
# ICON Community Interface

NatESM Workshop | Leipzig | 24.02.2026 | Nils-Arne Dreier (DKRZ)

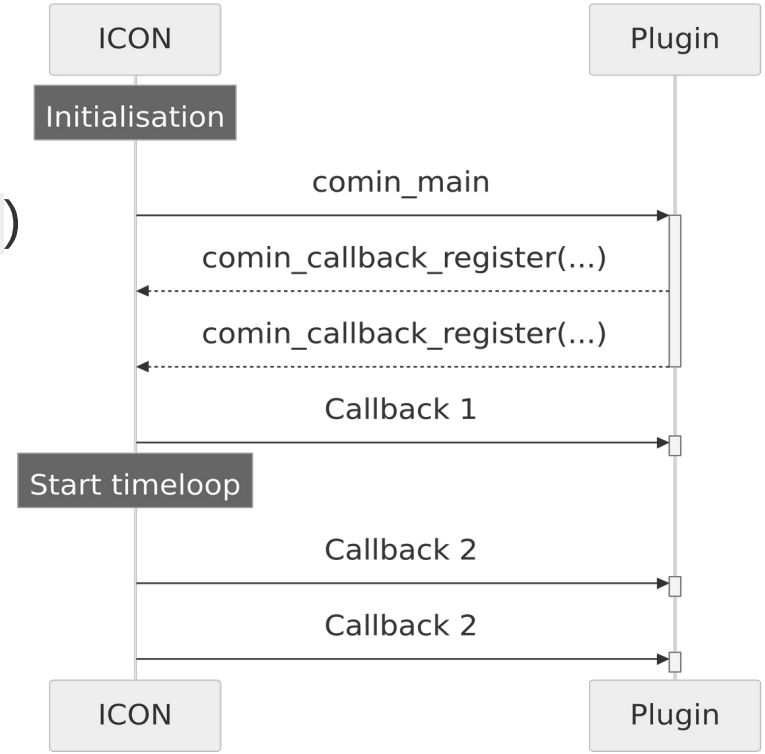
DKRZ DEUTSCHES KLIMARECHENZENTRUM Deutscher Wetterdienst DWD DLR JÜLICH Forschungszentrum WARM WORLD natESM

# What is ComIn? And Why?

- Extending ICON with custom code is *difficult*
- ICON is Open Source
- Extending ICON with custom code
  - Diagnostics
  - Specialized output
  - Physics
  - Insitu visualisation
  - …
- ComIn enables **everyone** to do sustainable science with ICON

