Using YAC for a better handling of climate model output

Vera Maurer (DWD)

Ralf Müller, Nils-Arne Dreier (DKRZ)

Why do we need a better handling of climate model output?

- ICON output in NetCDF format is not CF compliant
- variables like `tot_prec`, `rain_con`, …, `runoff_*`, `gust10`, `snow_melt`, `dursun`, `tmin_2m`, `tmax_2m` depend on specific settings (e.g. `precip_interval`, `runoff_interval`, …), explicitly set via the namelists → output is erroneous if intervals and respective output intervals are not consistent!
- variables which are not / cannot be accumulated / averaged internally (like fluxes and radiation, which are internally aggregated since model start) are aggregated using mvstream (`operation = 'mean'`)
  - mvstream aggregation can not be applied together with horizontal or vertical interpolation (i.e., only for surface variables or variables on model levels)
  - variables aggregated via mvstream are not correctly labelled (no `cell_methods` or `time_bnds`)
  - the time stamps are always set to the end of the output interval

Why do we need a better handling of climate model output?

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

current workaround in SPICE:

- ARCH corrects files for CF conventions (function `iconcor` [`functions.inc`], calls `correct_cf.py`)
  - variable renaming according to a table (`mapping_to_cosmo.csv`)
  - setting of `standard_name`
  - adding of `time_bnds`
  - adding of `cell_methods`
  - however, for aggregated variables, time stamp still not set correctly



- POST calls the `timeseries` function
  - cats together all time steps (by month) via `ncrcat`
  - calls `remap2rot` function
    - performs the remapping and and corrects the time stamps (function `cell_methods_time`)

CF corrections in ARCH


```
netcdf icon_20020101T000000Z {
dimensions:
        time = UNLIMITED ; // (1 currently)
        ncells = 4156 ;
        vertices = 3 ;
variables:
        double time(time) ;
                time:standard_name = "time" ;
                time:units = "minutes since 2002-1-1 00:00:00" ;
                time:calendar = "gregorian" ;
                time:axis = "T" ;
        double clon(ncells) ;
                clon:standard_name = "longitude" ;
                clon:long_name = "center longitude" ;
                clon:units = "radian" ;
                clon:bounds = "clon_bnds" ;
        double clon_bnds(ncells, vertices) ;
        double clat(ncells) ;
                clat:standard_name = "latitude" ;
                clat:long_name = "center latitude" ;
                clat:units = "radian" ;
                clat:bounds = "clat_bnds" ;
        double clat_bnds(ncells, vertices) ;
        float qhfl_s(time, ncells) ;
                qhfl_s:standard_name = "qhfl_s" ;
                qhfl_s:long_name = "surface moisture flux" ;
                qhfl_s:units = "Kg m-2 s-1" ;
                qhfl_s:param = "6.0.2" ;
                qhfl_s:CDI_grid_type = "unstructured" ;
                qhfl_s:number_of_grid_in_reference = 1 ;
                qhfl_s:coordinates = "clat clon" ;
                qhfl_s:_FillValue = -9.99e-08f ;
                qhfl_s:missing_value = -9.99e-08f ;
```

```
netcdf icon_20020101T000000Z {
dimensions:
        time = UNLIMITED ; // (1 currently)
        bnds = 2 ;
        ncells = 4156 ;
        vertices = 3 ;
variables:
        double time(time) ;
                time:standard_name = "time" ;
                time:bounds = "time_bnds" ;
                time:units = "seconds since 2002-1-1 00:00:00" ;
                time:calendar = "gregorian" ;
                time:axis = "T" ;
        double time_bnds(time, bnds) ;
        double clon(ncells) ;
                clon:standard_name = "longitude" ;
                clon:long_name = "center longitude" ;
                clon:units = "radian" ;
                clon:bounds = "clon_bnds" ;
        double clon_bnds(ncells, vertices) ;
        double clat(ncells) ;
                clat:standard_name = "latitude" ;
                clat:long_name = "center latitude" ;
                clat:units = "radian" ;
                clat:bounds = "clat_bnds" ;
        double clat_bnds(ncells, vertices) ;
        float AEVAP_S(time, ncells) ;
                AEVAP_S:standard_name = "water_evaporation_amount" ;
                AEVAP_S:long_name = "surface moisture flux" ;
                AEVAP_S:units = "kg m-2" ;
                AEVAP_S:CDI_grid_type = "unstructured" ;
                AEVAP_S:number_of_grid_in_reference = 1 ;
                AEVAP_S:coordinates = "clat clon" ;
                AEVAP_S:_FillValue = -9.99e-08f ;
                AEVAP_S:missing_value = -9.99e-08f ;
                AEVAP_S:cell_methods = "time: sum" ;
```

