CFGRIB: EASY AND EFFICIENT GRIB FILE ACCESS IN XARRAY

Alessandro Amici, B-Open, Rome

@alexamici

@alexamici

http://bopen.eu

Workshop on developing Python frameworks for earth
system sciences, 2018-10-30, ECMWF, Reading.

MOTIVATION

HERE AT ECMWF...

- ... we ❤ the GRIB format...
- ... and we ❤ Open Source...
- ... and we ❤ Python...
- ... but we were ☹ about GRIB support in Python

GOAL

We would love the GRIB format to be a first-class citizens in the Python numerical stack, with as good a support as netCDF!

ECMWF partnered with B-Open to make that happen.

DEVELOPMENT

REQUIREMENTS

- full GRIB support in *xarray*
  - gateway to the Python numerical stack: *Numpy*, *Matplotlib*, *Jupyter*, *Dask*, *Scipy*, *Pandas*, *Iris*, etc.
  - robust map to Unidata's *Common Data Model v4* with *CF-Conventions*
- delightful (!) install experience
  - full support of Python 3 and *PyPy*
  - major distribution channels: *PyPI*, *conda*, source

STATE OF THE ART

- *pygrib*, *pupygrib*, *ecCodes* - No CMD
- *PyNIO*
  - Pros: xarray backend, conda
  - Cons: partial CDM support, Python 2-only, no PyPI, read-only
- *Iris-grib*
  - Pros: xarray conversion, read-write, conda
  - Cons: Python 2-only, domain specific

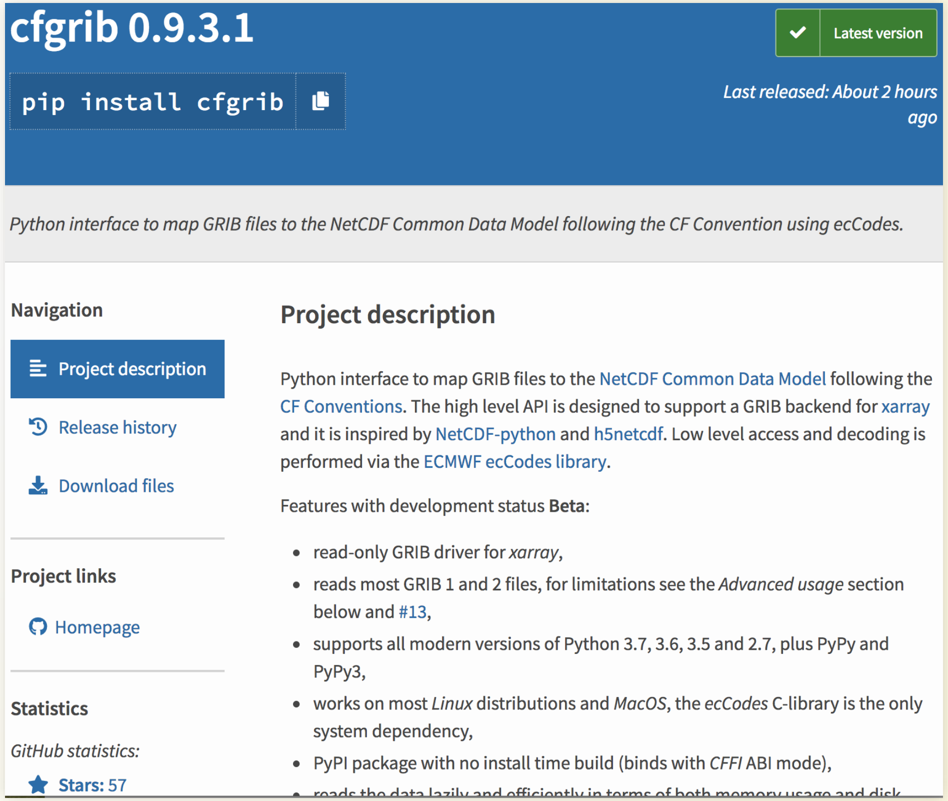
STORYLINE

- 2016-10: first prototype by ECMWF
- 2017-09: start of private *xarray-grib* by B-Open
- 2018-05: start of public *cfgrib* on GitHub
- 2018-07: first public **alpha** release of *cfgrib*
- 2018-10: *cfgrib* enters **beta**
- 2018-XX: *xarray* v0.11 will have a *cfgrib* backend

```
xr.open_dataset('data.grib', engine='cfgrib')
```

PRESENTING *CFGRIB*

- ecCodes bindings via CFFI for Python 3 and PyPy
- GRIB-level API: *FileStream*, *FileIndex* and *Message*
- CDM-level API: *Dataset* and *Variable*, inspired to *h5netcdf* and *netCDF4-Python*
- *xarray* read-only backend
- ... and more



cfgrib 0.9.3.1

```
pip install cfgrib
```

Last released: About 2 hours ago

Python interface to map GRIB files to the NetCDF Common Data Model following the CF Convention using ecCodes.

