cdb_query Documentation

Release 1.9.9.6

Freéec Laliberté,e, Paul J. Kushner

Apr 09, 2017

Contents

Contents:

CHAPTER 1

Introduction

This code is intended to provide an interface between the CMIP5/CORDEX Data Reference Syntax (hereafter DRS) for a local repository of CMIP5/CORDEX data, the Earth System Grid Federation (hereafter ESGF) search engine for the efficient, robust and timely computation of climate diagnostics.

The cdb_query package contains separate command sets for CMIP5 and CORDEX archives. The standard workflow for both packages is similar:

1. A *ask* command searches the archive for specific variables and user specified date information. Its output is used by the "validate" command.
2. The *validate* command refines the "ask" step to check on full availability of specified data over the specified date information. Its output contains full metadata for all the available datasets along with soft links to the data. Its output is used by the *reduce_soft_links*, *download_opendap* and *download_files* commands.
3. The *reduce_soft_links* command can apply a script onto the output of *validate* to enable subsetting.
4. The *download_opendap* or *download_files* commands then retrieves the data using the OPeNDAP protocol or a wget command, respectively.
5. The data retrieved from *download_files* is structured in a way that can be accessed by subsequent *ask* and other commands.
6. The data retrieved from *download_opendap* keep the same structure as *validate*. It can then be processed using the command *reduce*.

The source code for this project is available on github: https://github.com/laliberte/cdb_query

CHAPTER 2

Installation instructions

Attention: The long list of requirements may seem daunting but on many standard linux distributions they should be relatively easy to fulfill. Here it is assumed that Linux is used.

This package requires:

Core requirements

- Python 2.7.x.
- A recent version of the netCDF4 library.
- ESGF credentials.

Before proceeding further, are you willing to use a 3rd party Linux Distribution (Free for Academic Use)?

- If yes, follow the instructions install-distro
- If no, follow the instructions install-source

At this point in the installation, it is assumed that you have a working python distribution with the netCDF4 python package compiled.

Installing this package: *cdb_query*

This package can be installed with `pip`:

```
$ pip install cdb_query
```

Warning: If you are using a virtual environment, you must always `source $HOME/python/bin/activate` BEFORE using `cdb_query`. If you are using Anaconda, you must activate it (see install-distro).

Obtaining ESGF credentials

This package allows you to obtain and manage ESGF credentials transparently. The only actions a user should take is

1. Register at https://pcmdi.llnl.gov/user/add/?next=https://pcmdi.llnl.gov/projects/esgf-llnl/. When registering, you will create a *password* and receive an *openid*.
2. Go to https://pcmdi.llnl.gov/esg-orp/home.htm and login with your *openid* using your *password*.
3. Go to https://pcmdi.llnl.gov/esg-orp/registration-request.htm?resource=http%3A%2F%2Fpcmdi.llnl.gov%2Fthredds%2FfileServer%2Fesg_testroot%2Fregister%2Fcmip5_research.nc. Your browser will likely ask you to trust some certificates. CLICK *ALWAYS ALLOW*. If you don't do this, it won't work.
4. Register for *CMIP5 Research*. You do not need to download data.
5. If using https://ceda.ac.uk instead of https://pcmdi.llnl.gov, your *username* must be passed along your *openid* when using *cdb_query*.

Note: Previously the authentication would use local certificates. This option is still available but will be deprecated in the future.

Secondary tools used in the recipes

netCDF Operators (NCO)

Some of the recipes make use of *NCO*. These recipes were tested using version 4.4.3 linked against the netcdf libraries built from *anaconda* and from install-source. Please consult the project's webpage for information on how to install: http://nco.sourceforge.net/.

These recipes were tested using the *NCO* built using the BASH script found in *BASH script to compile and install NCO on CentOS*

Climate Data Operators (CDO)

The netCDF4 files generated by *cdb_query* are not compatible with *CDO*. *NCO* can be used to extract variables and remove the hierarchical structure. The retrieved data will then be compatible with *CDO*. With all the installed libraries, *CDO* is relatively easy to install.

