CDO - advanced data operations

Uwe Schulzweida
Luis Kornblueh
Ralf Müller
Karl-Hermann Wieners
Oliver Heidmann

MPI Met

6. Dezember 2016

Max-Planck-Institut
für Meteorologie

Overview

Overview

no, thank you - I guess you know it already

So I will concentrate on central **and** new features:

- interesting options
- possibly unknown operators
- new operators
- scripting with Python/Ruby

Max-Planck-Institut
für Meteorologie

Overview

no, thank you - I guess you know it already

So I will concentrate on central **and** new features:

- interesting options
- possibly unknown operators
- new operators
- scripting with Python/Ruby

Let's have some fun!

Max-Planck-Institut
für Meteorologie

Main Feature: One Rule to Combine them all!

Shared Memory Parallelisation

- Smallest IO unit is a *record*: one horizontal field - like a GRIB record
- Output stream of right operator is input stream of left operator

```
cdo -output -selname,temp2 <ifile>
```

Main Feature: One Rule to Combine them all!

Shared Memory Parallelisation

- Smallest IO unit is a *record*: one horizontal field - like a GRIB record
- Output stream of right operator is input stream of left operator

```
cdo -output -selname,temp2 <ifile>
```

What's the benefit?

- Huge files can be processed as long as a single record fits into memory
- No need for temporary files - all is done in parallel!
- Other parallelisation techniques can be use on top or below: File splitting, OpenMP, multiprocessing

Highlights: Useful options - Part I

Run multiple OpenMP threads

`-P <threads>`

OpenMP is mostly used in horizontal interpolation, ensemble analysis, filtering and eofs:

```
cdo -P 8  remapcon (conservative)
cdo -P 16 genlaf   (largst ares fraction)
cdo -P 2  coffee   (not yet released)
```

Highlights: Useful options - Part I

Run multiple OpenMP threads

`-P <threads>`

OpenMP is mostly used in horizontal interpolation, ensemble analysis, filtering and eofs:

```
cdo -P 8  remapcon (conservative)
cdo -P 16 genlaf   (largst ares fraction)
cdo -P 2  coffee   (not yet released)
```

Set netcdf header size

`--hdr_pad <numberOfBytes>`

If the memory dedicated to data definitions is large enough, meta information can be changed *without* rewriting the data. *[netcdf only]*

Highlights: Useful options - Part I

Run multiple OpenMP threads

-P <threads>

OpenMP is mostly used in horizontal interpolation, ensemble analysis, filtering and eofs:

```
cdo -P 8  remapcon (conservative)
cdo -P 16 genlaf   (largst ares fraction)
cdo -P 2  coffee   (not yet released)
```

Set netcdf header size

--hdr_pad <numberOfBytes>

If the memory dedicated to data definitions is large enough, meta information can be changed *without* rewriting the data. *[netcdf only]*

Set output precision

-b <numberOfBits>

Possible values are I8/I16/I32/F32/F64 for nc/nc2/nc4/nc4c

P1 - P24 for grb/grb2

Highlights: Useful options - Part II

colors

`-C|--color`

Get colourful output with this option.

Highlights: Useful options - Part II

colors

`-C|--color`

Get colourful output with this option.

keep calm

`-s|--silent`

Suppress all warnings

Max-Planck-Institut
für Meteorologie

Highlights: Useful options - Part II

colors

`-C|--color`

Get colourful output with this option.

keep calm

`-s|--silent`

Suppress all warnings

get rid of dimensions

`--reduce_dim`

Remove dimensions with length 1: time, lon, lat, lev

Max-Planck-Institut
für Meteorologie

Highlights - fine tuned data conversion

How to convert meta data of variables in a single step

setpartabn and *setpartabp* allow meta data transformations based on a fortran namelist syntax:

```
&parameter
  name            = topo
  out_name        = topography
  standard_name   = surface_height
  units           = "cm"
/
```

Other transformation keys are: long_name, missing_value, type, valid_min, factor, delete, convert, ...