CF corrections in ARCH


```
netcdf icon_20020101T000000Z {
dimensions:
        time = UNLIMITED ; // (1 currently)
        ncells = 4156 ;
        vertices = 3 ;
variables:
        double time(time) ;
                time:standard_name = "time" ;
                time:units = "minutes since 2002-1-1 00:00:00" ;
                time:calendar = "gregorian" ;
                time:axis = "T" ;
        double clon(ncells) ;
                clon:standard_name = "longitude" ;
                clon:long_name = "center longitude" ;
                clon:units = "radian" ;
                clon:bounds = "clon_bnds" ;
        double clon_bnds(ncells, vertices) ;
        double clat(ncells) ;
                clat:standard_name = "latitude" ;
                clat:long_name = "center latitude" ;
                clat:units = "radian" ;
                clat:bounds = "clat_bnds" ;
        double clat_bnds(ncells, vertices) ;
        float qhfl_s(time, ncells) ;
                qhfl_s:standard_name = "qhfl_s" ;
                qhfl_s:long_name = "surface moisture flux" ;
                qhfl_s:units = "Kg m-2 s-1" ;
                qhfl_s:param = "6.0.2" ;
                qhfl_s:CDI_grid_type = "unstructured" ;
                qhfl_s:number_of_grid_in_reference = 1 ;
                qhfl_s:coordinates = "clat clon" ;
                qhfl_s:_FillValue = -9.99e-08f ;
                qhfl_s:missing_value = -9.99e-08f ;
```

```
netcdf icon_20020101T000000Z {
dimensions:
        time = UNLIMITED ; // (1 currently)
        bnds = 2 ;
        ncells = 4156 ;
        vertices = 3 ;
variables:
        double time(time) ;
                time:standard_name = "time" ;
                time:bounds = "time_bnds" ;
                time:units = "seconds since 2002-1-1 00:00:00" ;
                time:calendar = "gregorian" ;
                time:axis = "T" ;
        double time_bnds(time, bnds) ;
        double clon(ncells) ;
                clon:standard_name = "longitude" ;
                clon:long_name = "center longitude" ;
                clon:units = "radian" ;
                clon:bounds = "clon_bnds" ;
        double clon_bnds(ncells, vertices) ;
        double clat(ncells) ;
                clat:standard_name = "latitude" ;
                clat:long_name = "center latitude" ;
                clat:units = "radian" ;
                clat:bounds = "clat_bnds" ;
        double clat_bnds(ncells, vertices) ;
        float AEVAP_S(time, ncells) ;
                AEVAP_S:standard_name = "water_evaporation_amount" ;
                AEVAP_S:long_name = "surface moisture flux" ;
                AEVAP_S:units = "kg m-2" ;
                AEVAP_S:CDI_grid_type = "unstructured" ;
                AEVAP_S:number_of_grid_in_reference = 1 ;
                AEVAP_S:coordinates = "clat clon" ;
                AEVAP_S:_FillValue = -9.99e-08f ;
                AEVAP_S:missing_value = -9.99e-08f ;
                AEVAP_S:cell_methods = "time: sum" ;
```

Why do we need a better handling of climate model output?

current workaround in SPICE:

- ARCH corrects files for CF conventions (function `iconcor` [`functions.inc`], calls `correct_cf.py`)
  - variable renaming according to a table (`mapping_to_cosmo.csv`)
  - setting of `standard_name`
  - adding of `time_bnds`
  - adding of `cell_methods`
  - however, for aggregated variables, time stamp still not set correctly

with the output coupling, we can do all these steps online!

- POST calls the `timeseries` function
  - cats together all time steps (by month) via `ncrcat`
  - calls `remap2rot` function
    - performs the remapping and and corrects the time stamps (function `cell_methods_time`)

Using YAC - the output coupling

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

```
mpirun -n 4 ./icon : \
       -n 1 python output_component_CLM.py output.nc \
            atm,icon_grid,temp \
            atm,icon_grid,u \
            --bounds -180 180 -90 90 \
            --res 360 180 \
            --output-interval PT1H
```

ICON

YAC 3.21.0
Yet Another Coupler

YAC interfaces in ICON
(mo_output_coupling.f90)
activated via namelist

CLIENT

output_component_CLM.py

- horizontal interpolation
- time management of the output
- temporal aggregation of fields
- writes the NetCDF file

SERVER

What the output coupling already can do



Output Coupling

The **output coupling** feature provides a generic interface for connecting ICON to external output components using the YAC (Yet Another Coupler) library. This allows users to develop custom output components that run independently of ICON, enabling flexible post-processing, analysis, and visualization workflows.

Key Features

- **Generic Interface**: Works with any YAC-compatible output component
- **Flexible Grid Interpolation**: YAC provides various interpolation methods
- **Temporal Interpolation**: YAC handles temporal coupling automatically
- **Metadata**: Transfers CF conventions and GRIB2 metadata defined in ICON to the output component
- **3D Field Support**: Handles multi-level atmospheric and oceanic data with collection sizes
- **Fractional Masking**: Optional validity masks for handling ocean data

https://docs.icon-model.org/documentation/infrastructure/output_coupling.html#ref-output-coupling



Using YAC - the output coupling


```
mpirun -n 4 ./icon : \
       -n 1 python output_component_CLM.py output.nc \
       CLM_config.yaml
```

ICON

YAC 3.21.0
Yet Another Coupler

YAC interfaces in ICON
(mo_output_coupling.f90)
activated via namelist

CLIENT

natESM

sprint application submitted

output_component_CLM.py

- horizontal interpolation
- time management of the output
- temporal aggregation of fields
- writes the NetCDF file
- add a user-interface including a variable mapping with an option to set further attributes

SERVER

Plan for natESM sprint

Output server for climate modelling

- ➔ provides a **user interface** for the output coupling (e.g. based on YAML files)
  - ➔ e.g. using a dictionary for translating **variable names**
  - ➔ custom settings for variable and global NetCDF **attributes**
  - ➔ different target **output grid** functionality (define the grid in the configuration via the grid spacing or via an input file); also native ICON grid should be possible
  - ➔ option for simple post operations (e.g. adding of 2 variables)
  - ➔ optional: statistics (e.g. monthly mean diurnal cycle, variance)
- ➔ tests with a higher resolution and a large number of output variables
- ➔ tests for ICON-XPP (global setup with ocean coupling)