
# Rewriting ICON for scalable development on emerging architectures

Claudia Frauen[1], Jörg Behrens[1], Sergey Kosukhin[2], Hendryk Bockelmann[1], Daniel Klocke[2] and the (pre)WarmWorld and ICON-C teams

ESM User Forum, March 16-17, 2022

[1] DKRZ, [2] MPI-M

# Overview of projects and initiatives

- **WarmWorld**: German national project initiative proposed to the BMBF hopefully starting in September 2022. Project partners: DKRZ, DWD, KIT, MPI-M, AWI, ECMWF, FZJ, JSC, Uni Köln, Uni Hamburg, Uni Leipzig
- **ICON-C**: Coordinated effort involving all ICON partners (C2SM, DKRZ, DWD, KIT, MPI-M) to rewrite ICON for scalable development on emerging architectures
- **preWarmWorld**: Preparatory project for WarmWorld; separately funded by BMBF; started in 2021; project partners: DKRZ, JSC, MPI-M

# ICON (ICOsahedral Nonhydrostatic model)

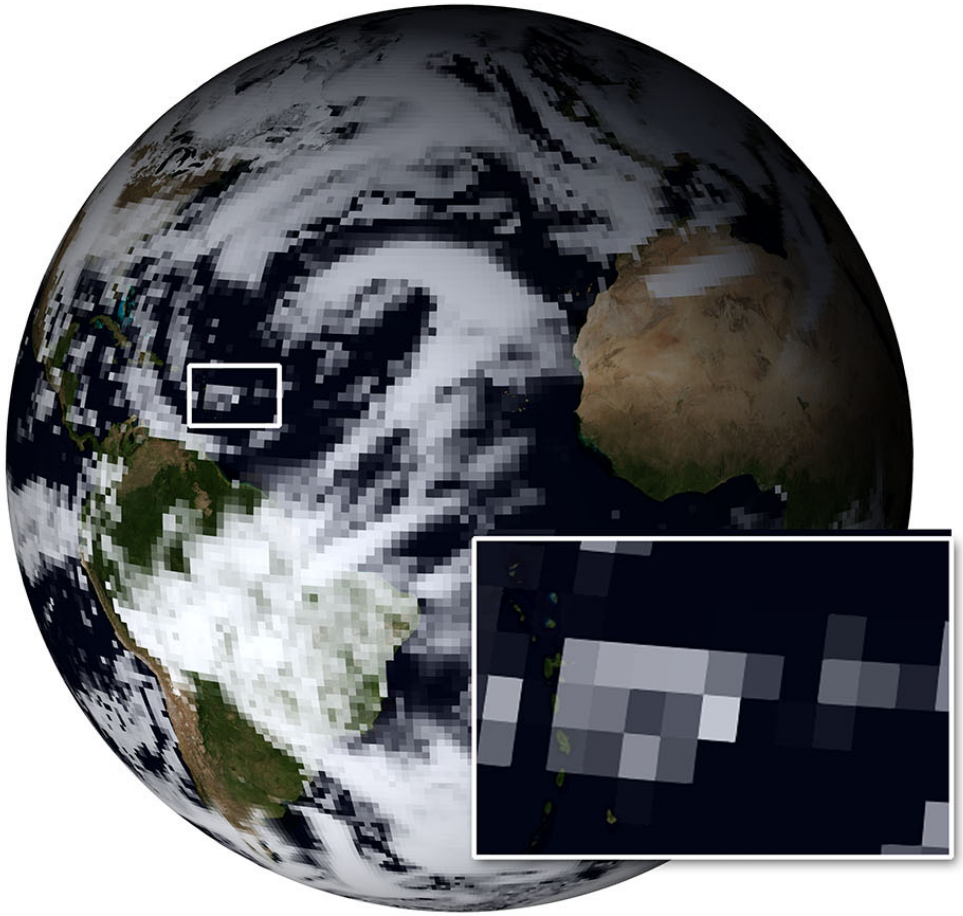
- ICON is a weather and climate model with atmosphere, ocean and land components
- Almost 2 decades development; initially by DWD and MPI-M, later also KIT and DKRZ and now also in collaboration with C2SM
- Mostly written in Fortran using MPI/OpenMP for parallelisation
- ~ 2 Million lines of code