Arbitrary attributes are supported with upcoming release cdo-1.8.0

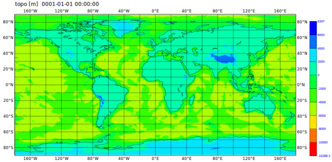
Highlights - CMORlite

cmorlite operator - upcoming cdo-1.8.0

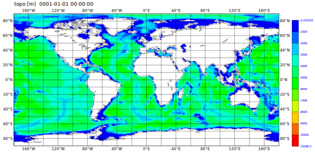
Similar to setpartabn, but with JSON input format based on CMOR3 or CMOR2 tables.

```
$ cdo partab -selname,tauu iconAtm.nc
&parameter
  name=tauu
  param=17.2.0
  standard_name=u_stress
  long_name="u-momentum flux at the surface (time mean)"
  units="N m-2"
/
$ cdo -partab -cmorlite,CMIP6_Amon.json,convert -selname,
    tauu iconAtm.nc tauu_cmorized.nc
&parameter
  name=tauu
  param=17.2.0
  standard_name=surface_downward_eastward_stress
  long_name="Surface Downward Eastward Wind Stress"
  units="Pa"
/
```

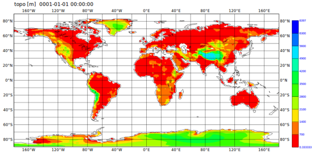
Highlights: built-in topography with *topo* operator


topo [m] 0001-01-01 00:00:00
160°W 120°W 80°W 40°W 0°E 40°E 80°E 120°E 160°E
80°N 60°N 40°N 20°N 0°N 20°S 40°S 60°S 80°S
6397
6000
4000
2000
0
-2000
-4000
-6000
-8000
-10000
-10288.3

```
cdo -topo topo.grb
```


topo [m] 0001-01-01 00:00:00
160°W 120°W 80°W 40°W 0°E 40°E 80°E 120°E 160°E
80°N 60°N 40°N 20°N 0°N 20°S 40°S 60°S 80°S
-0.333333
-1000
-2000
-3000
-4000
-5000
-6000
-7000
-8000
-9000
-10000
-10288.3

```
cdo -setrtomiss,0,10000
        -topo topo_ocean.grb
```


topo [m] 0001-01-01 00:00:00
160°W 120°W 80°W 40°W 0°E 40°E 80°E 120°E 160°E
80°N 60°N 40°N 20°N 0°N 20°S 40°S 60°S 80°S
6397
6300
5600
4900
4200
3500
2800
2100
1400
700
0.333333

```
cdo -setrtimiss,-20000,0
        -topo topo_land.grb
```

See also *temp*, *const*, *random* or *stdatm*.

Max-Planck-Institut für Meteorologie

Highlights - formulars with *expr*

More then + and -

```
cdo -f nc \
  -expr,'P = 1013.25 * exp(-1.60276977072154*log((exp(topo
      /10000.0)*213.15+75.0)/288.15))' \
  -topo surface_pressure.nc
```

Highlights - formulars with *expr*

More then + and -

```
cdo -f nc \
  -expr,’P = 1013.25 * exp(-1.60276977072154*log((exp(topo
      /10000.0)*213.15+75.0)/288.15))’ \
  -topo surface_pressure.nc
```

Mask valued expressions

== ,!= ,< ,<= ,>= ,> ,<=> ,&& ,|| ,?: (ternary operator)

```
cdo -f nc \
  -expr,’topo = ((topo >= 0.0)) ? topo : (topo/0.0)’  \
  -topo orog.nc
```

Max-Planck-Institut
für Meteorologie

Highlights - formulars with *expr*

expr vs. *aexpr*

aexpr performs a copy on all input fields to the output stream and appends the computation results to it. *expr* writes computed fields only.

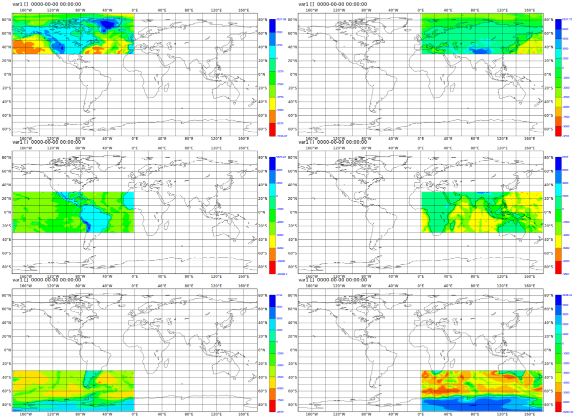
And what if formulars are getting lengthy?

exprf and *aexprf* accept textfile names as arguments from where the formulars will be read in. See here or here for more.

