idx}(n_{\text{ens}}, n_t)$ for all combinations
end if
// Step 5: Compute total weights and country mapping
for all grid cells (j_c, j_b) **do**
  $\texttt{tp\_country\_idx}(j_c, j_b) \leftarrow$ FINDCOUNTRYINDEX
  for each active category i (with $n_t = \texttt{active\_nt}(i)$, $n_c = \texttt{active\_nc}(i)$), each level k **do**
    $\texttt{total\_weight}(j_c, k, n_t, n_c, j_b) \leftarrow \Delta t \cdot \tilde{E} \cdot v$
  end for
end for

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

36: **// Step 6: Transfer data to GPU**
37: `!$ACC ENTER DATA COPYIN(lookup tables, weights, configuration)`

4.4 Main Emission Kernel (Section 4)

The main emission kernel executes at every physics timestep and represents the most performance-critical component of OEM. This section performs the actual tracer tendency updates using the pre-computed lookup tables and weights from Section 1, combined with the interpolated temporal scaling factors from Section 3. The implementation employs a two-kernel architecture that separates regular tracer updates from ensemble tracer updates, thereby minimizing thread divergence and maximizing GPU utilization.

4.4.1 Algorithmic Structure

Algorithm 2 presents the complete pseudocode for Section 4. Because the paper specifically addresses OpenACC porting, the pseudocode retains OpenACC-specific parallelization annotations (e.g., `GANG`, `VECTOR`, `COLLAPSE`) rather than abstract parallel-loop notation, so that the mapping between algorithm and implementation is immediately apparent. The computation proceeds in three phases: (1) calculation of sub-hourly interpolation weights, (2) regular tracer tendency updates, and (3) ensemble tracer tendency updates with lambda scaling. This separation ensures that the ensemble-specific conditional logic does not introduce branch divergence in the regular tracer kernel, where all threads follow identical execution paths.

Algorithm 2 OEM Main Emission Kernel (Section 4)

Require: Pre-computed weights W, temporal scaling factors $\mathrm{tsf}_{\mathrm{now}}$, $\mathrm{tsf}_{\mathrm{next}}$, lookup tables
Ensure: Updated tracer mixing ratios χ

// Phase 1: Compute sub-hourly interpolation weights
$n_2 \leftarrow \mathrm{seconds\_into\_hour}/3600$
$n_1 \leftarrow 1 - n_2$

// Phase 2: Regular tracer kernel (GPU parallel)
for $j_c = i_s$ **to** i_e **in parallel (GANG) do**
  for $k = 1$ **to** N_{lev} **in parallel (COLLAPSE with** j_c**) do**
    for $i_{\mathrm{cat}} = 1$ **to** N_{active} **sequentially do**
      $n_t \leftarrow \texttt{active\_nt}(i_{\mathrm{cat}})$
      $n_c \leftarrow \texttt{active\_nc}(i_{\mathrm{cat}})$
      $\tau \leftarrow \texttt{active\_trcr\_idx}(i_{\mathrm{cat}})$
      $w_s \leftarrow n_1 \cdot \mathrm{tsf}_{\mathrm{now}}(j_c, n_t, n_c, j_b) + n_2 \cdot \mathrm{tsf}_{\mathrm{next}}(j_c, n_t, n_c, j_b)$
      if $\texttt{is\_diag\_output}(n_t)$ **then**
        $\chi(j_c, k, j_b, \tau) \mathrel{+}= W(j_c, k, n_t, n_c, j_b) \cdot w_s / \Delta t$ {Flux diagnostic}
      else

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

16: $\chi(j_c, k, j_b, \tau) \mathrel{+}= W(j_c, k, n_t, n_c, j_b) \cdot w_s / \rho_{\text{air}}(j_c, k)$ {Mixing ratio}
17: **end if**
18: **end for**
19: **end for**
20: **end for**
21:
22: **// Phase 3: Ensemble tracer kernel (GPU parallel, if ensembles enabled)**
23: **if** $N_{\text{ens}} > 0$ **then**
24: **for** $j_c = i_s$ **to** i_e **in parallel (GANG) do**
25: $r \leftarrow \texttt{reg\_map}(j_c, j_b)$ {Region index for lambda lookup}
26: **for** $k = 1$ **to** N_{lev} **in parallel (VECTOR) do**
27: **for** $i_{\text{cat}} = 1$ **to** N_{active} **sequentially do**
28: **if** $\texttt{is\_ensemble\_tracer}(\texttt{active\_nt}(i_{\text{cat}}))$ **then**
29: $n_t \leftarrow \texttt{active\_nt}(i_{\text{cat}})$; $n_c \leftarrow \texttt{active\_nc}(i_{\text{cat}})$
30: $n_c^{\text{ens}} \leftarrow \texttt{nc\_ensemble\_map}(i_{\text{cat}})$
31: $w_s \leftarrow n_1 \cdot \text{tsf}_{\text{now}}(j_c, n_t, n_c, j_b) + n_2 \cdot \text{tsf}_{\text{next}}(j_c, n_t, n_c, j_b)$
32: **for** $n_{\text{ens}} = 1$ **to** $N_{\text{ens,max}}$ **sequentially do**
33: $\tau_{\text{ens}} \leftarrow \texttt{ens\_lookup\_int\_idx}(n_{\text{ens}}, n_t)$
34: **if** $\tau_{\text{ens}} > 0$ **then**
35: $\lambda \leftarrow \texttt{lambda\_mat}(\texttt{ens\_count}(n_t), n_c^{\text{ens}}, r, n_{\text{ens}})$
36: $\chi(j_c, k, j_b, \tau_{\text{ens}}) \mathrel{+}= W(j_c, k, n_t, n_c, j_b) \cdot w_s \cdot \lambda / \rho_{\text{air}}(j_c, k)$
37: **end if**
38: **end for**
39: **end if**
40: **end for**
41: **end for**
42: **end for**
43: **end if**

4.4.2 Temporal Interpolation

At the start of each timestep, the kernel computes linear interpolation weights for sub-hourly temporal resolution. Given that Section 3 pre-computes temporal scaling factors at the top of each hour (tsf_{now}) and for the following hour (tsf_{next}), the instantaneous scaling factor at time t within the hour is obtained by linear interpolation,

$$w_s(t) = \left(1 - \frac{t - t_h}{3600}\right) \cdot \text{tsf}_{\text{now}} + \frac{t - t_h}{3600} \cdot \text{tsf}_{\text{next}}, \tag{18}$$

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

where t_h is the timestamp at the beginning of the current hour and all times are in seconds. This interpolation preserves the diurnal cycle structure encoded in the temporal profiles while providing smooth transitions between hours, avoiding discontinuities that could introduce numerical artifacts in the tracer fields.

4.4.3 Regular Tracer Kernel

The regular tracer kernel processes all active emission categories and updates the corresponding tracer mixing ratios. While Algorithm 2 above captures the logical flow and parallelization strategy at an abstract level, Listing 3 presents the actual OpenACC implementation, showing the concrete directive syntax and data-access patterns that are essential for reproducibility and for readers adapting the approach to other modules. The kernel is structured to maximize GPU efficiency through several design choices:

1. Collapsed outer loops: The horizontal index j_c and vertical index k are collapsed using `COLLAPSE(2)`, creating a single parallel iteration space of $N_{\text{cells}} \times N_{\text{lev}} \approx 1.8 \times 10^7$ work items for the benchmark configuration. This large iteration count ensures high GPU occupancy.

2. Sequential inner loop: The category loop is marked as `SEQ` because N_{active} is typically small (for atmospheric inverse modeling purposes there are typically 2–20 categories). Parallelizing this dimension would introduce atomic operations or reduction overhead that exceeds the benefit of additional parallelism.

3. Coalesced memory access: The loop ordering places j_c as the fastest-varying index in the collapsed loop, ensuring that consecutive GPU threads access consecutive memory locations in the tracer and weight arrays.

4. Minimal register pressure: The `PRIVATE` clause explicitly declares loop-local variables, preventing the compiler from allocating excessive registers that would reduce occupancy.

Listing 3. Regular tracer emission kernel (Section 4, Phase 2).

```
! Compute sub-hourly interpolation weights
n2 = REAL(mtime_hour_sec, wp) / 3600.0_wp
n1 = 1.0_wp - n2

! Regular tracer kernel - process all active categories
!$ACC PARALLEL DEFAULT(PRESENT) PRIVATE(nt, nc, trcr_idx, temp_scaling_fact)
!$ACC LOOP GANG COLLAPSE(2)
DO jc = is, ie
  DO k = 1, nlev
    !$ACC LOOP SEQ
    DO icat = 1, n_active_cats
      nt = active_nt(icat)
      nc = active_nc(icat)
      trcr_idx = active_trcr_idx(icat)

      ! Linear interpolation of temporal scaling factor
      temp_scaling_fact = n1*tsf_now(jc,nt,nc,jb) + n2*tsf_next(jc,nt,nc,jb)
```

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026


```
18:
19:       IF (is_diag_output(nt)) THEN
20:         ! Diagnostic output: store flux in kg/m2/s
21:         p_tracer_now(jc,k,jb,trcr_idx) = p_tracer_now(jc,k,jb,trcr_idx) + &
22:           &  total_weight(jc,k,nt,nc,jb) * temp_scaling_fact / dtime
23:       ELSE
24:         ! Regular tracer: accumulate mixing ratio tendency
25:         p_tracer_now(jc,k,jb,trcr_idx) = p_tracer_now(jc,k,jb,trcr_idx) + &
26:           &  total_weight(jc,k,nt,nc,jb) * temp_scaling_fact / airmass_now(jc,k,jb)
27:       END IF
28:     END DO ! icat
29:   END DO ! k
30: END DO ! jc
31: !$ACC END PARALLEL
```

The distinction between diagnostic output tracers and regular tracers is handled through a pre-computed boolean flag `is_diag_output(nt)` rather than a runtime string comparison. Diagnostic tracers store the emission flux directly (in $\mathrm{kg\,m^{-2}\,s^{-1}}$) for post-processing purposes, while regular tracers accumulate the mixing ratio tendency by dividing by the air mass.

4.4.4 Ensemble Tracer Kernel

For atmospheric inversion applications, OEM supports ensemble simulations where each ensemble member (tracer) applies a different scaling factor (lambda) to the base emissions. The ensemble kernel, shown in Listing 4, processes these ensemble members efficiently using the pre-computed lookup tables constructed in Section 1. The ensemble kernel differs from the regular kernel in several important ways:

1. Region-dependent lambda lookup: Each grid cell is assigned to a region through the `reg_map` array. The lambda scaling factor is retrieved from a four-dimensional matrix indexed by ensemble count, emission category, region, and ensemble member number. This structure enables region-specific emission perturbations for inverse modeling.