# ICON at km-scales

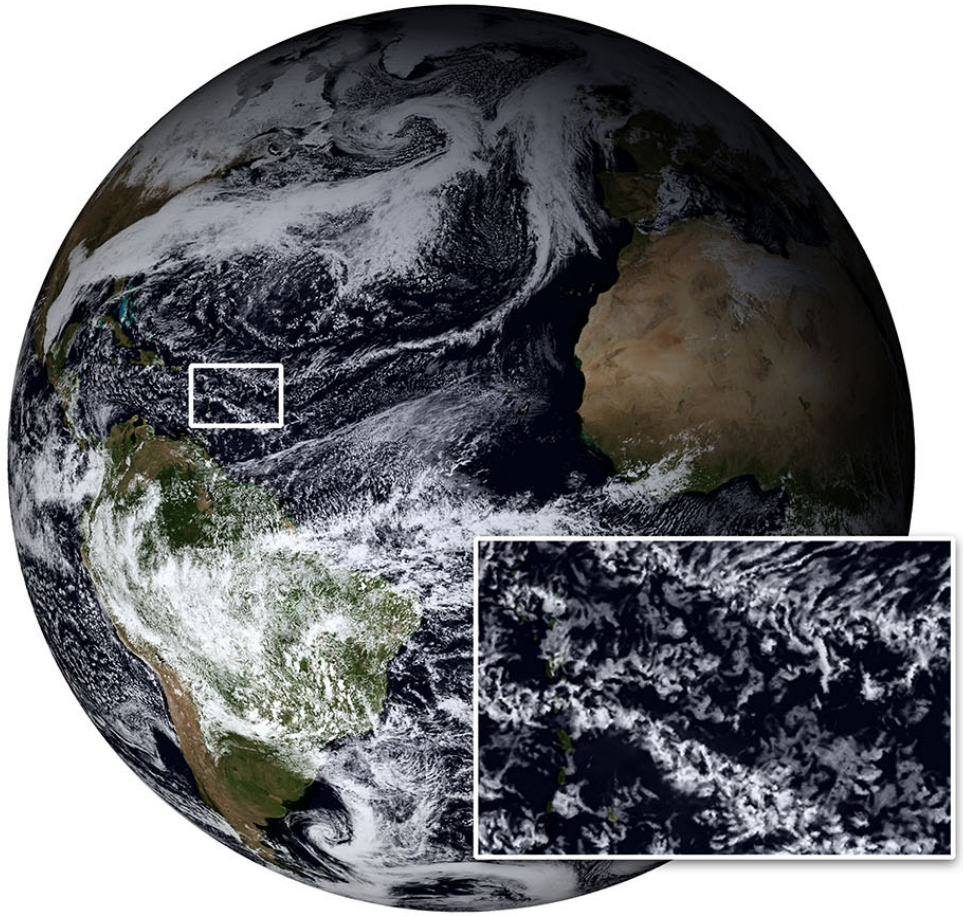
- HD(CP)$^2$ project enabled efficient km and hectometre (hm) scale applications over large regional domains
- These efforts enabled the use of ICON to perform the first global storm-resolving (SR) simulations in Europe
- ICON is one of only four models worldwide to have been run as an SR-ESM, i.e., coupled with km-scale resolution in the atmosphere and ocean

# Simulation of clouds

MPI-ESM HR, 80km



ICON R2B10, 2.5km



Florian Ziemen, DKRZ

# ICON on GPUs

- Huge efforts especially by CSCS, Meteo Swiss, Nvidia have lead to a GPU-enabled version of ICON-A (e.g. through the PASC ENIAC and IMPACT projects)
- Use of OpenACC directives + CLAW tool for the land model
- Successfully runs on Piz Daint and JUWELS Booster (QUBICC and Monsoon 2.0 projects)

# Current ICON performance

| $\Delta_h$ | $L_{atm}$ | $L_{ocn}$ | Nodes | Machine | SDPD |
|---|---|---|---|---|---|
| 5.0 km | 90 | n/a | 300 | Mistral (2xIntel BDW 36-cores) | 28 |
| 2.5 km | 90 | n/a | 510 | Mistral (2xIntel BDW 36-cores) | 7 |
| 2.5 km | n/a | 112 | 262 | Mistral (2xIntel BDW 36-cores) | 51 |
| 5.0 km | 90 | 128 | 420 | Mistral (2xIntel BDW 36-cores) | 25 |
| 5.0 km | 90 | n/a | 77 | JUWELS Booster (4xNVIDIA A100) | 120 |
| 2.5 km | 90 | n/a | 600 | Levante (2xAMD 7763 128-cores) | 20 |
| 5.0 km | n/a | 128 | 250 | Levante (2xAMD 7763 128-cores) | 375 |
| 5.0 km | 90 | 128 | 420 | Levante (2xAMD 7763 128-cores) | 96 |

Performance characteristics of ICON for different horizontal ($\Delta_h$) and vertical ($L_{atm}$ and $L_{ocn}$) resolutions.