Highlights: Split the grid with *distgrid* - *collgrid*

Break your regular grid into $n \times m$ parts

```
cdo -distgrid,2,3 -topo topo_splitted
```



Put your pieces together with

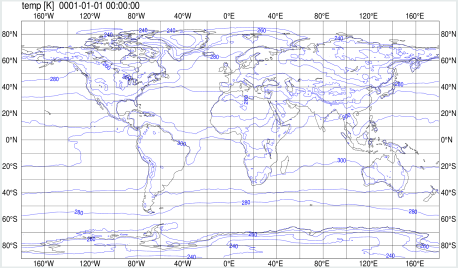
```
cdo -collgrid topo_splitted*grb collectedtopo.grb
```

ck-Institut
orologie

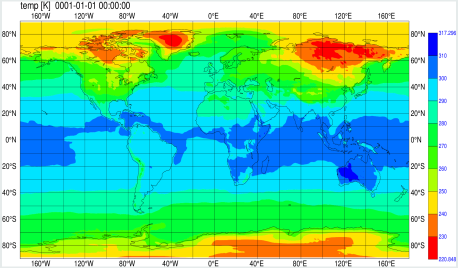
Highlights: Magics++ for plotting ... Watch out PIXAR!

Possible plot types

- *contour*



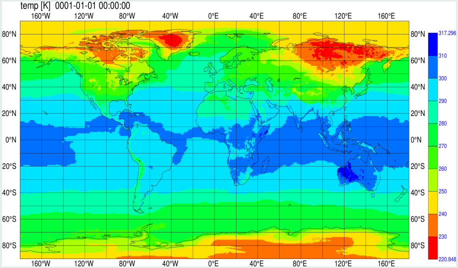
- *shaded*



Highlights: Magics++ for plotting ... Watch out PIXAR!

Possible plot types

- coloured cells: *grfill*

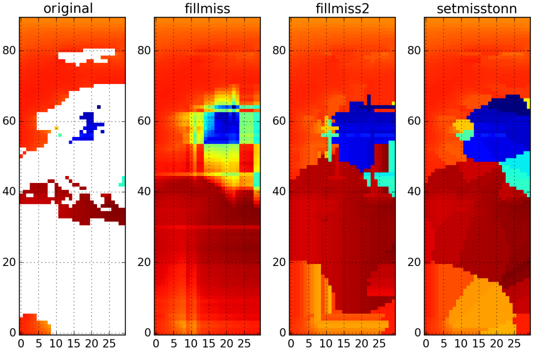


- more: line plots, vectors, animations, output formats: png, svg, ps, pdf, . . . more examples

Fill missing values

How to overwrite missing data with something reasonable

Model initial data for ocean salinity is on low resolution, usually 1deg.

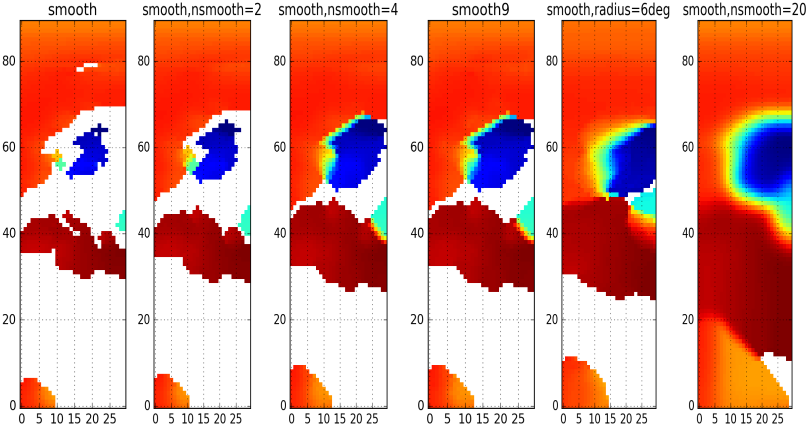


For higher resolution runs, a simple interpolation could lead to wrong values in the baltic see. Nearest-neighbor interpolation does the trick.