2. Modified parallelization strategy: The ensemble kernel uses `GANG` parallelism for the horizontal loop and `VECTOR` parallelism for the vertical loop, rather than collapsing them. This choice reflects the additional nested loops (categories and ensemble members) that increase register pressure.

3. Pre-computed ensemble indices: The `ens_lookup_int_idx` array provides direct $\mathcal{O}(1)$ access to the tracer index for each ensemble member, replacing the original $\mathcal{O}(N_{\mathrm{ens}} \times N_{\mathrm{tracer}})$ string matching algorithm.

4. Conditional ensemble participation: Not all base tracers participate in ensemble simulations. The `is_ensemble_tracer` flag, pre-computed during initialization, identifies which tracers require ensemble processing, allowing the kernel to skip non-ensemble tracers efficiently.

Listing 4. Ensemble tracer emission kernel (Section 4, Phase 3).

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

```
! Ensemble tracer kernel - apply lambda scaling for inversion ensembles
IF (ens_tracer > 0) THEN
  max_nens = SIZE(lambda_mat, dim=4)

  !$ACC PARALLEL DEFAULT(PRESENT) PRIVATE(nt, nc, nc_ensemble, nr, lambda, &
  !$ACC &        temp_scaling_fact, temp_tracer, ens_int_idx, ens_count)
  !$ACC LOOP GANG
  DO jc = is, ie
    nr = reg_map(jc,jb)  ! Region index for lambda matrix lookup

    !$ACC LOOP VECTOR
    DO k = 1, nlev
      !$ACC LOOP SEQ
      DO icat = 1, n_active_cats
        nt = active_nt(icat)

        IF (is_ensemble_tracer(nt)) THEN
          nc = active_nc(icat)
          nc_ensemble = nc_ensemble_map(icat)
          ens_count = ens_count_map(nt)
          temp_scaling_fact = n1*tsf_now(jc,nt,nc,jb) + n2*tsf_next(jc,nt,nc,jb)

          !$ACC LOOP SEQ
          DO nens = 1, max_nens
            ens_int_idx = ens_lookup_int_idx(nens, nt)

            IF (ens_int_idx > 0) THEN
              ! Retrieve region- and category-specific lambda scaling
              lambda = lambda_mat(ens_count, nc_ensemble, nr, nens)

              ! Compute scaled emission tendency
              temp_tracer = total_weight(jc,k,nt,nc,jb) * temp_scaling_fact * lambda &
                &          / airmass_now(jc,k,jb)

              ! Accumulate to ensemble tracer
              p_tracer_now(jc,k,jb,ens_int_idx) = p_tracer_now(jc,k,jb,ens_int_idx) &
                &                               + temp_tracer
            END IF
          END DO ! nens
        END IF ! is_ensemble_tracer
      END DO ! icat
    END DO ! k
  END DO ! jc
  !$ACC END PARALLEL
END IF
```

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

4.4.5 Memory Access Optimization

Both kernels are designed to maximize memory bandwidth utilization on GPU architectures. The critical optimization is the loop ordering that places the horizontal index j_c in the position where consecutive threads access consecutive memory addresses. For ICON's array layout with the horizontal dimension first (`array(nproma, nlev, nblk, ntracer)`), this ensures coalesced memory transactions. Table 3 summarizes the memory access patterns for the key arrays in Section 4.

Table 3. Memory access patterns in the main emission kernel. Coalesced access occurs when the stride equals 1 (consecutive threads access consecutive elements).

Array	Dimensions	Access stride	Pattern
`p_tracer_now`	(j_c, k, j_b, τ)	1	Coalesced
`total_weight`	(j_c, k, n_t, n_c, j_b)	1	Coalesced
`airmass_now`	(j_c, k, j_b)	1	Coalesced
`tsf_now`, `tsf_next`	(j_c, n_t, n_c, j_b)	1	Coalesced
`lambda_mat`	$(\mathrm{cnt}, n_c, r, n_{\mathrm{ens}})$	$N_r \cdot N_c$	Strided
`active_nt`, `active_nc`	(i_{cat})	Broadcast	Uniform

The `lambda_mat` array exhibits strided access because the region index r varies with grid cell while the array is indexed with region as the third dimension. However, this array is relatively small (typically $<$1 MB), and the strided access is mitigated by GPU caching. The lookup table arrays (`active_nt`, etc.) are accessed uniformly across all threads in a warp, enabling broadcast optimization where a single memory transaction serves all threads.

The reorganized loop hierarchy of Eq. (4) and the array layouts of Table 3 are not GPU-only refactors: they apply unchanged to the CPU code path, which compiles from the same source with the `!$ACC` directives ignored. Placing j_c as the fastest-varying index improves cache locality on CPU as well, since consecutive iterations of the inner loop access contiguous memory and benefit from hardware prefetchers and SIMD load coalescing on the ARM-based Grace CPU. We did not observe a CPU performance regression from the reordering; if anything, the elimination of per-cell string comparisons in the inner loop (Sect. 4.1.1) gives a modest CPU speedup that is included in the baseline reported in Sect. 6.2, although we have not isolated this contribution from the other refactors.

5 Implementation of GPU-Optimized VPRM

The new VPRM code computes biospheric fluxes based on the original formulation of Mahadevan et al. (2008), with vegetation class-specific parameters taken from Glauch et al. (2025). Several modifications and extended capabilities were implemented to support ensemble-based simulations. The present GPU optimization addresses several challenges specific to VPRM: conditional branching based on flux type (respiration vs. gross primary production), vegetation-class-dependent parameter selection,

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

and the need to support both regular and ensemble tracer configurations. This section details the mathematical framework, algorithmic structure, and OpenACC implementation of the GPU-optimized VPRM.

5.1 Mathematical Framework

As mentioned earlier, the net ecosystem exchange is computed as the difference between ecosystem Respiration (R) and gross primary production, where positive NEE indicates net CO_2 release to the atmosphere and negative NEE indicates net uptake by the biosphere. Both components are computed as weighted sums over vegetation classes present in each grid cell.

5.1.1 Respiration Flux

The respiration flux follows a linear temperature dependence for each vegetation class:

$$\mathrm{R} = \sum_{\ell=1}^{N_{\mathrm{veg}}} (\alpha_\ell \cdot T + \beta_\ell) \cdot f_\ell \cdot M_{\mathrm{CO_2}} \cdot 10^{-9}, \tag{19}$$

where α_ℓ and β_ℓ are vegetation-class-specific respiration parameters (with units that yield μmol m^{-2} s^{-1}), T is the 2-meter air temperature in degrees Celsius computed by the model, f_ℓ is the fractional coverage of vegetation class ℓ in the grid cell, and $M_{\mathrm{CO_2}} = 44.01$ g mol^{-1} is the molar mass of CO_2. The factor 10^{-9} converts from μmol to kmol for consistency with ICON's tracer units.

The vegetation-class-specific parameters (α_ℓ and β_ℓ) are either provided in the run script or taken as default values and are stored in lookup tables initialized on the CPU. The fractional vegetation coverage (f_ℓ) is derived from a mapping between 22 land-cover classes (GLOBCOVER 2009 by default) and 8 VPRM vegetation classes (Mahadevan et al., 2008), and is also initialized on the CPU. Both parameter tables and fractional coverages are then transferred to GPU memory for use in the online respiration flux computations.

5.1.2 Gross Primary Production

The GPP represents photosynthetic CO_2 uptake and incorporates multiple environmental scaling factors:

$$\mathrm{GPP} = \sum_{\ell=1}^{N_{\mathrm{veg}}} \lambda_\ell \cdot T_{\mathrm{scale},\ell} \cdot P_{\mathrm{scale},\ell} \cdot W_{\mathrm{scale},\ell} \cdot \mathrm{EVI}_\ell \cdot \frac{\mathrm{PAR}}{1 + \mathrm{PAR}/\mathrm{PAR}_{0,\ell}} \cdot f_\ell \cdot M_{\mathrm{CO_2}} \cdot 10^{-9}, \tag{20}$$

where λ_ℓ is the light-use efficiency parameter, PAR is Photosynthetically Active Radiation (W m^{-2}), $\mathrm{PAR}_{0,\ell}$ is the half-saturation PAR value, and EVI_ℓ is the Enhanced Vegetation Index for class ℓ. The scaling factors T_{scale}, P_{scale}, and W_{scale} modulate photosynthesis based on temperature, phenology, and water availability, respectively.

The satellite-derived indices required for GPP computations, namely the EVI and the Land Surface Water Index (LSWI, used for the water-scaling factor W_{scale}), are taken from MODIS observations. The photosynthetically active radiation (PAR) at the surface is computed from the model-provided shortwave radiation fluxes as:

$$\mathrm{PAR} = \frac{\mathrm{SW\_net\_sfc} + \mathrm{SW\_up\_sfc}}{0.505}, \tag{21}$$

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

where SW_net_sfc is the net shortwave radiation flux at the surface and SW_up_sfc is the upward flux at the surface, their sum yields the total downward shortwave flux available for vegetation photosynthesis.

5.1.3 Environmental Scaling Factors

The temperature scaling function captures the reduction in photosynthetic efficiency at extreme temperatures:

$$T_{\text{scale}} = \frac{(T - T_{\min})(T - T_{\max})}{(T - T_{\min})(T - T_{\max}) - (T - T_{\text{opt}})^2 + \epsilon}, \tag{22}$$

where $T_{\min}$, $T_{\max}$, and T_{opt} are vegetation-class-specific minimum, maximum, and optimal temperatures for photosynthesis, and $\epsilon = 10^{-12}$ prevents division by zero. This function yields values between 0 and 1, with maximum scaling at $T = T_{\text{opt}}$.

The water stress scaling depends on the vegetation class characteristics:

$$W_{\text{scale}} = \begin{cases} 1 & \text{if class is xeric (drought-adapted)} \\ \dfrac{1 + \text{LSWI}}{1 + \text{LSWI}_{\max}} & \text{if class is mesic} \end{cases} \tag{23}$$

where LSWI is the Land Surface Water Index and $\text{LSWI}_{\max}$ is the maximum LSWI observed during the growing season.

The phenology scaling captures seasonal vegetation development:

$$P_{\text{scale}} = \begin{cases} 1 & \text{if class is evergreen} \\ \dfrac{1 + \text{LSWI}}{2} & \text{if class is deciduous/cropland} \end{cases} \tag{24}$$

5.2 GPU Optimization Strategy

The VPRM GPU optimization employs strategies analogous to those used for OEM, adapted to the specific computational structure of biospheric flux calculations:

1. Elimination of string comparisons: The original implementation used string-based flux type selection (`vprm_flux_type == 'resp'` or `'gpp'`). These are replaced at the initialization stage with integer flags and separate kernel pathways.
2. Pre-computed vegetation parameters: Class-specific parameters (α, β, λ, PAR_0, $T_{\min}$, $T_{\max}$, T_{opt}) are organized into contiguous arrays indexed by vegetation class number.
3. Two-stage computation: Fluxes are first computed per vegetation class into a temporary array, then reduced across classes when updating tracers. This separation enables efficient vectorization.
4. Ensemble lookup tables: For ensemble simulations, pre-computed tables map ensemble member indices to tracer indices, analogous to the OEM ensemble tables.