JASPER

You will need to first install *jasper*:

```
$ wget http://www.ece.uvic.ca/~frodo/jasper/software/jasper-1.900.1.zip
$ unzip jasper-1.900.1.zip
$ cd jasper-1.900.1
$ ./configure --with-pic --prefix=$HOME/local/jasper-1.900.1
```

```
$ make
$ make install
```

PROJ

Next, you will need *proj*:

```
$ wget http://download.osgeo.org/proj/proj-4.8.0.tar.gz
$ tar xvfz proj-4.8.0.tar.gz
$ cd proj-4.8.0
$ ./configure --without-jni --prefix=$HOME/local/proj-4.8.0
$ make check
$ make install
```

CDO

Finally, you are ready to install *CDO*:

```
$ wget --no-check-certificate https://code.zmaw.de/attachments/download/7220/cdo-1.6.
↪3.tar.gz
$ tar xvfz cdo-1.6.3.tar.gz
$ cd cdo-1.6.3
$ ./configure --prefix=$HOME/local/cdo-1.6.3 \
              --with-proj=$HOME/local/proj-4.8.0 \
              --with-jasper=$HOME/local/jasper-1.900.1 \
              --with-netcdf=$HOME/anaconda/  \
              --with-hdf5=$HOME/anaconda/ \
              --with-zlib=$HOME/anaconda/ \
              -enable-cgribex=no CFLAGS=-DHAVE_LIBNC_DAP
$ make
$ make install
```

This installation installs *CDO* in `$HOME/local/cdo-1.6.3/bin` and this directory should be added to your path.

You can check that everything was done ok:

```
$ cdo -V
Climate Data Operators version 1.6.3 (http://code.zmaw.de/projects/cdo)
Compiler: gcc -std=gnu99 -DHAVE_LIBNC_DAP -pthread
version: gcc (GCC) 4.1.2 20080704 (Red Hat 4.1.2-54)
Compiled: (x86_64-unknown-linux-gnu)
Features: PTHREADS NC4 OPeNDAP SZ Z JASPER UDUNITS2 PROJ.4
Libraries: proj/4.8
Filetypes: srv ext ieg grb grb2 nc nc2 nc4 nc4c
CDI library version : 1.6.3
GRIB_API library version : 1.11.0
netCDF library version : 4.3.1-rc2
HDF5 library version : 1.8.11
SERVICE library version : 1.3.1
EXTRA library version : 1.3.1
IEG library version : 1.3.1
FILE library version : 1.8.2
```

The *Features* line indicates that netCDF4 files are accepted, OPeNDAP links can be read and that compressed variables can be created (SZ, Z).

CHAPTER 3

Recipes

Once installed, this package should make the command line tool `cdb_query` visible to the user's path. This is typically the case for common python installations.

The command `cdb_query CMIP5` contains several commands:

- `cdb_query CMIP5 ask`, searches the CMIP5 archive and produces a file with pointers to the data.
- `cdb_query CMIP5 validate` can be used to find all the experiments that have all of the avialalble years. This command outputs a file that points to data for every month of the requested period.
- `cdb_query CMIP5 download_files` and `cdb_query CMIP5 download_opendap`, reads the output from `cdb_query CMIP5 validate`, as an input and returns a **single** path to file. This makes it easy to retrieve data from simple scripts.

Hint: The variable descriptions (time_frequency, realm, cmor_table, ...) for the CMIP5 can be found in files http://cmip-pcmdi.llnl.gov/cmip5/docs/standard_output.pdf, http://cmip-pcmdi.llnl.gov/cmip5/docs/standard_output.xls.

1. Retrieving surface temperature for ONDJF (CMIP5)

Hint: Don't forget to use the extensive command-line helps `cdb_query -h`, `cdb_query CMIP5 -h`, etc.