Project description

Python interface to map GRIB files to the NetCDF Common Data Model following the CF Conventions. The high level API is designed to support a GRIB backend for xarray and it is inspired by NetCDF-python and h5netcdf. Low level access and decoding is performed via the ECMWF ecCodes library.

Features with development status **Beta**:

- read-only GRIB driver for *xarray*,
- reads most GRIB 1 and 2 files, for limitations see the *Advanced usage* section below and #13,
- supports all modern versions of Python 3.7, 3.6, 3.5 and 2.7, plus PyPy and PyPy3,
- works on most *Linux* distributions and *MacOS*, the *ecCodes* C-library is the only system dependency,
- PyPI package with no install time build (binds with *CFFI* ABI mode),
- reads the data lazily and efficiently in terms of both memory usage and disk

USER JOURNEY

INSTALL *ECCODES* C-LIBRARY

With conda

```
$ conda install eccodes
```

On Ubuntu

```
$ sudo apt-get install libeccodes0
```

On MacOS with Homebrew

```
$ brew install eccodes
```

INSTALL *CFGRIB*

Install *cfgrib*

```
$ pip install cfgrib
```

Run *cfgrib* selfcheck

```
$ python -m cfgrib selfcheck
Found: ecCodes v2.7.0.
Your system is ready.
```

Install *xarray*

```
$ pip install xarray>=0.10.9
```

GRIB DATASET

```
>>> import cfgrib
>>> ds = cfgrib.open_dataset('era5-levels-members.grib')
>>> ds
<xarray.Dataset>
Dimensions:        (isobaricInhPa: 2, latitude: 61, longitude: 120, number: 10, time: 4
Coordinates:
  * number         (number) int64 0 1 2 3 4 5 6 7 8 9
  * time           (time) datetime64[ns] 2017-01-01 ... 2017-01-02T12:00:00
    step           timedelta64[ns] ...
  * isobaricInhPa  (isobaricInhPa) float64 850.0 500.0
  * latitude       (latitude) float64 90.0 87.0 84.0 81.0 ... -84.0 -87.0 -90.0
  * longitude      (longitude) float64 0.0 3.0 6.0 9.0 ... 351.0 354.0 357.0
    valid_time     (time) datetime64[ns] ...
Data variables:
    z              (number, time, isobaricInhPa, latitude, longitude) float32 ...
    t              (number, time, isobaricInhPa, latitude, longitude) float32 ...
Attributes:
    GRIB_edition:            1
    GRIB_centre:             ecmf
    GRIB_centreDescription:  European Centre for Medium-Range Weather Forecasts
    GRIB_subCentre:          0
    history:                 GRIB to CDM+CF via cfgrib-0.9.../ecCodes-2...
```

NAMING FROM ECCODES

- Attributes with the `GRIB_` prefix are *ecCodes* keys both coded and computed. Mostly namespace and edition independent keys
- Variable name is defined by *ecCodes*:
  - `GRIB_cfVarName` → variable name
- CF attributes are provided *ecCodes*:
  - `GRIB_name` → `long_name`,
  - `GRIB_units` → `units`
  - `GRIB_cfName` → `standard_name`

GRIB DATAARRAY

```
>>> ds.t
<xarray.DataArray 't' (number: 10, time: 4, isobaricInhPa: 2, latitude: 61, longitude:
[585600 values with dtype=float32]
Coordinates:
  * number         (number) int64 0 1 2 3 4 5 6 7 8 9
  * time           (time) datetime64[ns] 2017-01-01 ... 2017-01-02T12:00:00
    step           timedelta64[ns] ...
  * isobaricInhPa  (isobaricInhPa) float64 850.0 500.0
  * latitude       (latitude) float64 90.0 87.0 84.0 81.0 ... -84.0 -87.0 -90.0
  * longitude      (longitude) float64 0.0 3.0 6.0 9.0 ... 351.0 354.0 357.0
    valid_time     (time) datetime64[ns] ...
Attributes:
    GRIB_paramId:                             130
    GRIB_shortName:                           t
    GRIB_units:                               K
    GRIB_missingValue:                        9999
    GRIB_typeOfLevel:                         isobaricInhPa
    GRIB_gridType:                            regular_ll
    ...
    standard_name:                            air_temperature
    long_name:                                Temperature
    units:                                    K
```

GEOGRAPHIC COORDINATES

Computed by *ecCodes* based on `GRIB_gridType`: `regular_ll`, `regular_gg`, etc.

```
>>> ds.latitude
<xarray.DataArray 'latitude' (latitude: 61)>
array([ 90.,  87.,  ... -87., -90.])
Coordinates:
  * latitude  (latitude) float64 90.0 87.0 84.0 81.0 ... -81.0 -84.0 -87.0 -90.0
Attributes:
    units:          degrees_north
    standard_name:  latitude
    long_name:      latitude
>>> ds.longitude
<xarray.DataArray 'longitude' (longitude: 120)>
array([  0.,   3.,  ... 354., 357.])
Coordinates:
  * longitude  (longitude) float64 0.0 3.0 6.0 9.0 ... 348.0 351.0 354.0 357.0
Attributes:
    units:          degrees_east
    standard_name:  longitude
    long_name:      longitude
```