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

As for OEM, all four refactors apply to the unified CPU/GPU source: the integer flags, parameter arrays, two-stage temporary buffers, and ensemble lookup tables are allocated and populated identically on both architectures, with the `!$ACC` directives ignored when compiling for CPU. The two-stage decomposition and the per-class flux array layout improve cache locality on the CPU as well, and the elimination of in-loop string comparisons removes a non-trivial source of CPU overhead. We did not introduce any architecture-specific code paths within VPRM and have not observed CPU performance degradation from the refactor.

5.3 Algorithmic Structure

Algorithm 3 presents the complete pseudocode for the GPU-optimized VPRM. The computation proceeds through initialization (executed once), followed by the main flux computation kernel (executed every timestep). The main kernel is further divided into respiration and GPP stages, each with regular and ensemble tracer variants.

Algorithm 3 GPU-Optimized VPRM Algorithm

Require: Meteorological inputs (T, PAR), vegetation indices (EVI, LSWI), class fractions f_ℓ

Ensure: Updated CO_2 tracer mixing ratios

// Initialization (once at first timestep)
if first timestep **then**
  Allocate temporary flux arrays: $F_{\text{resp}}(j_c, \ell)$, $F_{\text{gpp}}(j_c, \ell)$ for $\ell = 1, \ldots, N_{\text{veg}}$
  Pre-compute vegetation parameter arrays: α_ℓ, β_ℓ, λ_ℓ, $\text{PAR}_{0,\ell}$, etc.
  Build ensemble lookup tables: `vprm_ens_lookup_int_idx`, `vprm_ens_lookup_cat_idx`
  Identify lambda category indices: `lambda_cat_resp`(ℓ), `lambda_cat_gpp`(ℓ)
  `!$ACC ENTER DATA COPYIN(all parameter arrays, lookup tables)`
end if

// Main computation (every timestep)

// Stage 1a: Respiration flux per vegetation class (GPU parallel)
for $j_c = i_s$ **to** i_e **in parallel (GANG) do**
  $T \leftarrow T_{2\text{m}}(j_c)$ {2-meter temperature}
  for $\ell = 1$ **to** N_{veg} **in parallel (VECTOR) do**
    if $f_\ell(j_c) > 0$ **then**
      $F_{\text{resp}}(j_c, \ell) \leftarrow (\alpha_\ell \cdot T + \beta_\ell) \cdot f_\ell(j_c) \cdot M_{CO_2} \cdot 10^{-9}$
    else
      $F_{\text{resp}}(j_c, \ell) \leftarrow 0$
    end if
  end for

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

22: **end for**
23:
24: **// Stage 1b: GPP flux per vegetation class (GPU parallel)**
25: **for** $j_c = i_s$ **to** i_e **in parallel (GANG) do**
26: $T \leftarrow T_{2\mathrm{m}}(j_c)$; $\quad \mathrm{PAR} \leftarrow \mathrm{PAR}(j_c)$
27: **for** $\ell = 1$ **to** N_{veg} **in parallel (VECTOR) do**
28: **if** $f_\ell(j_c) > 0$ **and** $\mathrm{PAR} > 0$ **then**
29: Compute T_{scale}, W_{scale}, P_{scale} using Eqs. (22)–(24)
30: $F_{\mathrm{gpp}}(j_c, \ell) \leftarrow \lambda_\ell \cdot T_{\mathrm{scale}} \cdot P_{\mathrm{scale}} \cdot W_{\mathrm{scale}} \cdot \mathrm{EVI}_\ell \cdot \frac{\mathrm{PAR}}{1+\mathrm{PAR}/\mathrm{PAR}_{0,\ell}} \cdot f_\ell \cdot M_{\mathrm{CO_2}} \cdot 10^{-9}$
31: **else**
32: $F_{\mathrm{gpp}}(j_c, \ell) \leftarrow 0$
33: **end if**
34: **end for**
35: **end for**
36:
37: **// Stage 2a: Reduce fluxes and update regular tracer at surface level**
38: **for** $j_c = i_s$ **to** i_e **in parallel (GANG VECTOR) do**
39: $F_{\mathrm{total}} \leftarrow \sum_{\ell=1}^{N_{\mathrm{veg}}} F(j_c, \ell)$ {F is either F_{resp} or F_{gpp} depending on tracer}
40: $\chi(j_c, k_{\mathrm{sfc}}, \tau) \mathrel{+}= \Delta t \cdot F_{\mathrm{total}} / \rho_{\mathrm{air}}(j_c, k_{\mathrm{sfc}})$ {Update at lowest model level only}
41: **end for**
42:
43: **// Stage 2b: Update ensemble tracers (if enabled)**
44: **if** $N_{\mathrm{ens}} > 0$ **and** tracer is ensemble participant **then**
45: **for** $j_c = i_s$ **to** i_e **in parallel (GANG) do**
46: $r \leftarrow \texttt{reg\_map}(j_c)$ {Region index}
47: **for** $n_{\mathrm{ens}} = 1$ **to** N_{ens} **in parallel (VECTOR) do**
48: $\tau_{\mathrm{ens}} \leftarrow \texttt{vprm\_ens\_lookup\_int\_idx}(n_{\mathrm{ens}})$
49: **if** $\tau_{\mathrm{ens}} > 0$ **then**
50: $F_{\mathrm{ens}} \leftarrow \sum_\ell F(j_c, \ell) \cdot \lambda_{\mathrm{mat}}(\texttt{ens\_count}, n_{c,\ell}, r, n_{\mathrm{ens}})$ {$n_{c,\ell}$: lambda category for resp or gpp}
51: $\chi(j_c, k_{\mathrm{sfc}}, \tau_{\mathrm{ens}}) \mathrel{+}= \pm \Delta t \cdot F_{\mathrm{ens}} / \rho_{\mathrm{air}}(j_c, k_{\mathrm{sfc}})$ {Sign depends on flux type}
52: **end if**
53: **end for**
54: **end for**
55: **end if**

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

5.4 GPU Kernel Structure

As mentioned earlier, the new VPRM implementation uses a two-stage computation approach that separates the per-vegetation-class flux calculation from the reduction and tracer update. This design enables efficient parallelization while handling the varying number of vegetation classes present in each grid cell.

5.4.1 Stage 1: Per-Class Flux Computation

The first stage computes fluxes for each vegetation class independently. Listing 5 shows the respiration calculation, which demonstrates the key optimization patterns.

Listing 5. VPRM Stage 1: Per-vegetation-class respiration flux computation.

```
! Stage 1a: Compute respiration flux per vegetation class
!$ACC PARALLEL DEFAULT(PRESENT) PRIVATE(t2m, vclass_frac, resp_flux)
!$ACC LOOP GANG
DO jc = is, ie
  ! Get 2-meter temperature for this cell
  t2m = p_diag%temp_2m(jc,jb)

  !$ACC LOOP VECTOR
  DO l = 1, n_vprm_classes
    vclass_frac = vprm_data%vclass_frac(jc,l,jb)

    IF (vclass_frac > 0.0_wp) THEN
      ! Respiration: R = (alpha * T + beta) * frac * M_CO2 * 1e-9
      resp_flux = (vprm_par%resp_alpha(l) * t2m + vprm_par%resp_beta(l)) &
        &       * vclass_frac * mol_weight_co2 * 1.0e-9_wp

      ! Ensure non-negative respiration (numerical safety)
      temp_flux_resp(jc, l) = MAX(resp_flux, 0.0_wp)
    ELSE
      temp_flux_resp(jc, l) = 0.0_wp
    END IF
  END DO ! l (vegetation classes)
END DO ! jc
!$ACC END PARALLEL
```

The parallelization strategy employs `GANG` parallelism for the horizontal loop and `VECTOR` parallelism for the vegetation class loop. This mapping is effective because:

- The horizontal dimension provides sufficient work for gang-level parallelism.
- The vegetation class dimension ($N_{\text{veg}} = 8$) matches well with the vector width of modern GPUs (32 threads per warp on NVIDIA architectures).
- Each vegetation class computation is independent, with no data dependencies that would require synchronization.

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

5.4.2 GPP Computation with Environmental Scaling

The GPP calculation is more complex than respiration due to the multiple scaling factors. Listing 6 shows the core GPP computation with temperature, water, and phenology scaling.

Listing 6. VPRM GPP computation with environmental scaling factors.

```
! Stage 1b: GPP computation with environmental scaling
!$ACC PARALLEL DEFAULT(PRESENT) PRIVATE(t2m, par_val, evi_val, lswi_val, &
!$ACC &        lswi_max, t_scale, w_scale, p_scale, gpp_flux, denom)
!$ACC LOOP GANG
DO jc = is, ie
  t2m = p_diag%temp_2m(jc,jb)
  par_val = MAX(p_diag%swflx_par_sfc(jc,jb), 0.0_wp)  ! PAR >= 0

  !$ACC LOOP VECTOR
  DO l = 1, n_vprm_classes
    vclass_frac = vprm_data%vclass_frac(jc,l,jb)

    IF (vclass_frac > 0.0_wp .AND. par_val > 0.0_wp) THEN
      ! Retrieve vegetation indices for this class
      evi_val = vprm_data%evi(jc,l,jb)
      lswi_val = vprm_data%lswi(jc,l,jb)
      lswi_max = vprm_data%lswi_max(jc,l,jb)

      ! Temperature scaling (Eq. 4)
      denom = (t2m - vprm_par%tmin(l)) * (t2m - vprm_par%tmax(l)) &
        &   - (t2m - vprm_par%topt(l))**2 + 1.0e-12_wp
      t_scale = (t2m - vprm_par%tmin(l)) * (t2m - vprm_par%tmax(l)) / denom
      t_scale = MAX(0.0_wp, MIN(1.0_wp, t_scale))  ! Clamp to [0,1]

      ! Water stress scaling (class-dependent)
      IF (vprm_par%is_xeric(l)) THEN
        w_scale = 1.0_wp  ! Xeric vegetation: no water stress
      ELSE
        w_scale = (1.0_wp + lswi_val) / (1.0_wp + lswi_max + 1.0e-12_wp)
        w_scale = MAX(0.0_wp, MIN(1.0_wp, w_scale))
      END IF

      ! Phenology scaling (class-dependent)
      IF (vprm_par%is_evergreen(l)) THEN
        p_scale = 1.0_wp  ! Evergreen: constant phenology
      ELSE
        p_scale = (1.0_wp + lswi_val) / 2.0_wp
        p_scale = MAX(0.0_wp, MIN(1.0_wp, p_scale))
      END IF

      ! GPP flux with light saturation curve
      gpp_flux = vprm_par%lambda(l) * t_scale * p_scale * w_scale * evi_val &
        &      * par_val / (1.0_wp + par_val / vprm_par%par0(l)) &
        &      * vclass_frac * mol_weight_co2 * 1.0e-9_wp

```

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026


```
      ! GPP is positive (uptake); will be subtracted when computing NEE
      temp_flux_gpp(jc, l) = MAX(gpp_flux, 0.0_wp)
    ELSE
      temp_flux_gpp(jc, l) = 0.0_wp
    END IF
  END DO ! l
END DO ! jc
!$ACC END PARALLEL
```

5.4.3 Stage 2: Reduction and Tracer Update

The second stage reduces the per-class fluxes and updates the tracer concentrations. Crucially, the VPRM implementation processes respiration and GPP as *separate* tracers in the outer tracer loop: each tracer is associated with a flux type (`'resp'` or `'gpp'`), and the per-class fluxes computed in Stage 1 are summed and applied independently to the corresponding tracer. This design enables the inversion framework to optimize respiration and GPP fluxes separately. Since VPRM computes two-dimensional surface fluxes, the tracer update is applied directly at the lowest model level ($k = \texttt{nlev}$) rather than through a vertical distribution profile. Listing 7 shows this computation.