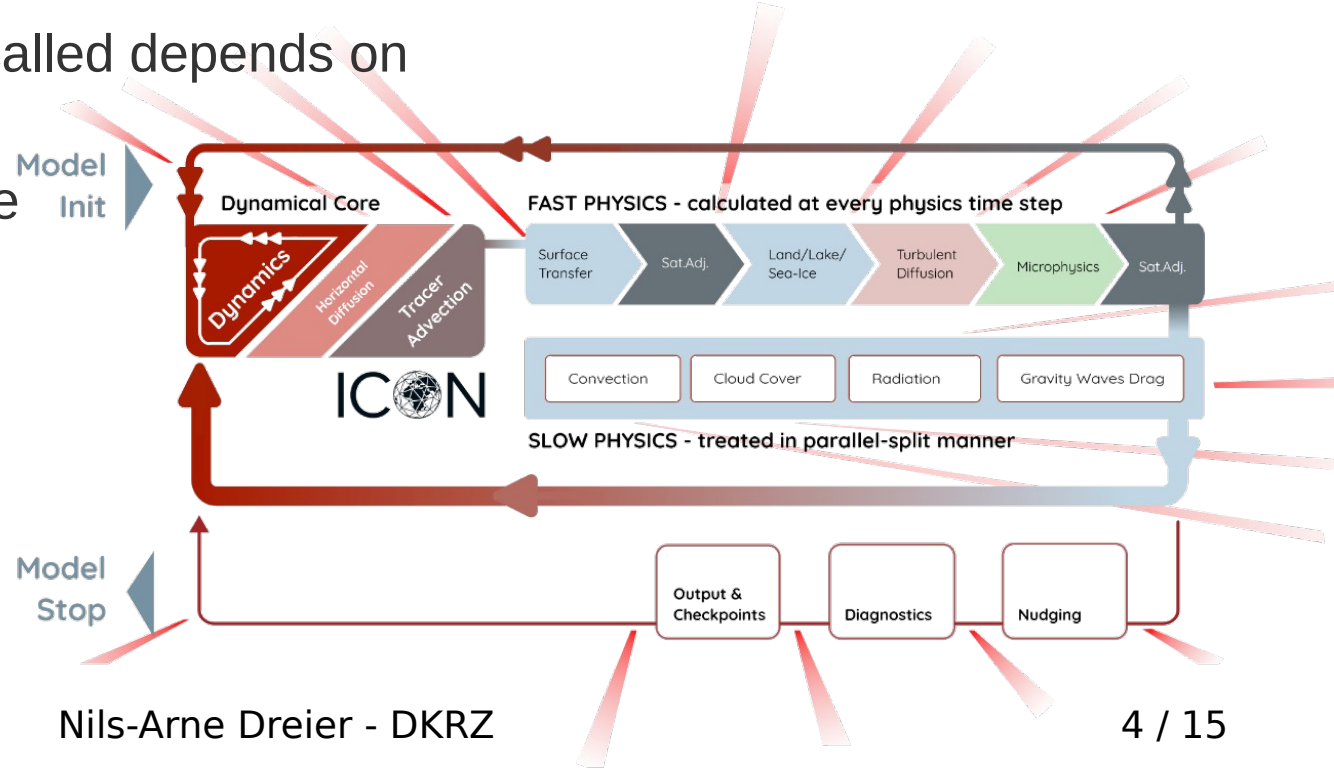
# Plugin Mechanism

- Developed and built independently from ICON
- Shared object loaded at runtime
- ICON callls primary constructor (`comin_main`)
- Plugins register callback for predefined entry points
- Use ComIn API in callback functions



# Callback Registry

- “before/after” nomenclature for example: `EP_ATM_WRITE_OUTPUT_BEFORE`
- Which entrypoints are called depends on the ICON configuration
- 42 entry points available



# Variable Access

- `comin_var_request_add`: Add variables to ICON (primary constructor)
  - Output with usual output modules
  - Datatypes: `double`, `float`, `int`
- `comin_var_get`: Get access to variable data
  - Read / Write
  - Automatic synchronisation for halo and GPU
- Variables are identified by a descriptor (Name, Domain ID)
- Access metadata

# Descriptive Data

- Grid information
  - cells/edge/vertices coordinates
  - ...
- Global
  - Number of Domains
  - ICON version
  - ...

◆ `comin_descrdata_types::t_comin_descrdata_global_data`

```
type comin_descrdata_types::t_comin_descrdata_global_data
```

Definition at line 40 of file comin_descrdata_types.F90.

**Data Fields**

| | | |
|---|---|---|
| character(len=:), allocatable | device_driver | Driver of the accelerator devic |
| character(len=:), allocatable | device_name | Name of the accelerator devic |
| character(len=:), allocatable | device_vendor | Vendor of the accelerator dev |
| integer(c_int) | grf_bdywidth_c | Cell row ordering (block index nudging zone. |
| integer(c_int) | grf_bdywidth_e | Edge row ordering (block inde |
| logical(c_bool) | has_device | Information about the compu Whether the host model uses accessed directly on the devi further details. |
| character(len=:), allocatable | host_git_branch | Host model version informati |
| character(len=:), allocatable | host_git_remote_url | Host model version informati |
| character(len=:), allocatable | host_git_tag | Host model version informati |
| character(len=:), allocatable | host_revision | Host model version informati |
| logical(c_bool) | lrestartrun | Whether this is a restarted ru |
| integer(c_int) | max_dom | Maximum number of model d |

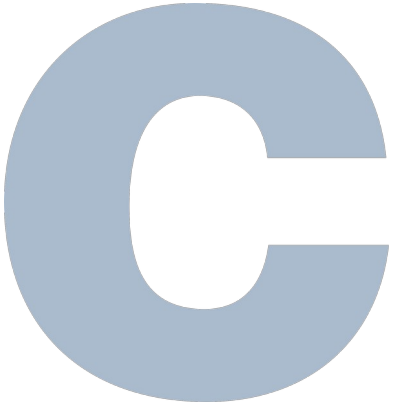
# Language Interoperability

```fortran
USE comin

SUBROUTINE comin_main() &
  BIND(C)

  comin_callback_register(…)

END SUBROUTINE
```



```c
#include <comin.h>

void comin_main(void){
  comin_callback_register(…)
}
```

```python
import comin

@comin.EP_SECONDARY_CONSTRUCTOR
def sec_ctr():
  …

@comin.EP_ATM_TIMELOOP_END
def timeloop_end():
  …
```

# Python example 🐍

```
import comin
import numpy as np

ep = comin.EP_ATM_WRITE_OUTPUT_BEFORE
dom_id = 1

comin.var_request_add((“temp_celsius”, dom_id), lmodexclusive=False)
comin.metadata_set((“temp_celsius”, dom_id), zaxis=comin.COMIN_ZAXIS_3D, units=”°C”)

@comin.EP_SECONDARY_CONSTRUCTOR
def init():
  global temp, temp_celsius
  temp = comin.var_get([ep], (“temp”, dom_id), comin.COMIN_FLAG_READ)
  temp_celsius = comin.var_get([ep], (“temp_celsius”, dom_id), comin.COMIN_FLAG_WRITE)

@ep
def before_output():
  np.asarray(temp_celsius)[…] = np.asarray(temp)-273.15
```

