ALFRED-WEGENER-INSTITUT
HELMHOLTZ-ZENTRUM FÜR POLAR-
UND MEERESFORSCHUNG

Pymor v1.0.0

CMORize your data with Pymor

Paul Gierz, Pavan Siligam, Miguel Andrés-Martínez - May 2025

HPC & Data Processing

HELMHOLTZ

Features

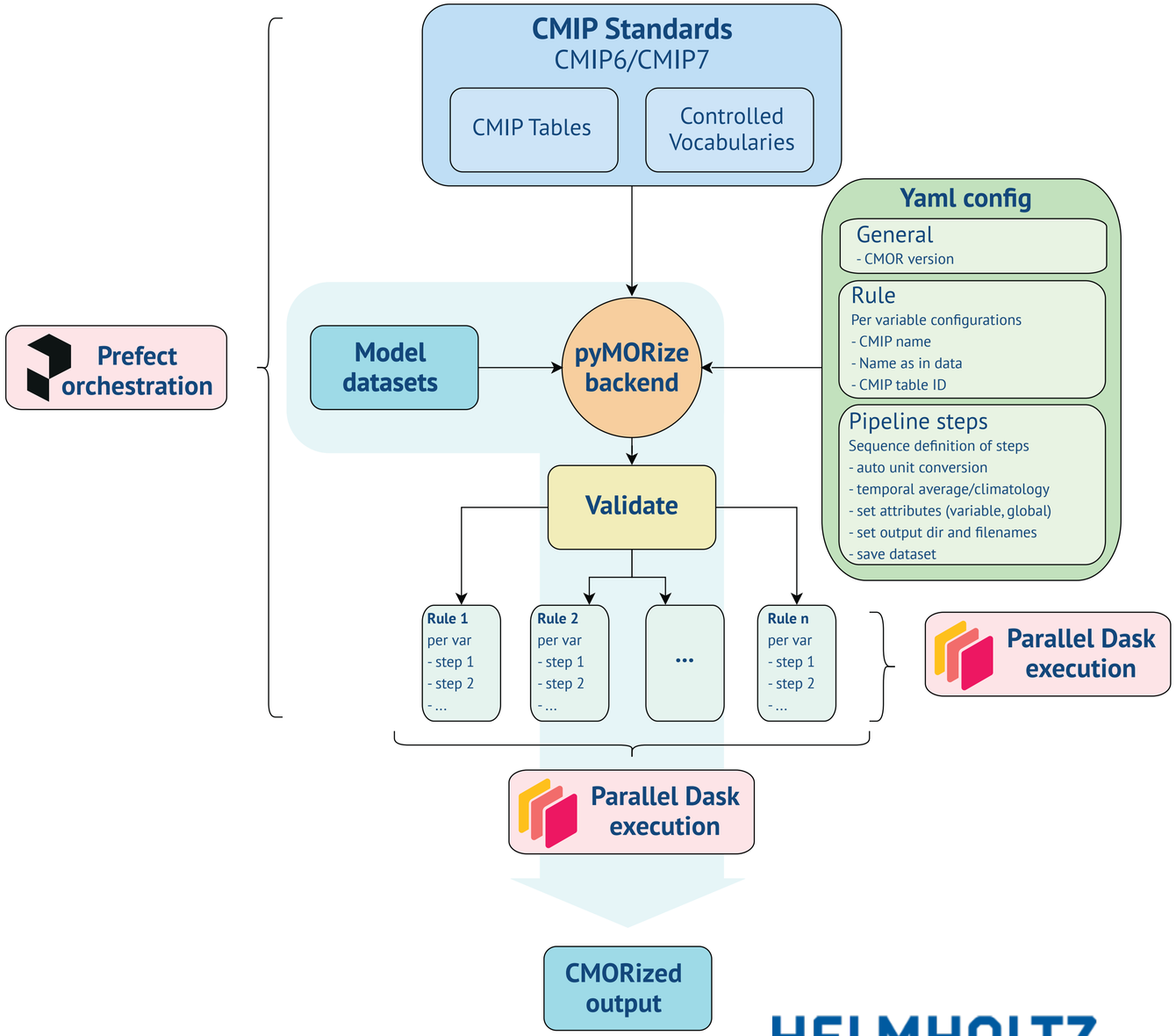
- Python package to CMORize data
- Built to support CMIP6. CMIP7 interface is planned as well.
- Provides full control on processing steps in the pipeline
- Easy to include custom processing steps in the pipeline
- Elegant units conversion handling
- Parallel execution of processing step on single or multiple nodes
- Uses yaml configuration as input
- Design with FESOM (ocean model) in mind, but could work for any netcdf