Listing 7. VPRM Stage 2: Flux reduction and regular tracer update.

```
! Stage 2: Reduce vegetation class fluxes and update tracer at surface
! (executed for each VPRM tracer: either respiration or GPP)
!$ACC PARALLEL DEFAULT(PRESENT) PRIVATE(newflux)
!$ACC LOOP GANG VECTOR PRIVATE(newflux)
DO jc = is, ie
  newflux = 0.0_wp

  !$ACC LOOP SEQ
  DO k = 1, n_vprm_classes
    newflux = newflux + temp_flux(jc, k)
  END DO
  newflux = MAX(newflux, 0.0_wp)

  IF (is_diag_output) THEN
    ! Diagnostic output: store per-class fluxes
    !$ACC LOOP SEQ
    DO k = 1, n_vprm_classes
      p_tracer_now(jc,k,jb,trcr_idx) = temp_flux(jc, k)
    END DO
  ELSE
    ! Regular tracer: accumulate total flux at lowest model level
    p_tracer_now(jc,nlev,jb,trcr_idx) = p_tracer_now(jc,nlev,jb,trcr_idx) + &
      &                                 dtime * newflux / airmass_now(jc,nlev,jb)
  END IF
END DO ! jc
!$ACC END PARALLEL
```

The reduction over vegetation classes is performed sequentially (`LOOP SEQ`) because the small number of classes (8) does not justify the overhead of a parallel reduction. The horizontal loop uses collapsed `GANG VECTOR` parallelism. Note that the

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

Net Ecosystem Exchange (NEE = R − GPP) is not computed within VPRM itself; instead, respiration and GPP are propagated as independent tracers, and any combination into NEE is performed downstream, for example by the inversion framework.

5.4.4 Ensemble Tracer Kernel

For ensemble simulations, the new VPRM code applies vegetation-class-specific lambda scaling factors to the base fluxes. Unlike OEM where lambda varies by emission category, VPRM lambda values are indexed by vegetation class for both respiration and GPP. Listing 8 shows the ensemble kernel structure.

Listing 8. VPRM ensemble tracer kernel with vegetation-class-specific lambda scaling.

```
! Stage 2b: Ensemble tracer update for VPRM fluxes
! (executed within the outer tracer loop for each ensemble-participating tracer)
IF (ens_tracer > 0 .AND. vprm_is_ensemble(nt)) THEN
  max_nens = SIZE(lambda_mat, dim=4)
  ens_count = vprm_ens_count_map(nt)

  !$ACC PARALLEL DEFAULT(PRESENT) PRIVATE(nr, new_ens_flux, lambda, nc, ens_int_idx)
  !$ACC LOOP GANG
  DO jc = is, ie
    nr = reg_map(jc,jb)  ! Region index for lambda lookup

    !$ACC LOOP VECTOR PRIVATE(new_ens_flux, lambda, nc, ens_int_idx)
    DO nens = 1, max_nens
      ens_int_idx = vprm_ens_lookup_int_idx(nens, nt)

      IF (ens_int_idx > 0) THEN
        new_ens_flux = 0.0_wp

        !$ACC LOOP SEQ
        DO l = 1, n_vprm_classes
          ! Select lambda category based on current tracer's flux type
          IF (vprm_flux_type(nt) == 'resp') THEN
            nc = lambda_cat_resp(l)
          ELSE
            nc = lambda_cat_gpp(l)
          END IF
          lambda = lambda_mat(ens_count, nc, nr, nens)
          new_ens_flux = new_ens_flux + temp_flux(jc, l) * lambda
        END DO ! l

        ! Update ensemble tracer at surface level
        ! Sign: subtract for background GPP, add otherwise
        p_tracer_now(jc,nlev,jb,ens_int_idx) = p_tracer_now(jc,nlev,jb,ens_int_idx) &
          & +/- dtime * new_ens_flux / airmass_now(jc,nlev,jb)
      END IF
    END DO ! nens
  END DO ! jc
  !$ACC END PARALLEL
END IF
```

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

The ensemble kernel differs from the regular VPRM kernel in several important aspects:

1. Separate lambda categories: Respiration and GPP may belong to different lambda categories in the inversion framework, allowing independent scaling of the two flux components. The pre-computed arrays `lambda_cat_resp` and `lambda_cat_gpp` map vegetation classes to their respective lambda category indices, and the appropriate set is selected based on the current tracer's flux type.

2. Vectorized ensemble loop: The ensemble member loop is parallelized with `VECTOR`, while the vegetation class reduction loop is marked as `SEQ` because the lambda matrix access pattern varies with vegetation class.

3. Sign-dependent update: The sign of the flux contribution depends on whether the current ensemble member corresponds to a background GPP subtraction (negative sign) or a direct ensemble flux (positive sign). This is determined by pre-computed flags (`vprm_ens_lookup_bg_gpp`).

4. Surface-level update: As with the regular kernel, ensemble tracer updates are applied at the lowest model level ($k =$ `nlev`) only.

5.5 Memory Management

The VPRM GPU implementation maintains several categories of data in device memory:

- Persistent parameters (transferred once at initialization): vegetation class parameters (α, β, λ, PAR_0, $T_{\min}$, $T_{\max}$, T_{opt}), class characteristic flags (`is_xeric`, `is_evergreen`), and ensemble lookup tables.
- Time-varying vegetation data (updated at vegetation index refresh interval): EVI, LSWI, and $\mathrm{LSWI}_{\max}$. These are typically updated weekly to monthly based on satellite observations.
- Timestep data: meteorological inputs ($T_{2\mathrm{m}}$, PAR), air mass, and tracer concentrations. These are already maintained on GPU by ICON's infrastructure.
- Temporary arrays: per-class flux arrays (`temp_flux_resp`, `temp_flux_gpp`) allocated within the kernel scope using `!$ACC DATA CREATE`.

5.6 Computational Complexity

The computational cost of VPRM scales as:

$$C_{\mathrm{VPRM}} = N_{\mathrm{cells}} \cdot N_{\mathrm{veg}} \cdot \underbrace{(C_{\mathrm{resp}} + C_{\mathrm{GPP}})}_{\text{Stage 1}} + N_{\mathrm{cells}} \cdot N_{\mathrm{lev}} \cdot \underbrace{(1 + N_{\mathrm{ens}} \cdot N_{\mathrm{veg}})}_{\text{Stage 2}}, \tag{25}$$

where C_{resp} and C_{GPP} are the per-class operation counts for respiration ($\sim$5 FLOPs) and GPP ($\sim$25 FLOPs including all scaling factors). For the benchmark configuration ($N_{\mathrm{cells}} \approx 3 \times 10^5$, $N_{\mathrm{veg}} = 8$, $N_{\mathrm{lev}} = 60$, $N_{\mathrm{ens}} = 45$), the ensemble reduction

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

in Stage 2 dominates the cost, contributing approximately $10^5 \times 60 \times 45 \times 8 \approx 6.5 \times 10^9$ operations per timestep. Table 4 summarizes the memory access patterns for the key VPRM arrays. The vegetation parameter arrays (`vprm_par%*`) benefit from broadcast optimization because all threads in a warp access the same vegetation class parameter simultaneously when processing different grid cells. The lambda matrix exhibits strided access similar to OEM, but the small size of this array (typically $<$100 KB) ensures it remains cache-resident.

Table 4. Memory access patterns in the VPRM GPU kernels.

Array	Dimensions	Access stride	Pattern
`temp_2m`, `swflx_par_sfc`	(j_c, j_b)	1	Coalesced
`vclass_frac`, `evi`, `lswi`	(j_c, ℓ, j_b)	1	Coalesced
`temp_flux_resp`, `temp_flux_gpp`	(j_c, ℓ)	1	Coalesced
`p_tracer_now`	(j_c, k, j_b, τ)	1	Coalesced
`vprm_par%*`	(ℓ)	Broadcast	Uniform
`lambda_mat`	(n_c, r, n_{ens})	N_r	Strided

6 Performance Evaluation

6.1 Benchmark Configuration

The benchmark simulations employ a European domain configuration (Hamzehloo, 2026) to model CO_2 across several countries on the continent, with key parameters listed in Table 5. The setup mimics a European CO_2 inversion at very high spatial resolution with a grid spacing of 3.3 km. A simulation of CO_2 was chosen because it requires both OEM and VPRM. The domain is sufficiently large to exploit the computational capabilities of the GH200 superchips on the ALPS HPC system, while still being able to fit on a single GPU node within memory constraints. A relatively small ensemble size of 45 was chosen; larger sizes would only be possible on multiple nodes.

Table 5. Benchmark simulation configuration parameters.

Parameter	Value
Horizontal grid spacing	3.3 km
Number of horizontal grid cells	∼306,000
Vertical levels	60
Ensemble members	45
Number of emission tracers	1
Number of emission categories	2

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

For GPU execution, the domain decomposition employs a single block per MPI rank with dynamically computed `nproma` values, maximizing GPU occupancy. For CPU execution, a fixed `nproma` = 8 is used with correspondingly more blocks; this value is the default in the current ICON configuration on which our setup is built (Lapillonne et al., 2026). The performance assessment was conducted on the ALPS supercomputer at the Swiss National Supercomputing Center (CSCS). ALPS is a hybrid system whose compute nodes each contain four NVIDIA Grace–Hopper (GH200) superchips; each superchip pairs one 72-core ARM-based Grace CPU with one Hopper GPU connected via NVLink-C2C. A single node thus provides four GPUs and four Grace CPUs. The CPU baseline uses all four Grace CPUs on one node (278 MPI ranks total) from the same hardware generation as the GPU tests, ensuring a fair comparison between CPU and GPU execution on identical nodes. This platform represents the national transition toward GPU-dominated HPC in Switzerland, making it the primary target architecture for ICON-ART developments. Comprehensive benchmarks of the broader ICON GPU port on this architecture are presented by Lapillonne et al. (2026). All experiments use identical ICON-ART source code, compiled with the NVHPC compiler (version 25.1) via Spack (Gamblin et al., 2015). The same compiler flags and precision settings were applied for both GPU and CPU runs to ensure numerical and algorithmic comparability.

