Task Description of Group Competition

The group competition aims to foster cooperation and friendship among teams. The group competition for the ASC26 Final Competition is the optimization of ICON application.

The source code and a sample test case are available for download from the following link:

https://github.com/ASC-Competition/ASC26-icon

(Git LFS is required; please run `git lfs install` before clone)

1. Introduction

ICON, which obtains its name from the usage of spherical grids derived from the icosahedron (ICO) and the non-hydrostatic (N) dynamics, is a flexible, scalable, high-performance modelling framework for weather, climate and environmental prediction. ICON includes component models for the atmosphere, the ocean and the land, as well as chemical and biogeochemical cycles, all implemented on the basis of common data structures and sharing the same efficient technical infrastructure. ICON can be used in the most diverse resolutions and configurations to enable a whole range of applications — from global and regional weather forecasts and climate projections to very high-resolution digital twins of the Earth system. By providing actionable information for society and advances our understanding of the Earth's climate system. Thanks to these modeling capabilities, ICON is evolving into an internationally recognized and widely used modeling framework that advances our understanding of the Earth's climate system and provides actionable information for solving problems of high societal relevance.

2. Test case

- The test case provided is an aquaplanet configuration on the R02B04 grid.
- The test case simulation is based on a restart configuration that defines an atmosphere in equilibrium.
- This directory includes:
    - The test case configuration `ape_from_spinup.config`
    - The restart data directory `ape_from_spinup_restart_atm_19790701T000000Z.nc`
    - The restart date file `ape_from_spinup.date`
    - The restart run file `ape_from_spinup.run` (for reference)
    - Necessary input files (`pool/*`) (**do not change** the input file internal directory structure)

The following is the structure of the directories relevant to this contest:

ASC26-icon

```
├──── icon-model
│       ├──── AUTHORS.TXT
│       ├──── collect.extra-libs.in
│       ├──── config
│       ├──── config.h.in
│       ├──── configure
```

```
│   ├── configure.ac
│   ├── CONTRIBUTING.md
│   ├── data
│   ├── depgen.c.config.in
│   ├── depgen.f90.config.in
│   ├── deplist.config.in
│   ├── doc
│   ├── etc
│   ├── externals
│   ├── icon.mk.in
│   ├── inlib.mk.in
│   ├── LICENSES
│   ├── m4
│   ├── Makefile.in
│   ├── make_runscripts
│   ├── ragnarok
│   ├── README.md
│   ├── RELEASE_NOTES.md
│   ├── REUSE.toml
│   ├── run
│   ├── schedulers
│   ├── scripts
│   ├── src
│   ├── support
│   ├── test
│   ├── utils
│   └── vertical_coord_tables
└── testcase
    ├── ape_from_spinup.config
    ├── ape_from_spinup.date
    ├── ape_from_spinup_restart_atm_19790701T000000Z.nc
    ├── ape_from_spinup.run
    └── pool
```

3. Source code of ICON

You may use the `icon-model` source code in the provided package. However, this method may require additional manual configuration of external submodules.(there's lfs files in it, makesure you using git lfs install)

Recommendation: For a more straightforward way to obtain ICON along with all submodules, you can retrieve source code from the official repository, by cloning the `release-2025.10-public` branch of the open-source ICON version with the `--recurse-submodules` option.

4. Software and datasets

1) The required environment for running ICON is as follows:

- C, CXX and Fortran compilers
- GNU Make v3.81+, CMake v3.18+, Perl v5.10+
- Software Libraries: MPI, CURL, ZLIB, HDF5, NetCDF (C and Fortran, with NetCDF-4 support), FYAML, XML2, ECCODES, LAPACK and BLAS
- Python v3.9+, a Python environment with the 'six' and 'jinja2' packages installed

*Other dependencies may also be required, depending on the hardware platform you are using, see ICON building documentation for more information.

2) All required input data is included in the provided package; no external datasets are needed.

5. Build ICON

- Configure ICON options and compilation flags with a configuration wrapper file:
  - Navigate to the `icon-model` directory and create a new directory at the path `config/asc/<config-file-name>`.
  - Create your configuration wrapper file using as a reference the templates in the `config/` directory (e.g., `config/generic/gcc`). This configuration file might need to be adapted or adjusted to match your target hardware platform, refer to the following resources:
    - ICON building documentation
    - Info on supported environments
  - Prepare the build by running your wrapper file from the `icon-model` directory
- Build the source code with `make` (use flags like `-j32` to speedup)
- This step will generate an executable binary file `icon` at `bin/`

6. Runscript Configuration

6.1 Quick start

- Create and activate a conda environment with the required dependencies
- Go to directory `icon-model`
- Copy `ape_from_spinup.config` config file to `run/`
  - adjust the `INPUT_ROOT` parameter to point to the provided input file directory `pool/data` if necessary.
  - The case settings are:
    EXP_ID = ape_from_spinup
    INITIAL_DATE = 1979-01-01
    FINAL_DATE = 1979-09-01
    ATMO_TIME_STEP = PT1M
    INTERVAL = P2M
- Run `./utils/mkexp/mkexp run/ape_from_spinup.config` to create the experiment directory tree including `scripts/`, `data/`, `work/`, `log/`
- Copy `ape_from_spinup.date` to `experiments/ape_from_spinup/scripts`, the restart date is `1979-07-01T00:00:00.000`
- Copy `ape_from_spinup_restart_atm_19790701T000000Z.nc` to `experiments/ape_from_spinup/work/`

6.2 Notes

- Model and infrastructure settings are configured through the `mkexp` tool. The experiment config (e.g. `<case_name>.config`) is used to create a directory structure and populate it with some files, including the run files (`<case_name>.run_start` for runs without restart, `<case_name>.run` for runs with restart)
- Run files include experiment default settings for both model and infrastructure configuration, which can be overridden manually in the files. **Do not modify** model setting parameters in the experiment config file `<case_name>.config`
- Model parallelization settings are provided with parallel_nml (see namelist overview documentation and chapter 8 of the icon tutorial)
- If you want to rename the experiment, rename the config file to `<new_case_name>.config`, set EXP_ID = <new_case_name>; and update the input file names accordingly.
- Run at least two months of simulation time to generate stable outputs.

7. Run the Test case

- Go to `icon-model/experiments/ape_from_spinup/scripts`, adjust run parameters in the `ape_from_spinup.run` file as needed based on your platform
- Submit the run script `ape_from_spinup.run`
    - In case the simulation crashes due to `*symlink not found*`, go to `ape_from_spinup/work/run_*/` and add the symlink `multifile_restart_atm.mfr` pointing to `ape_from_spinup/work/ape_from_spinup_restart_atm_19790701T000000Z.nc` (running `ls multifile_restart_atm.mfr` should show `attributes.nc`, `patch1_0.nc`)
- Find the experiment log in `ape_from_spinup.run.log` (log archive can be found in `ape_from_spinup/log`). If the simulation completes successfully, you will see this in log file:

```
============================
Script run successfully: OK
============================
```

- The output files will be generated in `work/run_<start_time>-<end_time>`

8. Important Notes

- **This software is permitted to be run and modified only on CPUs. Use of GPUs is strictly prohibited, as doing so will result in a score of zero.**
- All computational workloads **must be validated to ensure result accuracy**, with optimization targeted at minimizing execution time. Furthermore, the modified code must be mathematically equivalent and general-purpose; it should not be tailored to perform well only for some specific test cases.