Overview

- Yaml config files holds configuration for all aspects
  - Rule for each variable
  - Pipeline steps for each rule
  - Dask configuration
- Parallelisation happens with each rule and across rules
- Prefect caches results at all processing stages

https://github.com/WCRP-CMIP/CMIP6_CVs

pyMORize workflow



HELMHOLTZ

Yaml config

```yaml
general:
  cmor_version: "CMIP6"
  CMIP_Tables_Dir: "../../cmip6-cmor-tables/Tables"
  CV_Dir: ../../cmip6-cmor-tables/CMIP6_CVs
pymor:
  warn_on_no_rule: False
  dask_cluster: "local"
  enable_output_subdirs: False
rules:
  - name: process_CO2f
    inputs:
      - path: ../data
        pattern: CO2f_fesom_.*nc
    cmor_variable: fgco2
    model_variable: CO2f
    output_directory: .
    variant_label: r1i1p1f1
    experiment_id: piControl
    source_id: AWI-CM-1-1-HR
    model_component: seaIce
    grid_label: gn
    pipelines:
      - default
pipelines:
  - name: default
    steps:
      - "pymor.core.gather_inputs.load_mfdataset"
      - "pymor.std_lib.generic.get_variable"
      - "pymor.std_lib.time_average"
      - "pymor.std_lib.units.handle_unit_conversion"
      - "pymor.std_lib.global_attributes.set_global_attributes"
      - "pymor.std_lib.generic.trigger_compute"
      - "pymor.std_lib.files.save_dataset"
  - name: partial
    steps:
      - "pymor.core.gather_inputs.load_mfdataset"
      - "pymor.std_lib.generic.get_variable"
      - "pymor.std_lib.units.handle_unit_conversion"
```

Sections

- general
- pymor
- rules
- pipelines
- distributed
- jobqueue

Features list

- Rules
- Pipelines
- Unit
- Time average
- Custom step

Rules

ALFRED-WEGENER-INSTITUT
HELMHOLTZ-ZENTRUM FÜR POLAR- UND MEERESFORSCHUNG

table_id in Rule

```
rules:
  - name: process_CO2f
    cmor_variable: fgco2
    model_variable: CO2f
    table_id: Omon
    …
```

```
rule:
  - name: process_CO2f
    cmor_variable: fgco2
    model_variable: CO2f
    table_id: Omon
    …
```

OR

no table_id in Rule

```
rules:
  - name: process_CO2f
    cmor_variable: fgco2
    model_variable: CO2f
    …
```

```
rule:
  - name: process_CO2f
    cmor_variable: fgco2
    model_variable: CO2f
    table_id: Omon
    …
```

```
rule:
  - name: process_CO2f
    cmor_variable: fgco2
    model_variable: CO2f
    table_id: Oyr
    …
```

CMORIZER

Frequency checker applied on each rule

```
Source data temporal
frequency must be finer
than table frequency
```

```
rule:
  - name: process_CO2f
    cmor_variable: fgco2
    model_variable: CO2f
    table_id: Omon
    …
```

OR

```
rule:
  - name: process_CO2f
    cmor_variable: fgco2
    model_variable: CO2f
    table_id: Omon
    …
```

HELMHOLTZ

Pipelines

ALFRED-WEGENER-INSTITUT
HELMHOLTZ-ZENTRUM FÜR POLAR-
UND MEERESFORSCHUNG

Single pipeline

```
rules:
  - name: process_CO2f
    …
    pipelines:
      - default
pipelines:
  - name: default
    steps:
      - “pymor.core.gather_inputs.load_mfdataset"
      - “pymor.std_lib.generic.get_variable”
      - “pymor.std_lib.time_average”
      - “pymor.std_lib.units.handle_unit_conversion”
      - “pymor.std_lib.global_attributes.set_global_attributes”
      - “pymor.std_lib.generic.trigger_compute”
      - "pymorize.files.save_dataset"
```