6.2 Scaling Results

Table 6 presents the scaling performance of the GPU-optimized implementation compared to the CPU baseline. Several key observations emerge from the performance evaluation. On a single node the GPU implementation achieves a wall-clock reduction from 2763 s (CPU) to 1438 s (GPU), a node-level speedup of 1.9×. Scaling to multiple GPU nodes—which increases total hardware resources—further reduces the wall-clock time to 475 s on five nodes (a factor of 5.8× relative to the single CPU node). We note that multi-node comparisons represent strong-scaling behavior rather than like-for-like speedup, since additional nodes contribute proportionally more compute resources.

The single-node speedup of 1.9× is modest relative to the raw memory bandwidth ratio of the GH200 (approximately 4 TB s^{-1} HBM3 vs. ∼0.5 TB s^{-1} LPDDR5X). This reflects two factors: first, the total simulation time is dominated by components beyond OEM and VPRM—notably the dynamical core, diffusion, and I/O—which are already GPU-enabled but constrained by communication and synchronization overhead; second, the large number of ensemble tracers (45 members, each requiring full 3D fields) places substantial pressure on device memory, limiting the problem size that can fit on a single node and preventing more aggressive optimization strategies such as batched kernel launches or occupancy-tuned tiling.

The OEM and VPRM modules specifically show a reduction from 350 s (CPU) to 289 s (1 GPU node), representing a 1.21× module-level speedup. This more modest speedup for the emission modules compared to the overall simulation reflects several factors: (1) the emission modules represent only 12.7% of the total CPU runtime, limiting the impact of their optimization on overall performance according to Amdahl's law; (2) the ensemble computations, while optimized, require significant arithmetic intensity with lambda matrix lookups that are not fully compute-bound; and (3) the temporal scaling computations, though pre-computed hourly, still contribute to per-timestep overhead. Importantly, however, the GPU implementation prevents these modules from becoming a bottleneck as other ICON components are accelerated. The Streaming Multiprocessor (SM) utilization of 58–69% indicates reasonable GPU occupancy, with headroom that could be exploited by future kernel-level tun-

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

ing. The decrease in utilization with increasing node count reflects the reduced per-GPU workload as the domain is distributed across more devices. This is expected behavior for strong scaling scenarios where the problem size remains fixed.

Table 6. Performance scaling results comparing CPU and GPU execution. The CPU baseline uses 1 node with 278 MPI ranks. GPU results show scaling from 1 to 5 nodes with 4 MPI ranks per node (one per GPU socket).

Config	Nodes	MPIs	OEM+VPRM [s]	Total [s]	Fraction [%]	Speedup (Total)	SM Util. [%]	Energy [Wh]
CPU	1	278	350	2763	12.7	1.0×	—	360.9
GPU	1	4	289	1438	20.1	1.9×	69	555.9
GPU	2	8	140	842	16.6	3.3×	65	615.1
GPU	3	12	96	640	15.0	4.3×	61	645.1
GPU	4	16	72	528	13.6	5.2×	60	677.7
GPU	5	20	59	475	12.4	5.8×	58	695.8

Figure 2 presents a comprehensive analysis of the performance and energy characteristics of the GPU-optimized implementation. The four panels examine complementary aspects of computational efficiency that are critical for assessing the practical utility of GPU acceleration in production atmospheric modeling.

6.2.1 Strong Scaling Behavior

Panel (a) of Fig. 2 displays the strong scaling of wall-clock time as a function of GPU node count. The total simulation time decreases from 1438 s on a single GPU node to 475 s on five nodes, while the OEM+VPRM component decreases from 289 s to 59 s over the same range. Compared to the CPU baseline (2763 s total, 350 s for OEM+VPRM), these results demonstrate substantial acceleration across all configurations.

The dashed lines indicate ideal linear scaling from the single GPU node baseline. A notable observation is that the OEM+VPRM modules (red) track their ideal scaling line more closely than the total simulation (blue). This behavior reflects the highly data-parallel nature of the emission computations, which exhibit minimal inter-process communication and near-perfect load balance across the domain decomposition. In contrast, the total simulation includes communication-intensive components such as halo exchanges for the dynamical core and implicit solvers, which introduce parallel overhead that degrades scaling efficiency at higher node counts.

6.2.2 Speedup and Parallel Efficiency

Panel (b) quantifies the speedup relative to the CPU baseline and the parallel efficiency of the GPU implementation. The total simulation achieves speedups of 1.92× to 5.82× on 1 to 5 GPU nodes, respectively. Remarkably, the OEM+VPRM modules achieve speedups of 1.21× to 5.93×, with the 5-node speedup slightly exceeding the theoretical ideal of ∼5.9× (calculated

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

as $350\,\mathrm{s}/59\,\mathrm{s}$). This near-linear behavior for OEM+VPRM at high node counts likely reflects improved cache utilization as the per-GPU workload decreases.

The parallel efficiency (green triangles, right axis), defined as the ratio of actual speedup to ideal speedup normalized to the single-node GPU performance, decreases from 100% at 1 node to 60.5% at 5 nodes for the total simulation. This efficiency loss is characteristic of strong scaling in atmospheric models and arises from three principal factors: (1) communication overhead for halo exchanges that scales with the surface-to-volume ratio of the domain decomposition; (2) load imbalance at domain boundaries where some MPI ranks have fewer active grid cells; and (3) fixed overhead costs (initialization, I/O, diagnostics) that do not scale with processor count. The 60% efficiency at 5 nodes remains acceptable for production use and is comparable to scaling results reported for the full ICON model (Lapillonne et al., 2026). It is worth noting that further improvements in ICON GPU performance, with a focus on exascale computing, are being developed through the EXCLAIM project using the GT4Py domain-specific language, as recently reported by Dipankar et al. (2026).

6.2.3 Energy Consumption

Panel (c) examines the energy consumption characteristics of CPU and GPU execution. The GPU implementation consumes 555.9–695.8 Wh per simulation depending on node count, compared to 360.9 Wh for the CPU baseline. This represents a 54–93% increase in absolute energy consumption, with the percentage increasing at higher node counts due to the additional GPU hardware being powered.

At first glance, this energy increase might appear problematic for sustainability considerations. Although GPUs offer superior FLOP-per-watt ratios for compute-bound workloads, the total node power draw includes memory subsystems, interconnects, and idle components that do not scale with arithmetic throughput; moreover, communication-bound phases of the simulation underutilize the GPU while the full node remains powered. As a result, absolute energy consumption alone is an incomplete metric for assessing computational efficiency, particularly for time-critical applications such as operational weather forecasting or near-real-time emission monitoring. The relevant question is not simply how much energy is consumed, but how effectively that energy is converted into scientific output within time constraints. Panel (d) addresses this question through the energy–delay product (EDP), originally introduced as a balanced energy–performance metric in the microprocessor design community (Gonzalez and Horowitz, 1996) and defined as the product of energy consumption and execution time,

$$\mathrm{EDP} = E \times T, \tag{26}$$

where E is the energy in watt-hours and T is the wall-clock time in seconds. The EDP metric, widely used in computer architecture and HPC performance analysis, captures the trade-off between energy efficiency and time-to-solution. Lower EDP values indicate superior overall computational efficiency. The CPU baseline achieves an EDP of 997.2 kWh·s, while the GPU implementation achieves 799.4 kWh·s on a single node, that is a 20% improvement despite the higher absolute energy consumption. This improvement grows substantially with additional nodes: 2 nodes achieve 517.9 kWh·s (48% improvement), and 5 nodes achieve 330.5 kWh·s (67% improvement over CPU).

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

The monotonic decrease in GPU EDP with increasing node count demonstrates that the performance gains from parallelization consistently outweigh the energy cost of additional hardware. This result has important implications for operational centers where both time-to-solution and energy costs factor into resource allocation decisions. For ensemble-based inversion systems that must process large numbers of tracers within fixed time windows (e.g., to assimilate satellite observations before the next overpass), the GPU implementation offers a favorable balance of speed and energy efficiency.

6.2.4 Implications for Operational Use

The performance characteristics documented in Fig. 2 have several practical implications for ICON-ART deployment:

1. Time-critical applications: For operational forecasting or real-time monitoring where wall-clock time is the binding constraint, GPU acceleration provides substantial benefits. A 5-node GPU configuration completes simulations in 17% of a single-node CPU time while achieving 67% lower EDP.

2. Throughput-oriented workloads: For research applications where many independent simulations must be completed (e.g., sensitivity studies, parameter sweeps), the choice between 1 GPU node (1.92× speedup, 20% EDP reduction) and 5 GPU nodes (5.82× speedup, 67% EDP reduction) depends on queue wait times and allocation policies.

3. Ensemble simulations: The near-ideal scaling of OEM+VPRM (5.93× on 5 nodes) indicates that emission computations will not become a bottleneck as ensemble sizes increase, supporting the use of large tracer ensembles (100–500 members) for atmospheric inversion applications.

The SM utilization of 58–69% (Table 6) indicates that additional performance gains may be achievable through kernel optimization, though the current implementation already provides production-ready performance for the target applications. The scaling efficiency (ratio of actual to ideal speedup) decreases from 100% at 1 node to approximately 60% at 5 nodes. This sub-linear scaling is characteristic of strong scaling in atmospheric models and arises from communication overhead for halo exchanges, load imbalance at domain boundaries, and fixed overhead costs that do not scale with the number of processors.

7 Code Validation

7.1 Probabilistic Testing

To ensure the scientific integrity of the GPU port, we adopted the probabilistic testing framework (`probtest`) originally developed for the ICON model (Lapillonne et al., 2026). Probabilistic testing provides a robust, statistically grounded alternative to bitwise comparison, which is generally infeasible for large, chaotic Earth system models executed on heterogeneous architectures. The approach recognizes that minor numerical deviations are inevitable due to differences in floating-point arithmetic, instruction ordering, and hardware-specific compiler optimizations.