Smooth'em all

Even more control over how to fill missing values

Interpolation based on neighborhood or distance for arbitrary grids



Check for more options with: `cdo -h smooth`

Max-Planck-Institut für Meteorologie

Scripting with Ruby/Python

cdo.{rb,py}

- is a *smart* caller of a CDO binary (with all the pros and cons)
- doesn't need to be re-installed for a new CDO version
- directly bridges your data to the scientific package in Ruby/Python

Max-Planck-Institut
für Meteorologie

Scripting with Ruby/Python

cdo.{rb,py}

- is a *smart* caller of a CDO binary (with all the pros and cons)
- doesn't need to be re-installed for a new CDO version
- directly bridges your data to the scientific package in Ruby/Python
- ! isn't a shared library, which keeps everything in memory
- ! doesn't allow write access to files via the numpy or masked arrays

Max-Planck-Institut
für Meteorologie

Scripting with Ruby/Python

cdo.{rb,py}

- is a *smart* caller of a CDO binary (with all the pros and cons)
- doesn't need to be re-installed for a new CDO version
- directly bridges your data to the scientific package in Ruby/Python
- ! isn't a shared library, which keeps everything in memory
- ! doesn't allow write access to files via the numpy or masked arrays

homepage:

`https://code.zmaw.de/projects/cdo/wiki/Cdo{rbpy}`

or directly join development at

`https://github.com/Try2Code/cdo-bindings`

Max-Planck-Institut
für Meteorologie

Usage: Basic - Python 2.7/3.x

Interface examples

```
from cdo import *
cdo = Cdo()

# concatenate list of files, relative time axis
cdo.cat(input = ’ ’.join(ofiles),
        output = ofile,
        options = ’-r’)
# vertical interpolation
cdo.intlevel(100,200,500,1000,
             input=’Temperatures_L199.grb’,
             output=’TempOnTargetLevels.grb’)
# zonal mean after interpolation in nc4 classic format
cdo.zonmean(input = "-remapbil,r1400x720 "+myData,
            output = zonmeanFile,
            options = ’-P 8 -f nc4c’)
```

Usage: Advanced

return numpy and masked arrays

```
cdo.div(input='salinity.nc landSeaMask.nc',
    returnArray='S')
cdo.copy(input='-div salinity.grb landSeaMask.grb',
    returnMaArray='S', options='-f nc')
```

See this interactive session. Please share yours!

smooth operator tests

Usage: Advanced

return numpy and masked arrays

```
cdo.div(input=’salinity.nc landSeaMask.nc’,
    returnArray=’S’)
cdo.copy(input=’-div salinity.grb landSeaMask.grb’,
    returnMaArray=’S’, options=’-f nc’)
```

See this interactive session. Please share yours!

smooth operator tests

get cdf handles: access to all variables

```
cdf   = cdo.fldmin(input=ifile,returnCdf=true)
tData = cdf.variables[’T’][:]
```

Max-Planck-Institut
für Meteorologie

Usage: Advanced

return numpy and masked arrays

```
cdo.div(input=’salinity.nc landSeaMask.nc’,
    returnArray=’S’)
cdo.copy(input=’-div salinity.grb landSeaMask.grb’,
    returnMaArray=’S’, options=’-f nc’)
```

See this interactive session. Please share yours!

smooth operator tests

get cdf handles: access to all variables

```
cdf   = cdo.fldmin(input=ifile,returnCdf=true)
tData = cdf.variables[’T’][:]
```

conditional output: no execution if output file is present

```
cdo.forceOutput = False #or
cdo.operator(.....,force=False)
```

Usage: Parallelism with Python

Beyond the shell

```
def grepYear(ifiles,year):
  yearFiles = []
  for ifile in ifiles:
    if (year in cdo.showyear(input = ifile).split()):
        yearFiles.append(ifile)
  cdo.cat(input = ' '.join(yearFiles),
          output = yearFile)
```

```
pool      = multiprocessing.Pool(8)
yearFiles = []
for year,files in filesOfYears.iteritems():
    yearFile = pool.apply_async(grepYear, [files,str(year)])
    yearFiles.append([year,yearFile,yearMeanFile])

pool.close()
pool.join()
```

CDO's Future

Our Plans

- Have changed to C++ for using more advanced high-level data structure and algorithms for future developments.
- provide operators as library for use in many more apps (direct Python or Ruby usage, maybe even models)
- extend for user defined operators by a plugin system
- additional optimization (OpenMP, OpenACC, OpenCL) and parallelization techniques based on distributed memory (HPX, MPI)

Max-Planck-Institut
für Meteorologie

CDO's Future

Our Plans

- Have changed to C++ for using more advanced high-level data structure and algorithms for future developments.
- provide operators as library for use in many more apps (direct Python or Ruby usage, maybe even models)
- extend for user defined operators by a plugin system
- additional optimization (OpenMP, OpenACC, OpenCL) and parallelization techniques based on distributed memory (HPX, MPI)

What feature do YOU need most?