Split pipeline

```
rules:
  - name: process_CO2f
    …
    pipelines:
      - stage1
      - stage2
pipelines:
  - name: stage1
    steps:
      - “pymor.core.gather_inputs.load_mfdataset"
      - “pymor.std_lib.generic.get_variable”
      - “pymor.std_lib.time_average”
  - name: stage2
    steps:
      - “pymor.std_lib.units.handle_unit_conversion”
      - “pymor.std_lib.global_attributes.set_global_attributes”
      - “pymor.std_lib.generic.trigger_compute”
      - “pymor.std_lib.files.save_dataset”
```


HELMHOLTZ

Units handling

- Pymorize has built-in support for handling chemical elements in units

  Consider following unit conversion

  ```
  mmolC —> kg  # Express milli moles of Carbon in kilograms
  ```

  Typically, end users do this by hand.

  ```
  1) mmolC -> molC: / 1e3
  2) molC -> gC: * 12.0107  # molecular weight of Carbon in grams
  3) gC -> kgC: / 1e3
  —> 1/1e3 * 12.0107 / 1e3
  ```

- No need to do manual conversion anymore

pint

CalebBell/
chemicals

chemicals: Chemical database of Chemical Engineering Design Library (ChEDL)

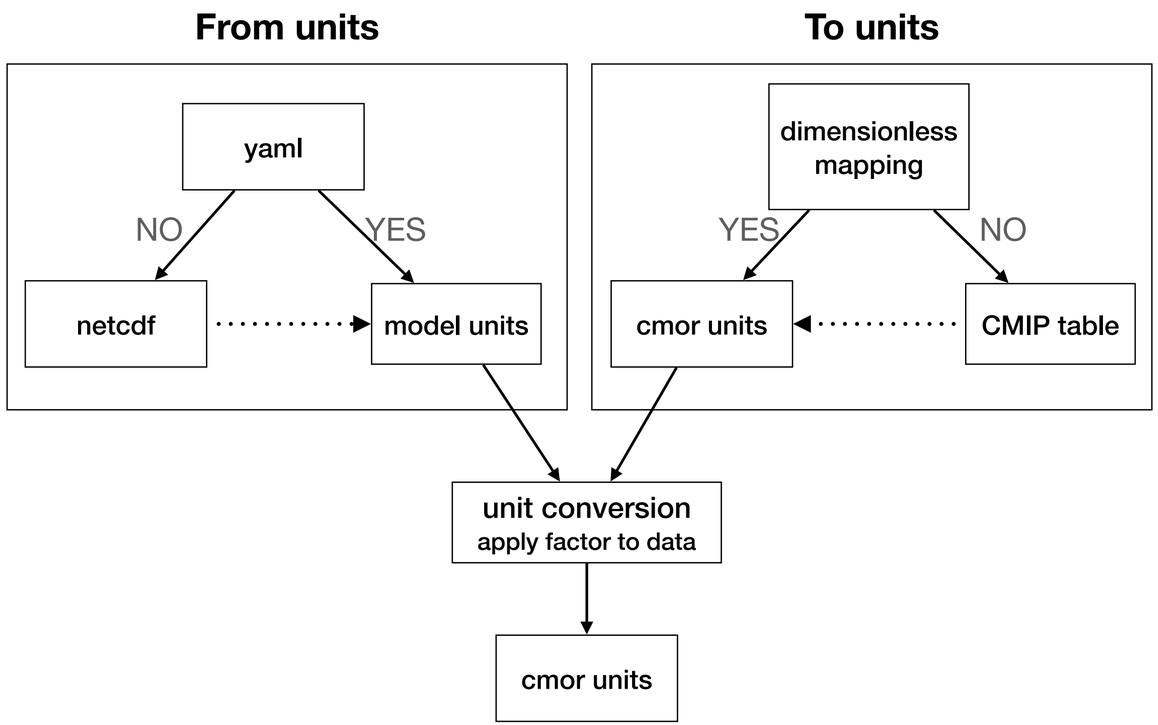
```
➜ grep -i "molC" $(ls -rtd logs/pymor-process* | tail -n 1 )
2025-03-13 09:06:37.158 | INFO     | pymor.units:handle_unit_conversion:148 - Converting units: (CO2f -> fgco2) mmolC/m2/d -> kg m-2 s-1
2025-03-13 09:06:37.158 | DEBUG    | pymor.units:handle_chemicals:67 - Chemical element Carbon detected in units mmolC/m2/d.
2025-03-13 09:06:37.158 | DEBUG    | pymor.units:handle_chemicals:68 - Registering definition: molC = 12.0107 * g
2025-03-13 09:06:37.470 | INFO     | pymor.units:handle_unit_conversion:148 - Converting units: (CO2f -> fgco2) mmolC/m2/d -> kg m-2 s-1
```