In the probabilistic test, a CPU ensemble is constructed by introducing controlled perturbations of magnitude comparable to rounding errors (typically at the 10^{-14}–10^{-12} level) in selected input and state variables. The ensemble spread derived from

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

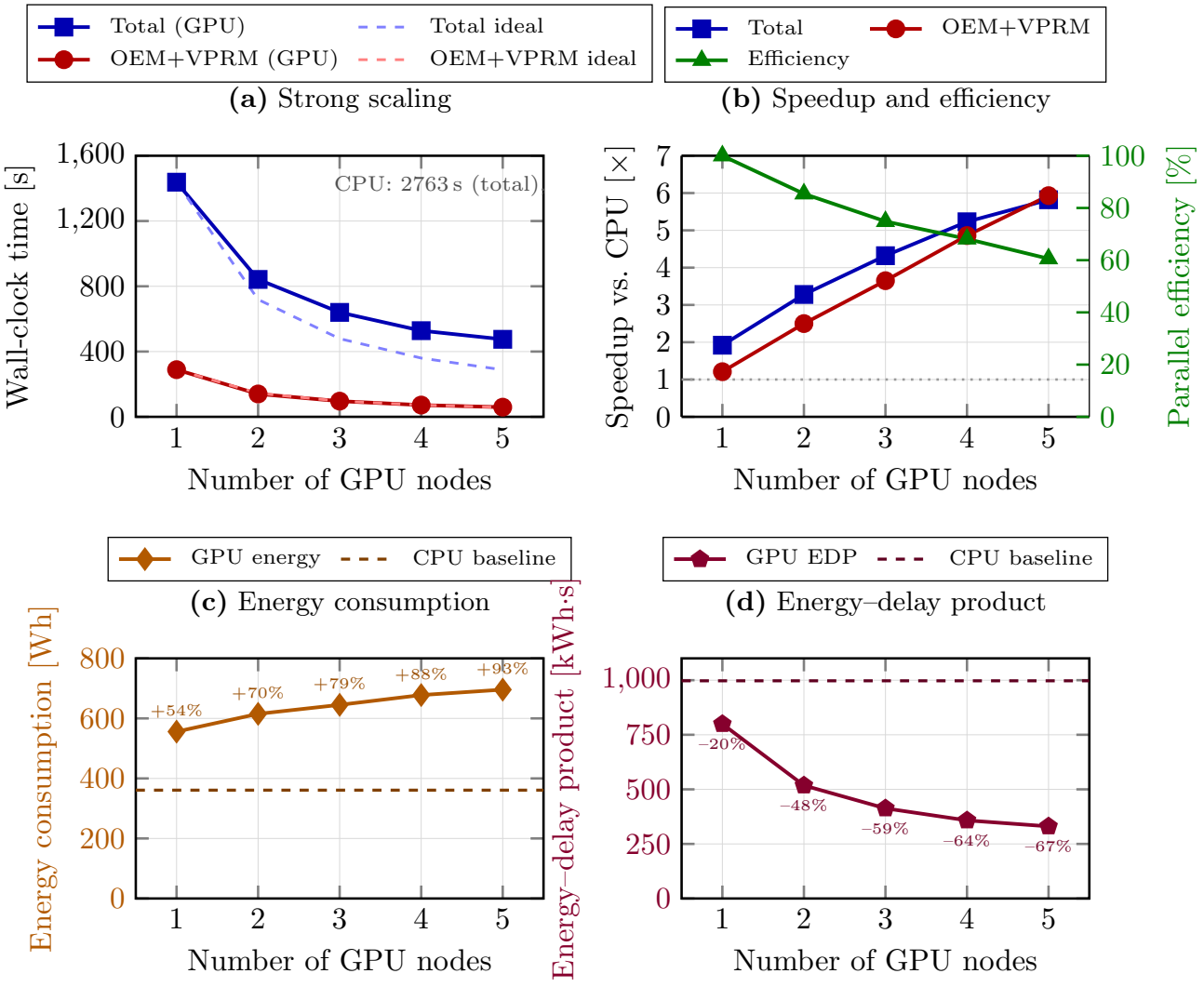

Figure 2. Performance and energy characteristics of the GPU-optimized ICON-ART on ALPS (GH200). (a) Strong scaling of wall-clock time for total simulation (blue) and OEM+VPRM modules (red); dashed lines indicate ideal linear scaling from the single GPU node baseline. The CPU reference (1 node, 278 MPI ranks) requires 2763 s total and 350 s for OEM+VPRM. (b) Speedup relative to the CPU baseline (left axis) and parallel efficiency (right axis, green triangles). The OEM+VPRM modules achieve a speedup of 5.93× on 5 nodes relative to the single node GPU baseline (289 s / 5 = 57.8 s expected vs. 59 s measured), demonstrating near-ideal strong scaling. The total simulation shows 60.5% parallel efficiency at 5 nodes due to communication overhead and load imbalance. (c) Energy consumption per simulation showing 54–93% higher absolute energy on GPUs compared to the CPU baseline (dashed line). Percentages indicate increase relative to CPU. (d) Energy–delay product (EDP = energy × time), a combined metric balancing energy efficiency and time-to-solution. Despite higher absolute energy consumption, GPU execution achieves 20–67% lower EDP than CPU, demonstrating superior computational efficiency for time-critical applications. The continuous EDP improvement with additional nodes indicates that the performance gains outweigh the energy cost.

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

these perturbed runs defines a tolerance envelope for each prognostic variable. The GPU simulation is then compared against this ensemble, and its results are deemed statistically consistent if all examined fields lie within the CPU-ensemble variance bounds. The `probtest` suite was applied using reduced ICON-ART configurations. All GPU runs passed the probabilistic test, demonstrating that numerical perturbations introduced by OpenACC parallelism and Fused Multiply–Add (FMA) optimizations remain within the intrinsic stochastic variability of the CPU reference ensemble.

7.2 Spatial and Temporal Validation

Following probabilistic consistency checks, the refactored GPU implementation is evaluated against the CPU baseline using ICON-ART simulations. For these experiments, emission fluxes from OEM and VPRM are computed simultaneously on CPU and GPU using identical meteorological drivers and tracer configurations. The results are compared across spatial and temporal scales spanning from hourly grid-cell values to daily averages aggregated over the European benchmark domain of Table 5.

Figure 3 presents a representative comparison of simulated CO_2 ensemble tracer concentrations between CPU and GPU executions for the European domain at 12:00 UTC on 1 January 2018 (12 hours into the simulation). The upper panels display the absolute CO_2 mixing ratio (vertically averaged over the lowest 10 model levels) for the CPU baseline (1 node), single GPU node, and 3 GPU nodes (an intermediate configuration chosen to illustrate scaling behavior while remaining within routine allocation constraints for validation runs). The spatial patterns, including elevated concentrations over major emission hotspots in central Europe and the Po Valley, as well as the biospheric signal variations across vegetated regions, are visually indistinguishable across all three configurations.

The lower panels of Fig. 3 quantify the relative differences between configurations. Remarkably, all pixel-wise differences remain within $\pm 0.01\%$, manifesting as uniform gray shading across the entire domain. This magnitude corresponds to the numerical noise floor arising from differences in floating-point operation ordering, fused multiply–add instructions, and parallel reduction sequences between CPU and GPU architectures. Critically, no systematic spatial patterns or regional biases are evident in the difference fields, confirming that the GPU implementation does not introduce spurious gradients or localized numerical artifacts.

The absence of structure in the difference fields is particularly significant for ensemble-based inversion applications, where even small systematic biases could propagate through the Ensemble Square Root Filter to corrupt posterior flux estimates. The validation results demonstrate that the GPU-accelerated OEM and VPRM modules can be used interchangeably with their CPU counterparts without affecting the scientific conclusions of downstream analyses.

Despite the near-identical comparisons shown in Fig. 3, strict bitwise reproducibility between GPU and CPU executions is not expected and the differences may grow with time due to the chaotic nature of the atmosphere. Infinitesimal perturbations in initial conditions or rounding sequences amplify exponentially through nonlinear feedback. Even mathematically equivalent computations can diverge after a sufficient number of integration steps due to the accumulation of floating-point noise.

As discussed for the full ICON model by Lapillonne et al. (2026), variations between CPU and GPU results primarily reflect small-scale rounding differences introduced by distinct compiler optimizations, fused multiply–add operations, and the order of parallel reductions. These differences project onto the high-frequency part of the model's attractor but do not alter its

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

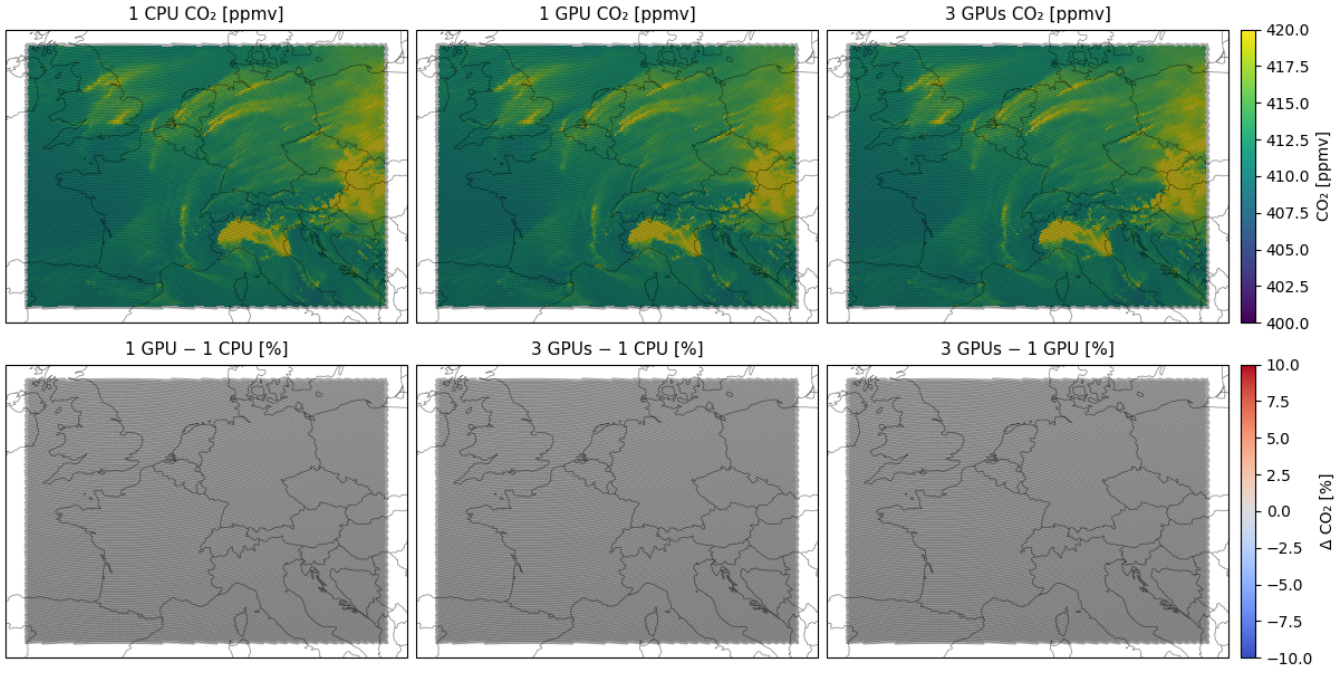

Figure 3. Comparison of ensemble CO_2 tracer concentrations (mean over lowest 10 vertical levels, in ppmv) across Europe between CPU and GPU simulations after 12 hours of integration (1 January 2018, 12:00 UTC). Upper panels: Absolute mixing ratios for (left) CPU 1-node baseline, (center) GPU 1-node, and (right) GPU 3-nodes configurations. The spatial distributions are visually identical, capturing emission hotspots in industrial regions and biospheric flux patterns. Lower panels: Relative differences (%) between configurations. All differences are at or below $\pm 0.01\%$, corresponding to the numerical noise floor arising from floating-point arithmetic variations. The uniform gray shading indicates that no systematic spatial biases exist between CPU and GPU implementations, confirming that the refactored modules preserve the scientific integrity of the original algorithms.

statistical invariants. What matters scientifically is not bit-identical trajectories but the preservation of ensemble statistics and physical balances (mass, energy, and tracer budgets). Numerical equivalence is demonstrated when GPU and CPU solutions are statistically indistinguishable within the envelope of the system's intrinsic predictability. Accordingly, the validation strategy focuses on consistency in mean fields, flux budgets, and variability statistics rather than deterministic reproducibility.