VERTICAL LEVEL COORDINATE

Variable name from *ecCodes* `GRIB_typeOfLevel`:

`isobaricInhPa`, `surface`, `hybrid`, etc.

```
>>> ds.isobaricInhPa
<xarray.DataArray 'isobaricInhPa' (isobaricInhPa: 2)>
array([850., 500.])
Coordinates:
  * isobaricInhPa  (isobaricInhPa) float64 850.0 500.0
Attributes:
    units:          hPa
    positive:       down
    standard_name:  air_pressure
    long_name:      pressure
```

EVERYTHING LOOKS PERFECT, RIGHT?

WRONG!

Very first bug report:

```
>>> ds = cfgrib.open_dataset('nam.t00z.awp21100.tm00.grib2')
Traceback (most recent call last):
  File "\<stdin\>", line 1, in <module>
  ...
  File ".../cfgrib/dataset.py", line 150, in enforce_unique_attributes
    raise ValueError("multiple values for unique attribute %r: %r" % (key, values))
ValueError: multiple values for unique attribute
    'typeOfLevel': ['hybrid', 'cloudBase', 'unknown', 'cloudTop']
```

THE DEVIL IS IN THE DETAILS

COMMON DATA MODEL

- *xarray* is based on the concept of hypercubes
- `xr.DataArray` is N-dimensional array
- Dimensions are labeled by 1D coordinates
- `xr.Dataset` is a container of data variables **with homogeneous coordinates**

GRIB DATA MODEL

- A GRIB *stream*, a file, is list of GRIB *messages*
- A GRIB *message* contains a single geographic *field* with `latitude`, `longitude`
- *Message* metadata (keys) can be regarded as additional coordinates: `time`, `level`, etc.
- *MARS* retrievals are typically nice hypercubes
- *Messages* in a *stream* are completely independent, **there's no guarantee**

GRIB IS A GENERIC CONTAINER

- North American Model (NAM) GRIB2
  - variable `gh` for `isobaricInhPa`, `cloudBase`, `cloudTop`, `maxWind` and `isothermZero`
- Global Forecast System (GFS) v4 GRIB2
  - variables `gh` and `clwmr` are defined on different values of `isobaricInhPa`

MESSAGE FILTERING

```
>>> cfgrib.open_dataset('nam.t00z.awp21100.tm00.grib2',
...     backend_kwargs=dict(filter_by_keys={'typeOfLevel': 'cloudTop'}))
<xarray.Dataset>
Dimensions:     (x: 93, y: 65)
Coordinates:
    time        datetime64[ns] ...
    step        timedelta64[ns] ...
    cloudTop    int64 ...
    latitude    (y, x) float64 ...
    longitude   (y, x) float64 ...
    valid_time  datetime64[ns] ...
Dimensions without coordinates: x, y
Data variables:
    pres        (y, x) float32 ...
    gh          (y, x) float32 ...
    t           (y, x) float32 ...
Attributes:
    GRIB_edition:            2
    GRIB_centre:             kwbc
    GRIB_centreDescription:  US National Weather Service - NCEP
    GRIB_subCentre:          0
    history:                 GRIB to CDM+CF via cfgrib-0.9.../ecCodes-2.8...
```

TO SUMMARISE

CFGRIB FEATURES IN BETA

- *xarray* backend starting with v0.11
- reads most GRIB 1 and 2 files,
- supports all modern versions of Python 3.7, 3.6, 3.5 and 2.7, plus PyPy and PyPy3,
- works on most *Linux* distributions and *MacOS*, *ecCodes* C-library is the only system dependency,
- you can `pip install cfgrib` with no compile,
- reads the data lazily and efficiently in terms of both memory usage and disk access.

CFGRIB WORK IN PROGRESS

- **Alpha** supports writing the index of a GRIB file to disk, to save a full-file scan on open,
- **Pre-Alpha** support to write carefully-crafted `xarray.Dataset`'s to a GRIB2 file.

CFGRIB LIMITATIONS

- no *conda* package, for now,
- *PyPI* binary package does not include *ecCodes*, for now,
- incomplete documentation, for now,
- no *Windows* support, for now,
- rely on *ecCodes* for the CF attributes of the data variables,
- rely on *ecCodes* for the `gridType` handling.

THE TEAM

- ECMWF
  - Stephan Siemen, Iain Russell and Baudouin Raoult
- B-Open
  - Alessandro Amici, Aureliana Barghini and Leonardo Barcaroli

THANK YOU!

Alessandro Amici, B-Open, Rome

@alexamici

@alexamici

http://bopen.eu

Slides:

https://gitpitch.com/alexamici/talks