# CMake Integration

- Build C and Fortran plugins
- Auxiliary functions for adding tests (CTest) with the replay tool

```
project(MyPlugin)

include(CTest)

find_package(ComIn REQUIRED)

add_library(dummy_plugin MODULE
  dummy.c )
target_link_libraries(dummy_plugin
  PRIVATE ComIn::Plugin )

comin_add_replay_test(
  NAME test_1
  REPLAY_DATA "test_nwp_R02B04" )

comin_test_add_plugin(
  TEST test_1
  NAME "plugin"
  PLUGIN_LIBRARY "$<TARGET_FILE:dummy_plugin>" )
```

# ICON Namelist

```
&comin_nml
  plugin_list(1)%name           = "my_plugin"
  plugin_list(1)%plugin_library = "libpython_adapter.so"
  plugin_list(1)%options        = "/path/to/python_plugin.py"
!
  plugin_list(2)%name           = "my_c_plugin"
  plugin_list(2)%plugin_library = "/path/to/plugin.so"
/
```

- Other optional options:
  - Communicator
  - Verbosity
  - Custom primary constructor

# Record & Replay

Serialize ICON data to replay the experiment with ComIn Plugins for testing and debugging

- comin_run_recorder_**plugin**
  - Descriptive data
  - Sequence of Entrypoints
  - Output: NetCDF
- comin_var_recorder_**plugin**
  - Set of variables at one entrypoint
  - Output: NetCDF
- comin_replay (executable)
  - Mimics ICON
  - Reads NetCDF from run_recorder
  - Provides Descriptive Data
  - Calls Entrypoint in recorded order
- comin_var_replay_**plugin**
  - Reads NetCDF from var_recorder
  - Fills variable with data at one entrypoint

# GPU Support 🚀

- Request explicit access to device memory: `COMIN_FLAG_DEVICE`
- If **not** set: Automatic host ↔ device synchonization
  - Only where necessary (before read access / after write access)
  - Ensures that also not-ported Plugins work in GPU settings

## Fortran

- OpenACC annotations

## C/C++

- `comin_var_get_ptr_device`
- Use any GPU framework (Kokkos / Cuda / ...)

## Python

- CUDA array interface
- `cupy` / `numba`
- `pytorch` (?)

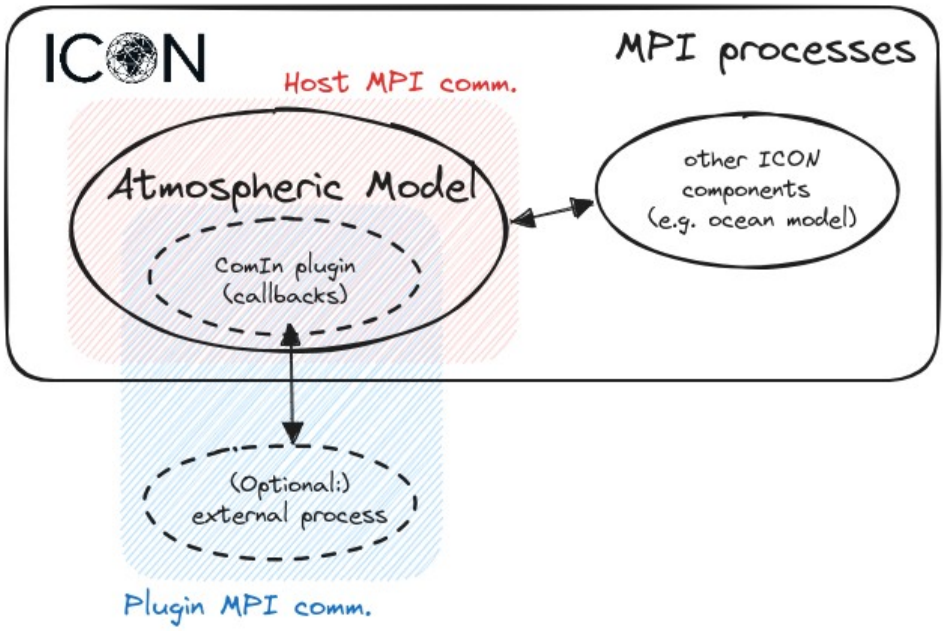
# MPI Communicators

**Host Communicator**

- All ICON processes

**Plugin Communicator**

- Offload work to dedicated processes
- MPI Handshake to be compatible with YAC and other components
- Requires MPI MPMD settings




```
mpirun -n 42 ./icon : -n 1 python script.py
```

# Recap

- **Extend ICON Easily**
  ComIn enables sustainable extension of the ICON model.
- **Plugin Architecture**
  Uses independent plugins & a callback system.
- **Multi-Language Support**
  Supports Fortran, C++, & Python development.
- **Debug with Record & Replay**
  Facilitates testing via recording & replaying runs.
- **Portable and Extensible**
  MPI and GPU aware

# Resources

**Documentation**
https://comin.icon-model.org/

**Reach the developers**
comin@icon-model.org

**ComIn Excercise Notebooks**
https://gitlab.dkrz.de/icon-comin/comin-training-exercises

**GitLab Project**
https://gitlab.dkrz.de/icon-comin/comin

# Thank you!