# WarmWorld Goals

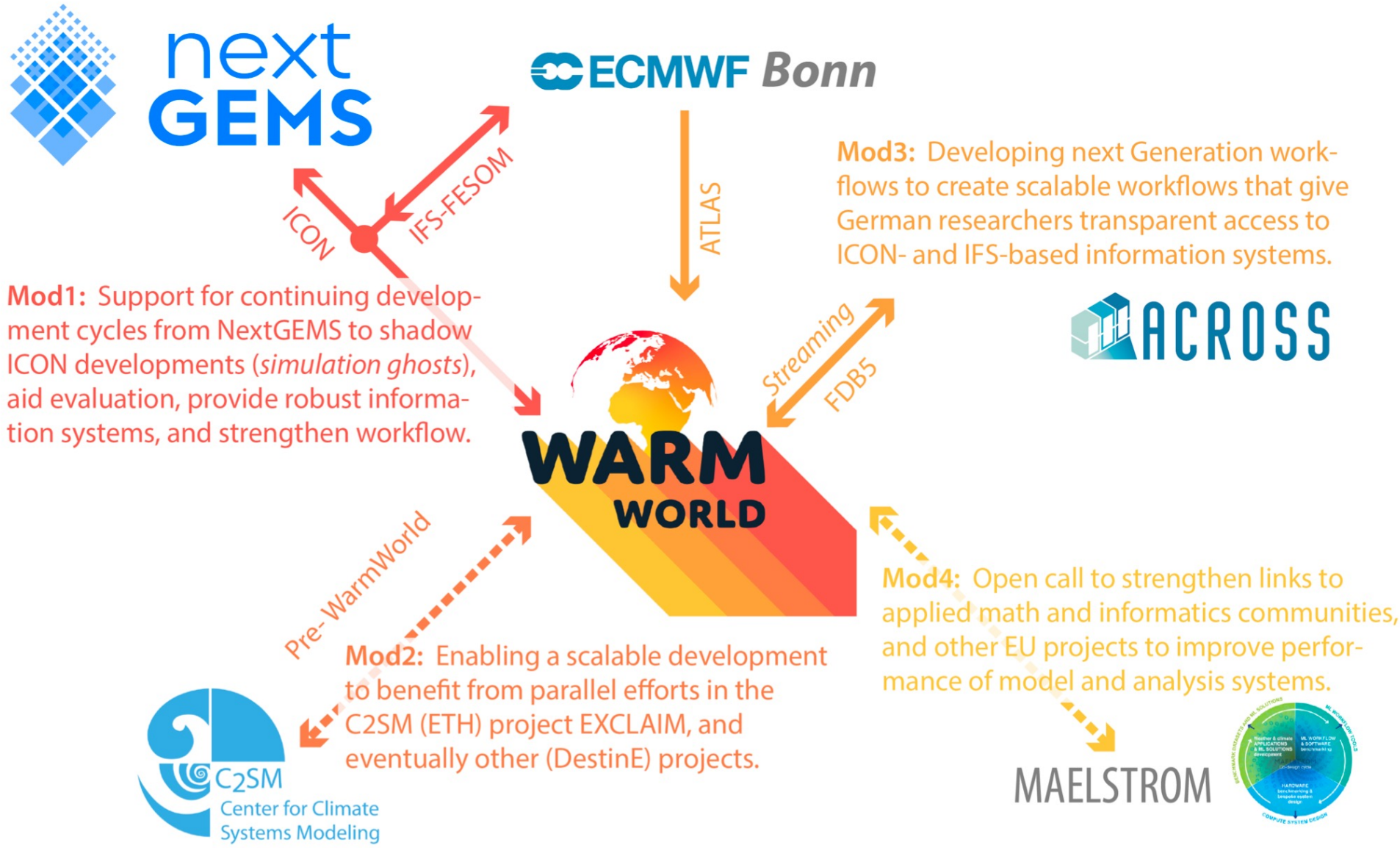
**Assess the detailed trajectory of global warming and the quantitative implications of this warming for human and natural systems**

- Coupled ICON running with an acceptable simulation quality on km scale > 0.5 SYPD by 2026
- **ICON-C**: A free and open source software implementation of the fully (land, ocean, atmosphere) coupled ICON to enable scalable development
- Integrated workflow to expose information of ICON alongside IFS-based solutions and observational data

# WarmWorld Modules

- **Better**: Responsible for defining and testing the model configurations
- **Faster**: Responsible for transforming the ICON code base into an open, scalable, modularized and flexible code
- **Easier**: Responsible for developing novel methods to make information visible, accessible, and interoperable
- **Smarter**: Aims to involve the applied maths and informatics communities, to improve the workflow and the model performance