8 Discussion

The refactored and GPU-enabled implementations of OEM and VPRM represent a significant advancement in the computational capabilities of ICON-ART for atmospheric composition modeling. This section discusses the implications of these developments for scientific applications, the broader context of GPU acceleration in Earth system modeling, and directions for future work.

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

8.1 Demonstrated Scientific Applications

The GPU-enabled modules have already been successfully deployed in recent research applications, demonstrating their scientific maturity and operational readiness for high-resolution greenhouse gas and air-quality modeling. Ponomarev et al. (2026) employed the GPU-enabled ICON-ART framework for atmospheric inverse modeling of CO_2 fluxes in the metropolitan regions of Zurich and Paris. Their study used the model in combination with the CarbonTracker Data Assimilation System (CTDAS) to perform ensemble-based inversions at kilometer-scale resolution. The simulations exploited the full functionality of the OEM and VPRM modules, including online anthropogenic emissions, biospheric fluxes, and ensemble tracer handling with up to 300 ensemble members. The results demonstrated excellent consistency between modeled and observed atmospheric CO_2 concentrations and confirmed the system's capability to quantify regional emission patterns with unprecedented spatial granularity. This work represented a milestone in applying ICON-ART to urban-scale greenhouse-gas inversion and provided the first comprehensive evaluation of the ART emission modules in an operational ensemble framework on GPUs.

In complementary work, Thanwerdas et al. (2025) improved the Ensemble Square-Root Filter (EnSRF) within the Community Inversion Framework (CIF) using ICON-ART 2024.01 as the transport model. Importantly, their experiments were executed on CPUs and relied on the refactored and optimized OEM (not the GPU port) to propagate large tracer ensembles efficiently and robustly. The study focused on assimilation methodology and ensemble behavior, demonstrating stable convergence and posterior uncertainty reduction across synthetic experiments. While the computational setup was CPU-based, the adoption of the refactored OEM (featuring cleaner interfaces, explicit temporal and vertical modulation, and some pre-computed lookup structures) was instrumental for scalability and reproducibility, and it provided a clear pathway for subsequent GPU enablement within the same algorithmic structure.

8.2 Implications for High-Resolution Atmospheric Inversion

The GPU enablement of OEM and VPRM fundamentally expands the feasible resolution and complexity envelope for atmospheric inversion experiments. By removing major computational bottlenecks in the surface-flux chain, it becomes possible to perform kilometer-scale and sub-kilometer-scale simulations within practical wall-clock times. Such spatial resolutions are essential for assimilating and interpreting observations from current high-resolution satellite instruments, notably TROPOMI with its $\sim 5.5 \times 3.5\,\mathrm{km}^2$ footprint, and upcoming missions such as CO2M, which will provide even finer-scale CO_2 column measurements.

When combined with ensemble data assimilation, the accelerated emission modules enable quantification of fine-scale emission heterogeneity and facilitate identification of sector-specific emission sources. Regional variations in anthropogenic emissions from traffic, industry, and residential heating can be resolved at scales relevant to urban planning and emission verification. Similarly, biospheric flux variations driven by land-use heterogeneity, phenological gradients, and local meteorological conditions can be captured with sufficient spatial detail to improve process understanding and reduce systematic errors in inverse flux estimates. These capabilities provide policymakers with scientifically robust, evidence-based insights into emission processes and their temporal evolution, strengthening the link between Earth system modeling and climate mitigation policy.

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

8.3 Modularity and Reusability

Although the GPU-enabled OEM and VPRM codes are embedded within the ART component of ICON, they were designed with modularity in mind. Each module exposes a well-defined FORTRAN interface, separating grid indexing and tracer-array layout from the model's internal physics. This modular structure facilitates potential reuse of the emission and biospheric flux components in alternative modeling frameworks with minimal adaptation effort. The open-source nature of ICON-ART further promotes such reuse, encouraging integration of these GPU-ready modules into other atmospheric modeling systems.

8.4 Performance Tuning Considerations

The present GPU version of OEM and VPRM has been optimized for atmospheric inverse modeling experiments with tracers from large ensembles, where $N_{\text{tracer}} \times N_{\text{cat}}$ is small. For air quality simulations, where $N_{\text{tracer}} \times N_{\text{cat}}$ is typically large (50–100 categories with many emitted species), the current implementation may not be optimal. Further optimizing OEM for such applications will be challenging due to the many different use cases of ICON-ART. Specifically, due to the directive-based nature of OpenACC, the optimal balance between `gang` and `vector` parallelism, as well as the placement of `collapse(2)` clauses and loop fusion strategies, can depend on problem size, tracer count, and GPU architecture. Users are encouraged to conduct brief performance tests when applying these modules to different computational regimes (e.g., species-rich chemistry simulations with few ensembles, or very high-resolution domains with correspondingly larger horizontal loop counts). The flexibility of OpenACC enables such case-specific tuning without altering scientific outcomes, ensuring long-term maintainability and portability across evolving GPU generations.

8.5 Roadmap for Extended GPU Acceleration

The current GPU port provides complete device-resident execution for the emission and biospheric flux calculations within ICON-ART. Ongoing development efforts aim to extend GPU acceleration to additional ART components, particularly the aerosol microphysics and gas-phase chemistry modules, which currently represent significant fractions of the total ART computational cost. Achieving a fully GPU-enabled atmospheric composition framework will permit end-to-end simulations of emissions, transport, chemistry, and deposition on accelerator-based supercomputers without host–device synchronization within the physics integration loop. When combined with ensemble data-assimilation frameworks such as CTDAS and CIF, this comprehensive GPU implementation will support efficient, high-resolution air-quality and greenhouse-gas monitoring from global to urban scales. The present achievement thus represents an essential step toward scientifically robust, high-performance, and sustainable atmospheric modeling for the coming era of accelerator-dominated HPC.

9 Conclusions

This work presents the directive-based GPU porting, systematic refactoring, optimization, and functional extension of the Online Emission Module (OEM) and the Vegetation Photosynthesis and Respiration Model (VPRM) within the ICON-ART

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

atmospheric modeling framework. The implementation preserves the scientific formulations of the original algorithms while exposing their inherent horizontal and vertical parallelism for execution on modern GPU architectures using OpenACC compiler directives. The resulting single-source codebase maintains full portability between CPU and GPU platforms without code duplication or architecture-specific branches.

The principal technical contributions of this work are threefold. First, we eliminated GPU-incompatible string-based index lookups by constructing pre-computed integer-indexed lookup tables during initialization. This transformation reduces the computational complexity of tracer and category resolution from $\mathcal{O}(N_{\text{tracers}} \times N_{\text{categories}} \times N_{\text{timesteps}})$ string comparisons to $\mathcal{O}(N_{\text{timesteps}})$ array accesses per grid cell, fundamentally altering the algorithmic profile of the emission modules. Second, we implemented a temporal decomposition strategy that partitions computations by update frequency: one-time initialization and periodic boundary updates execute on the CPU to leverage existing I/O infrastructure, while hourly temporal scaling and per-timestep emission kernels execute entirely on the GPU. This hybrid architecture ensures that the computationally dominant operations occur on the accelerator without host–device communication overhead. Third, we reorganized loop hierarchies and memory layouts to maximize coalesced access patterns across ICON's horizontal grid dimension, with pre-computed ensemble lookup tables enabling efficient processing of large tracer ensembles without runtime string comparisons. Additionally, the functionality of both OEM and VPRM was extended to support ensemble-based atmospheric inverse modeling, which typically requires several hundred ensemble tracers.

Direct spatial comparisons of CO_2 concentration fields demonstrated pixel-wise agreement at the numerical noise floor ($<0.01\%$), with no systematic spatial biases between CPU and GPU configurations, and between single and multiple nodes. These results establish that the accelerated modules can be used interchangeably with their CPU counterparts without affecting the scientific integrity of downstream analyses, including ensemble-based atmospheric inversions where even small systematic errors could propagate significantly.

Performance benchmarks on the ALPS supercomputer, equipped with NVIDIA Grace–Hopper (GH200) superchips, demonstrated substantial computational gains. The GPU implementation achieves overall simulation speedups of $\sim$1.9$\times$ on a single node to $\sim$5.8$\times$ on five nodes relative to a 278-rank CPU baseline. Notably, the OEM and VPRM modules exhibit near-ideal strong scaling, achieving $\sim$5.9$\times$ speedup on five nodes. While absolute energy consumption increases by 54–93% compared to the CPU baseline, the energy–delay product (a metric capturing the trade-off between energy efficiency and time-to-solution) improves by 20–67%, demonstrating that the performance gains consistently outweigh the energy costs for time-critical applications.

The computational advances reported here directly enable new scientific capabilities for atmospheric composition modeling. Ensemble-based inversions with hundreds of tracers can now be executed at kilometer-scale resolution within practical wall-clock times, supporting assimilation of high-resolution satellite observations from, for example, TROPOMI and the forthcoming CO2M mission. The GPU-enabled modules have already been deployed in recent ICON-ART applications for regional CO_2 flux inversion over European metropolitan areas and for methodological development of ensemble square-root filter algorithms. The modular design of the refactored code facilitates potential reuse in alternative modeling frameworks, while

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

the directive-based implementation ensures adaptability to future GPU architectures through recompilation rather than code rewriting.

