A Pythonic way to use YAC

Nov 14 2023 - Nils-Arne Dreier - natESM Training

Why Python?!

- **simple**
  - accessible for *beginners*
  - rapid prototyping
- **open source** and active community
- extensive **libraries** – *batteries included*
  - scientific computing: `numpy`, `scipy`, …
  - visualization: `matplotlib`, `cartopy`, …
  - analytics: `pandas`, `xarray`, …
  - machine learning: `pytorch`, `tensorflow`, …
  - …

YAC python bindings

- **slim** python wrappers around `yac_interface.h`
- generated with **cython**
- `./configure --enable-python-bindings`
- depends on `numpy` (data buffers)
- YAC IDs → classes

Applications

- in-situ visualizations
- hiopy: hierarchical output on healpix grid
- asynchronous input - ozone and aerosols

planned:

- tropical cyclone tracker
- model - sensor comparison

Coupling modes

- parallel:
  - MPI MPMD paradigm, e.g.

```
mpirun -n 4 ./icon : -n 1 python myscript.py
```

- sequential:
  - manually: embedded python
  - spoiler: *ComIn* allows to embed python into ICON

HowTo: definitions

```python
from yac import *
yac = YAC()

comp = yac.def_comp("python_component")
lon = np.linspace(0, 2*np.pi, 360, endpoint=False)
lat = np.linspace(0, np.pi, 180)
grid = Reg2dGrid("python_grid", lon, lat)
points = grid.def_points(Location.CORNER, lon, lat)

field = Field.create("tas", comp, points, collection_size=1,
                     timestep="PT3H", timeunit=TimeUnit.ISO_FORMAT)
```

HowTo: configure coupling

- in the API

```
nnn = InterpolationStack()
nnn.add_nnn(NNNReductionType.AVG, 1, 1.)

yac.def_couple("atmo", "icon_atmos_grid", "tas"
               "python_component", "python_grid", "tas",
               coupling_timestep="PT3H", timeunit=TimeUnit.ISO_FORMAT,
               time_reduction=Reduction.TIME_NONE, interp_stack=nnn)
```

- *or in a yaml file*

HowTo: synchronization

config synchronization:

```python
yac.sync_def()
```

access metadata

```python
for comp_name in yac.component_names:
    print(yac.get_component_metadata(comp))
    for field_name in yac.get_field_names(comp_name, "some_grid"):
        print(yac.get_field_metadata(comp_name, "some_grid", field_name))
```

end definition phase:

```python
yac.enddef()
```

HowTo: get/put

```
data = None
for t in range(no_timesteps):
    data, info = field.get(data)
    ## Do anything with data
```

Questions?

Hands-On!

- groups of 2 or more people

Setup

- based on `run/exp.esm_bb_ruby0`
- one *Levante* node (32 processes x 4 threads)
- coupled:
  - atmosphere: R2B4 (16 procs)
  - ocean: R2B6 (16 procs)
- “output coupling”

```
Registers all variables from ICON's variable list in YAC (with CF metadata)
```

Allocate an interactive session

```
salloc -p compute -A <account> --reservation=natESM -t 02:00:00 -N 1
```

How to run

- create an experiment directory
- execute

```
/work/k20200/k202160/natesm-workshop/exp.esm_bb_ruby0.run
```

creates all needed files in the ***current*** *directory*

- add components to `mpmd.conf` (steal processes from atmo or ocean)
- rerun by running `./exp.esm_bb_ruby0.run`

Example components

- `/work/k20200/k202160/natesm-workshop/examples`
  - `simple_output.py`

    receives one field on a regular grid and stores it in a NetCDF file. Takes the filename and the source field description as arguments
  - `plot_barbs.py`

    visualizes wind over europe
  - `dump_metadata.py`

    dumps all metadata that exists after `enddef` in a yaml file `metadata.yaml`

Tasks

- start with
  1. run the experiment
  2. understand and run the examples
- inspiration:
  - *optimize* `plot_barbs.py` such that only the cells that are needed for the visualization are registered in YAC
  - plot the ocean surface temperature in `plot_barbs.py`
  - modify `simple_output.py` for regional output (e.g. the Canaries)
  - write a YAC component that computes the mean surface temperature of the northern atlanic for every timestep and finally plots it in a figure

Cheatsheet

- run a interactive session:

```
salloc -p compute -A <account> --reservation=natESM -t 02:00:00 -N 1
```

- Tasks:
  1. run the experiment
  2. understand and run the examples

important paths and files:

```
/work/k20200/k202160/natesm-workshop/exp.esm_bb_ruby0.run
/work/k20200/k202160/natesm-workshop/examples
mpmd.conf
```

Documentation: https://dkrz-sw.gitlab-pages.dkrz.de/yac/