# WarmWorld collaborations



# Faster objectives

- Transform the ICON code base into an open, scalable, modularized and flexible code named **ICON-C** (“ICON-consolidated”).
- Refactor ICON with the goal of scalable development to enable portable performance improvements – ultimately making ICON faster
- Initiate target performance ports to meet throughput (>0.5SYPD on a 2.5km or finer grid) goals
- Progressively redefine the ICON code structure to expose areas of performance improvement for targeted exploration of new programming concepts

# ICON redesign

- WarmWorld is just one piece of the puzzle towards ICON-C; larger coordinated effort involving all ICON partners
  - Balancing between different needs: Operational numerical weather forecast and cutting-edge climate modelling
- EXCLAIM (ETH Zürich): **Ex**treme scale **c**omputing and data p**l**atform for cloud-resolving we**a**ther and cl**i**mate **m**odelling
  - Approach: Re-write ICON code into a descriptive user code based on Python, which is then translated into standard imperative language (e.g. C++) for specific architectures using a toolchain based on GT4Py (GridTools for Python)

# ICON realities

- ICON is largely monolithic:
  - Huge code base is compiled in
  - Namelists are used to (de-)activate large tracts of code
  - Minimal unit testing
- Git submodules are used, but only few can be decoupled from compilation
- Components are not cleanly separated
- Uses complex derived types
- Contains unused code

# ICON-C first development steps

- Refining the Development Process
- Implementation of a disable functionality, initially via #ifdef, and clean-up
- Modularisation of components: Proposal and prototypes
- Prototype Data Management
- Infrastructure Measures
- Testing Hierarchy and Tools

# preWarmWorld

- Funded by the BMBF as a separate project to prepare for **WarmWorld** and to facilitate timely coordination with external projects such as EXCLAIM
- Provide a technical blueprint in terms of modularization and programming paradigms
- Overlap between the latter phase of **preWarmWorld** and the start of **WarmWorld** allows these plans to be coordinated before delivering the development environment (repository, test structure, licenses) for use in **WarmWorld**

# Planned assessment of programming paradigms in preWarmWorld

- Evaluation, comparison and prototypical implementation of selected modules using modern programming paradigms targeting heterogeneous hardware
- Implement granule using GridTools framework
- Implement granule using a *generic DSL*, like AnyDSL
- Implement granule using a *domain independent generic library*, like Kokkos and/or DPC++ / SYCL
- Implement granule using the concept of an *embedded DSL*
- Analyse the applicability and, if suitable, implement the interfacing of the above concepts to the front-end developed in the ESCAPE2 project

# C++ Data / Memory Management – Why?

- Need to move away from Fortran-centric view of memory management to open up ICON for new possibilities
- Better compiler support for C/C++ than for Fortran
- Enable easier language interoperability with e.g., Gt4Py, Kokkos, …
- But: Legacy Fortran code still needs to be able to access data in the same way

# C++ Data / Memory Management – Why?

- ICON variables and their meta-data are organized in a linked list (varlist) implemented in Fortran
- Current functionality includes adding and removing elements and searching the list
- A C++ varlist implementation based on a quasi ordered map can make use of existing standard C++ functionality
- Future extension of functionality will also be simpler than a Fortran implementation

# C-Fortran interface

- Exploring the CFI (C-Fortran-Interface) in ISO_Fortran_binding.h
- *Agnostic* creation of arrays in C and their later clean use in Fortran
- Using the Fortran Standard (2018) or TS29113 (2012)
- Not yet supported by all compiler vendors

# Stand-alone advection

- Atmospheric tracer advection ideal example to test new concepts and develop software blueprint:
  - Representative for whole ICON, has 3D-stencil operations
  - Extensively utilizes ICON infrastructure
  - But conceptionally could be own submodule
  - OpenACC implementation exists for some methods

# Stand-alone advection

- Problems with current implementation:
  - More than 100 Fortran module dependencies
  - Data flow through global model data is not obvious
  - Complex interface, e.g.: `step_advection(p_patch, p_int_state, + 19 simple args)`
  - Complexity comes from huge derived types: `t_patch`, `t_int_state`, but derived type content mostly not used or not used directly

# Stand-alone advection

- Goals:
  - pool together “advection-owned” code and data
  - make data flow obvious
  - minimize interface complexity
  - have a stand-alone version using the same advection code as in the full model

# Summary and Conclusion

- Significant rewriting and refactoring of ICON is needed for scalable development on emerging architectures => **ICON-C**
- **preWarmWorld**: Assessment of programming paradigms and modular software blueprint
- Steps are underway in **preWarmWorld** together with partners in **ICON-C** and EXCLAIM to
  - rewrite the memory management in C++ with a C-Fortran interface
  - prepare a stand-alone version of the atmospheric tracer advection as a playground, on which to try out different programming paradigms

# Thank you for your attention!