Units handling

- Wrong units in source netcdf files
  - Provide the correct units in yaml file (`model_units`)

- Dimensionless units in `cmor_units`
  - Provide mapping for dimensionless units
  - Pymor uses the mapping for unit conversion only.
  - It always writes `cmor_units` in the output netcdf file
  - Example:

```
> cat dimensionless_mappings.yaml
# cmor_variable_name:
#   cmor_unit_string: pint_friendly_SI_units
so:
  "0.001": g/kg
sos:
  "0.001": g/kg
intpp:
  # primary (organic carbon) production by phytoplankton
  "mol m-2 s-1": "molC m-2 s-1”
```



Time average

- Uses frequency string from table to determine time average function
- Time average methods:
  - Instantaneous (`da.resample(…).first()`)
  - Mean (`da.resample(…).mean()`)
  - Climatology (`da.groupby(…).mean('time')`)
- Set `` `adjust_timestamp: {first, middle, last, 14days, 0.3}` `` on Rule to offset the time stamp after averaging
  - keyword arguments: `` `first`, `middle`, `last` ``
  - Literal offset: `` `14days` (x days) ``
  - Floating number: `` `0.1 … 0.9` ``
  - Defaults to `` `first` ``

Custom step

- Custom functions must have the following function signature

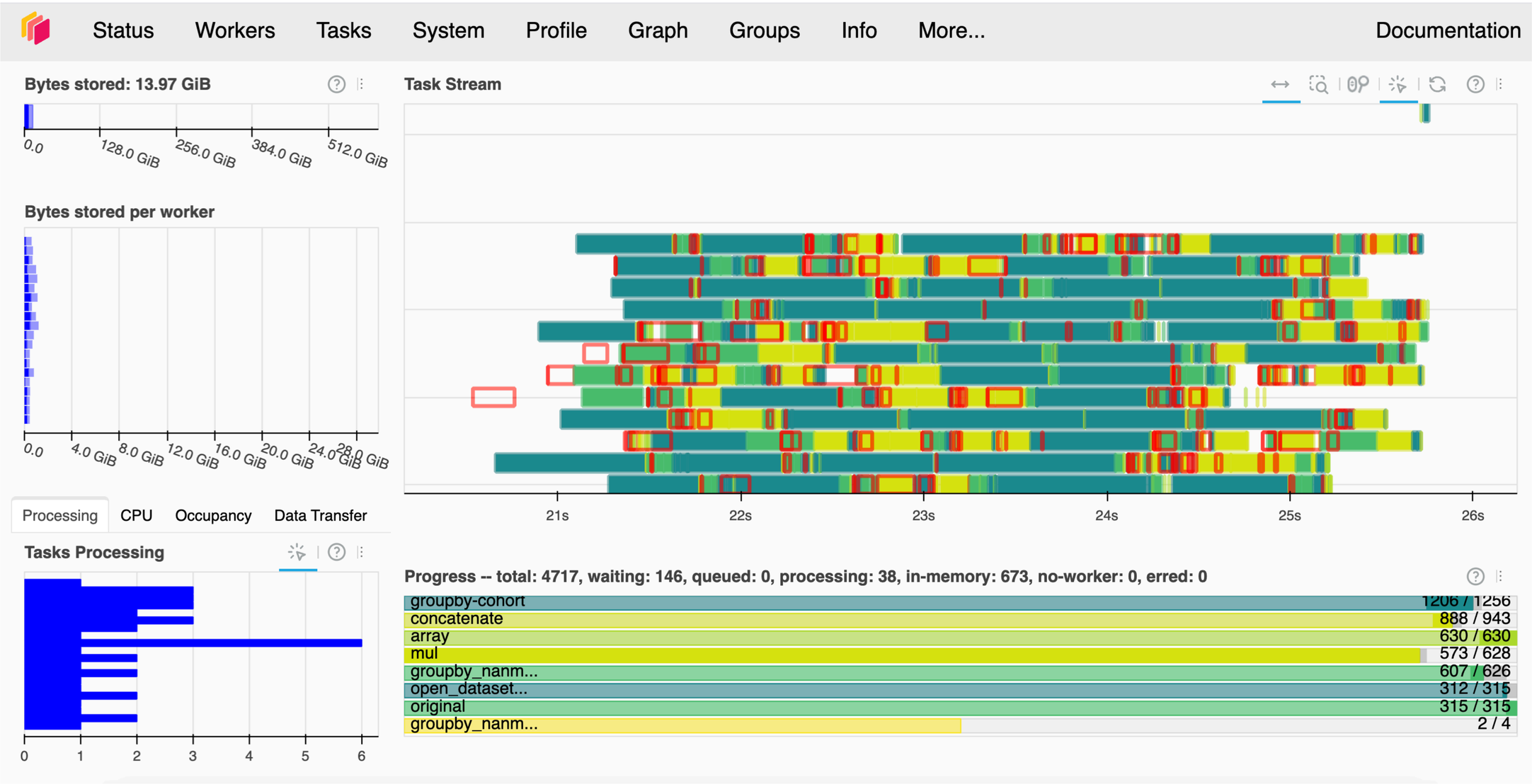
```
def my_custom_step(data, rule):
    # Do something with the data
    return data
```

```
Examples:
1. To compute derived variable from  other variables.
2. Re-meshing
3. Computing AMOC
```