Code and data availability.

The ICON-ART modeling framework (Release 2026.04), including the GPU-ported OEM and VPRM codes described in this manuscript, is open source, distributed under the BSD-3-Clause license through the ICON Partnership, and archived on Zenodo (ICON Partnership, 2026, https://doi.org/10.5281/zenodo.20762965). The scripts and input data needed to reproduce the simulations, scaling measurements, and figures reported here are archived on Zenodo under a CC-BY-4.0 licence (Hamzehloo, 2026, https://doi.org/10.5281/zenodo.21390631).

Author contributions.

AH developed, optimized, refactored, and ported the OEM and VPRM codes to GPUs; performed and post-processed the computational simulations on HPC clusters; led the study and drafted the original manuscript; and edited the revised versions. EFMK, NP, and MS contributed to extending the functionality of OEM and VPRM; prepared test case files; and provided valuable input through frequent discussions. SW, in his role as code maintainer of ART, organized the code review exercise and facilitated the successful merge of the GPU developments into the ICON-ART main repository. XL and WS supervised the GPU porting developments and provided valuable comments during regular meetings. MJ contributed to the development of the original CPU versions of OEM and VPRM codes. LE provided financial support, contributed to the strategic direction of the research, and reviewed and approved the manuscript. DB secured the main project funding and supervised the project; contributed to the conceptual development of the refactored OEM and VPRM codes; contributed to manuscript editing and revising; and provided valuable input through frequent discussions and meetings.

Competing interests.

The authors declare that they have no conflict of interest.

Acknowledgements. This work was funded by the Platform for Advanced Scientific Computing (PASC) through project HAM and ART Acceleration for Many-Core Architectures (HAMAM). The authors acknowledge the use of computational resources of the Swiss National Supercomputing Center (CSCS), on Piz Daint and ALPS platforms, under project IDs c27 and s1302. The authors would also like to thank all members of the Atmospheric Modeling and Remote Sensing group of the Air Pollution / Environmental Laboratory at Empa for their support and fruitful discussions, as well as the ICON-ART community through KIT and C2SM. We acknowledge the use of automated grammar and

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

punctuation tools provided by Overleaf. In addition, large language models were used in certain cases to improve the clarity of phrasing and to enhance the visual presentation of code listings and algorithms.

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

References

Berchet, A., Sollum, E., Thompson, R. L., Pison, I., Thanwerdas, J., Broquet, G., Chevallier, F., Aalto, T., Berchet, A., Bergamaschi, P., et al.: The Community Inversion Framework v1.0: a unified system for atmospheric inversion studies, Geoscientific Model Development, 14, 5331–5354, https://doi.org/10.5194/gmd-14-5331-2021, 2021.

Dipankar, A., Bianco, M., Bukenberger, M., Ehrengruber, T., Farabullini, N., Fuhrer, O., Gopal, A., Hupp, D., Jocksch, A., Kellerhals, S., et al.: Toward exascale climate modelling: a python DSL approach to ICON's (icosahedral non-hydrostatic) dynamical core (icon-exclaim v0.2.0), Geoscientific Model Development, 19, 713–729, https://doi.org/10.5194/gmd-19-713-2026, 2026.

Gamblin, T., LeGendre, M., Collette, M. R., Lee, G. L., Moody, A., De Supinski, B. R., and Futral, S.: The Spack package manager: bringing order to HPC software chaos, in: Proceedings of the international conference for high performance computing, networking, storage and analysis (SC '15), pp. 1–12, ACM, https://doi.org/10.1145/2807591.2807623, 2015.

Giorgetta, M. A., Sawyer, W., Lapillonne, X., Adamidis, P., Alexeev, D., Clément, V., Dietlicher, R., Engels, J. F., Esch, M., Franke, H., et al.: The ICON-A model for direct QBO simulations on GPUs (version icon-cscs: baf28a514), Geoscientific Model Development, 15, 6985–7016, https://doi.org/10.5194/gmd-15-6985-2022, 2022.

Glauch, T., Marshall, J., Gerbig, C., Botía, S., Gałkowski, M., Vardag, S. N., and Butz, A.: pyVPRM: a next-generation vegetation photosynthesis and respiration model for the post-MODIS era, Geoscientific Model Development, 18, 4713–4742, https://doi.org/10.5194/gmd-18-4713-2025, 2025.

Gonzalez, R. and Horowitz, M.: Energy dissipation in general purpose microprocessors, IEEE Journal of Solid-State Circuits, 31, 1277–1284, https://doi.org/10.1109/4.535411, 1996.

Hamzehloo, A.: Input data and reproduction scripts for GPU-accelerated anthropogenic and biosphere flux computations in ICON-ART (v2026.04), https://doi.org/10.5281/zenodo.21390631, dataset, 2026.

Herdman, J., Gaudin, W., McIntosh-Smith, S., Boulton, M., Beckingsale, D. A., Mallinson, A. C., and Jarvis, S. A.: Accelerating hydrocodes with OpenACC, OpenCL and CUDA, in: 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, pp. 465–471, IEEE, https://doi.org/10.1109/SC.Companion.2012.66, 2012.

Hoshyaripour, G. A., Baer, A., Bierbauer, S., Bruckert, J., Brunner, D., Förstner, J., Hamzehloo, A., Hanft, V., Keller, C., Klose, M., et al.: The atmospheric composition component of the ICON modeling framework: ICON-ART version 2025.10, Geoscientific Model Development, 19, 1645–1681, https://doi.org/10.5194/gmd-19-1645-2026, 2026.

ICON Partnership: ICON 2026.04 Release, https://doi.org/10.5281/zenodo.20762965, computer software. Contributing institutions: Max Planck Institute for Meteorology, German Meteorological Service, German Climate Computing Centre, Karlsruhe Institute of Technology, Center for Climate Systems Modeling, 2026.

Jähn, M., Kuhlmann, G., Mu, Q., Haussaire, J.-M., Ochsner, D., Osterried, K., Clément, V., and Brunner, D.: An online emission module for atmospheric chemistry transport models: implementation in COSMO-GHG v5.6a and COSMO-ART v5.1–3.1, Geoscientific Model Development, 13, 2379–2392, https://doi.org/10.5194/gmd-13-2379-2020, 2020.

Lapillonne, X., Hupp, D., Gessler, F., Walser, A., Pauling, A., Lauber, A., Cumming, B., Osuna, C., Müller, C., Merker, C., et al.: Operational numerical weather prediction with ICON on GPUs (version 2024.10), Geoscientific Model Development, 19, 755–772, https://doi.org/10.5194/gmd-19-755-2026, 2026.

https://doi.org/10.5194/egusphere-2026-2531
Preprint. Discussion started: 18 September 2026

Mahadevan, P., Wofsy, S., Matross, D., Xiao, X., Dunn, A. L., Lin, J. C., Gerbig, C., Munger, J. W., Chow, V. Y., and Gottlieb, E. W.: A satellite-based biosphere parameterization for net ecosystem CO_2 exchange: Vegetation Photosynthesis and Respiration Model (VPRM), Global Biogeochemical Cycles, 22, GB2005, https://doi.org/10.1029/2006GB002735, 2008.

Malenza, G., Cesare, V., Santimaria, M. E., Birke, R., Vecchiato, A., Becciani, U., and Aldinucci, M.: Performance portability via C++ PSTL, SYCL, OpenMP, and HIP: the Gaia AVU-GSR case study, in: SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1152–1163, IEEE, https://doi.org/10.1109/SCW63240.2024.00157, 2024.

Ponomarev, N., Steiner, M., Koene, E., Rubli, P., Grange, S., Constantin, L., Ramonet, M., David, L., Hamzehloo, A., Emmenegger, L., et al.: Estimation of CO2 fluxes in the cities of Zurich and Paris using the ICON-ART CTDAS inverse modelling framework, Atmospheric Chemistry and Physics, 26, 547–570, https://doi.org/10.5194/acp-26-547-2026, 2026.

Rieger, D., Bangert, M., Bischoff-Gauss, I., Förstner, J., Lundgren, K., Reinert, D., Schröter, J., Vogel, H., Zängl, G., Ruhnke, R., and Vogel, B.: ICON–ART 1.0 – a new online-coupled model system from the global to regional scale, Geoscientific Model Development, 8, 1659–1676, https://doi.org/10.5194/gmd-8-1659-2015, 2015.

Schröter, J., Rieger, D., Stassen, C., Vogel, H., Weimer, M., Werchner, S., Förstner, J., Prill, F., Reinert, D., Zängl, G., et al.: ICON-ART 2.1: A flexible tracer framework and its application for composition studies in numerical weather forecasting and climate simulations, Geoscientific Model Development, 11, 4043–4068, https://doi.org/10.5194/gmd-11-4043-2018, 2018.

Steiner, M., Cantarello, L., Henne, S., and Brunner, D.: Flow-dependent observation errors for greenhouse gas inversions in an ensemble Kalman smoother, Atmospheric Chemistry and Physics, 24, 12 447–12 463, https://doi.org/10.5194/acp-24-12447-2024, 2024.

Thanwerdas, J., Berchet, A., Constantin, L., Tsuruta, A., Steiner, M., Reum, F., Henne, S., and Brunner, D.: Improving the ensemble square root filter (EnSRF) in the Community Inversion Framework: a case study with ICON-ART 2024.01, Geoscientific Model Development, 18, 1505–1544, https://doi.org/10.5194/gmd-18-1505-2025, 2025.

Van Der Laan-Luijkx, I. T., Van Der Velde, I. R., Van Der Veen, E., Tsuruta, A., Stanislawska, K., Babenhauserheide, A., Zhang, H. F., Liu, Y., He, W., Chen, H., et al.: The CarbonTracker Data Assimilation Shell (CTDAS) v1.0: implementation and global carbon balance 2001–2015, Geoscientific Model Development, 10, 2785–2800, https://doi.org/10.5194/gmd-10-2785-2017, 2017.

Wolfe, M., Lee, S., Kim, J., Tian, X., Xu, R., Chapman, B., and Chandrasekaran, S.: The OpenACC data model: Preliminary study on its major challenges and implementations, Parallel Computing, 78, 15–27, https://doi.org/10.1016/j.parco.2018.07.003, 2018.

Zängl, G., Reinert, D., Rípodas, P., and Baldauf, M.: The ICON (ICOsahedral Nonhydrostatic) modelling framework of DWD and MPI-M: Description of the nonhydrostatic dynamical core, Quarterly Journal of the Royal Meteorological Society, 141, 563–579, https://doi.org/10.1002/qj.2378, 2015.

Zängl, G., Reinert, D., and Prill, F.: Grid refinement in ICON v2.6.4, Geoscientific Model Development, 15, 7153–7176, https://doi.org/10.5194/gmd-15-7153-2022, 2022.