Max-Planck-Institut für Meteorologie

Play the wildcard ... with files

Problem

How to keep the chaining of operators working, when their number of input streams is abitrary? - Polish notation only works for operators with fixed arity

Might not be a problem for operators like *info* or *copy*, but concatenation (*cat*) and merging (*merge*/*mergetime*) would create large temporary data

Max-Planck-Institut für Meteorologie

Play the wildcard ... with files

Problem

How to keep the chaining of operators working, when their number of input streams is abitrary? - Polish notation only works for operators with fixed arity

Might not be a problem for operators like *info* or *copy*, but concatenation (*cat*) and merging (*merge*/*mergetime*) would create large temporary data

... let CDO do the wildcard evaluation

Given single quoted wildcard as input stream, CDO evaluates it into a fixed length list

```
    cdo -timmean -cat ’exp004_201?_global.nc*’ exp004.nc
```

Max-Planck-Institut
für Meteorologie

Play the wildcard ... with variables

Problem

How to select collections of data without explicitly given names or parameters

Play the wildcard ... with variables

Problem

How to select collections of data without explicitly given names or parameters

... use *select*

CDO's *select* operator accepts wildcards for the 'name' and 'param' key

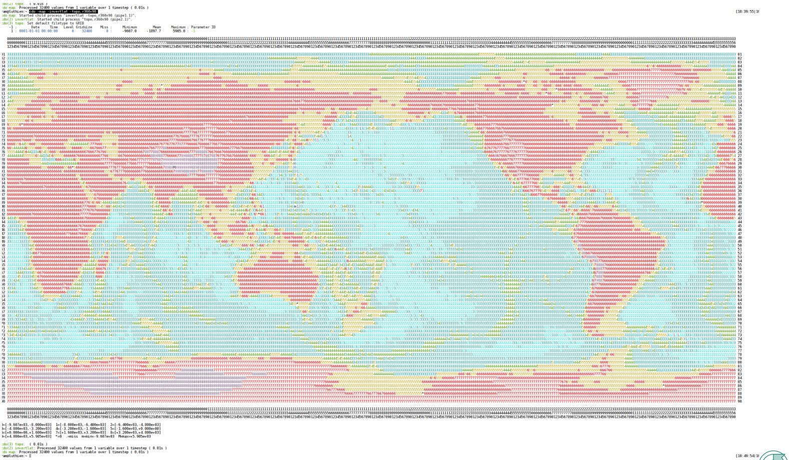
```
cdo -select,'name=s*' $ifile $ofile
cdo -select,'param=1.?.0' $ifile $ofile
```

Max-Planck-Institut
für Meteorologie

Another hidden gem: *map*

Visualization, a terminal-based approach:

```
cdo -map -invertlat -topo,r360x90
```



```
:do(2) topo:  ( 0.01s )
:do map: Processed 32400 values from 1 variable over 1 timestep ( 0.01s )
ram@luthien:~ cdo -map -invertlat -topo,r360x90                                   [10:39:55]
:do map: Started child process "invertlat -topo,r360x90 (pipe1.1)".
:do(2) invertlat: Started child process "topo,r360x90 (pipe2.1)".
:do(3) topo: Set default filetype to GRIB
   -1 :       Date     Time   Level Gridsize    Miss :     Minimum        Mean     Maximum : Parameter ID
    1 : 0001-01-01 00:00:00       0    32400       0 :     -9607.0     -1897.7      5905.0 : -1
```

```
)=[-9.607e+03,-8.000e+03]  1=[-8.000e+03,-6.400e+03]  2=[-6.400e+03,-4.800e+03]
3=[-4.800e+03,-3.200e+03]  4=[-3.200e+03,-1.600e+03]  5=[-1.600e+03,+0.000e+00]
6=[+0.000e+00,+1.600e+03]  7=[+1.600e+03,+3.200e+03]  8=[+3.200e+03,+4.800e+03]
9=[+4.800e+03,+5.905e+03]  *=0  .=miss  m=min=-9.607e+03  M=max=+5.905e+03

:do(3) topo:  ( 0.01s )
:do(2) invertlat: Processed 32400 values from 1 variable over 1 timestep ( 0.01s )
:do map: Processed 32400 values from 1 variable over 1 timestep ( 0.01s )
ram@luthien:~                                                                     [10:49:54]
```