- Include the custom function in the pipeline as follows

```
pipelines:
 - name: custom_pipeline
   steps:
      - script:///albedo/home/pgierz/Code/playground/my_custom_step.py:my_custom_step
```

Dask


Status
Workers
Tasks
System
Profile
Graph
Groups
Info
More...
Documentation
Bytes stored: 13.97 GiB
0.0
128.0 GiB
256.0 GiB
384.0 GiB
512.0 GiB
Bytes stored per worker
0.0
4.0 GiB
8.0 GiB
12.0 GiB
16.0 GiB
20.0 GiB
24.0 GiB
28.0 GiB
Processing
CPU
Occupancy
Data Transfer
Tasks Processing
0
1
2
3
4
5
6
Task Stream
21s
22s
23s
24s
25s
26s
Progress -- total: 4717, waiting: 146, queued: 0, processing: 38, in-memory: 673, no-worker: 0, erred: 0
groupby-cohort 1206 / 1256
concatenate 888 / 943
array 630 / 630
mul 573 / 628
groupby_nanm... 607 / 626
open_dataset... 312 / 315
original 315 / 315
groupby_nanm... 2 / 4

CMIP support

- So far, only CMIP6 support
- With placeholders for CMIP7 interface/data requirements
- Same Pymor interface with different backend CMIP versions

```python

class ControlledVocabularies(dict, metaclass=MetaFactory):
    @classmethod
    def from_directory(cls, directory: str) -> "ControlledVocabularies":
        """Create ControlledVocabularies from a directory of CV files"""
        raise NotImplementedError
```

```python

class CMIP6ControlledVocabularies(ControlledVocabularies):
    """Controlled vocabularies for CMIP6"""

    def __init__(self, json_files):
        """Create a new ControlledVocabularies object from a list of json files
```

```python
class CMIP7ControlledVocabularies(ControlledVocabularies):
    pass
```

Links

- GitHub: https://github.com/esm-tools/pymorize
- Doc: https://pymorize.readthedocs.io/
- HPC Group @ AWI: https://www.awi.de/en/science/special-groups/scientific-computing/high-performance-computing.html
- Mail: paul.gierz@awi.de, pavankumar.siligam@awi.de, miguel.andres-martinez@awi.de