Discovering the data

The script is run using:

```
$ cdb_query CMIP5 ask \
                      --ask_month=1,2,10,11,12 \
                      --ask_var=tas:day-atmos-day,orog:fx-atmos-fx \
                      --ask_experiment=amip:1979-2004 \
```

```
                      --model=CanAM4 --model=CCSM4 --model=GISS-E2-R --model=MRI-
↪CGCM3 \
                      --num_procs=10 \
                      tas_ONDJF_pointers.nc
This is a list of simulations that COULD satisfy the query:
NCAR,CCSM4,r7i1p1,amip
NCAR,CCSM4,r2i1p1,amip
NCAR,CCSM4,r1i1p1,amip
NCAR,CCSM4,r4i1p1,amip
NCAR,CCSM4,r3i1p1,amip
NCAR,CCSM4,r5i1p1,amip
CCCMA,CanAM4,r2i1p1,amip
CCCMA,CanAM4,r1i1p1,amip
CCCMA,CanAM4,r4i1p1,amip
CCCMA,CanAM4,r3i1p1,amip
MRI,MRI-CGCM3,r4i1p2,amip
MRI,MRI-CGCM3,r2i1p1,amip
MRI,MRI-CGCM3,r1i1p1,amip
MRI,MRI-CGCM3,r3i1p1,amip
NASA-GISS,GISS-E2-R,r6i1p1,amip
NASA-GISS,GISS-E2-R,r6i1p3,amip
cdb_query will now attempt to confirm that these simulations have all the requested␣
↪variables.
This can take some time. Please abort if there are not enough simulations for your␣
↪needs.
```

Obtaining the tentative list of simulations should be very quick (a few seconds to a minute) but confirming that these simulations have all the requested variables should take a few minutes, depending on your connection to the ESGF node. It returns a self-descriptive netCDF4 file with pointers to the data. The `num_procs` flag substantially speeds up the discovery, and comes with a very small price to the user.

Hint: Try looking at the resulting netCDF file using `ncdump`:

```
$ ncdump -h tas_ONDJF_pointers.nc
```

As you can see, it generates a hierarchical netCDF4 file. `cdb_query CMIP5 list_fields` offer a tool to query this file.

Querying the discovered data

For example, if we want to know how many different simulations were made available, we would run

```
$ cdb_query CMIP5 list_fields -f institute -f model -f ensemble tas_ONDJF_pointers.nc
CCCMA,CanAM4,r0i0p0
CCCMA,CanAM4,r1i1p1
CCCMA,CanAM4,r2i1p1
CCCMA,CanAM4,r3i1p1
CCCMA,CanAM4,r4i1p1
MRI,MRI-CGCM3,r0i0p0
MRI,MRI-CGCM3,r1i1p1
MRI,MRI-CGCM3,r2i1p1
MRI,MRI-CGCM3,r3i1p1
NCAR,CCSM4,r0i0p0
NCAR,CCSM4,r1i1p1
```

```
NCAR,CCSM4,r2i1p1
NCAR,CCSM4,r3i1p1
NCAR,CCSM4,r4i1p1
NCAR,CCSM4,r5i1p1
NCAR,CCSM4,r7i1p1
```

This test was run on July 2nd, 2016 and these results represent the data presented by the ESGF node on that day. The r0i0p0 ensemble name is the ensemble associated with fixed (time_frequency=fx) variables and its presence suggests that these three institutes have provided the requested orog variable. These results also indicate that models CanAM4, MRI-CGCM3 and CCSM4 provided 4, 3 and 6 simulations,respectively.

If this list of models is satisfying, we next check the paths

```
$ cdb_query CMIP5 list_fields -f path tas_ONDJF_pointers.nc
...
http://esgf-data1.ceda.ac.uk/thredds/fileServer/esg_dataroot/cmip5/output1/NCAR/CCSM4/
↪amip/fx/atmos/fx/r0i0p0/v20130312/orog/orog_fx_CCSM4_amip_
r0i0p0.
↪nc|SHA256|87b29a7d2731e6b028d81b07edbe84c3f06e1321986401482f8c5d76d5361516|2b43ce02-
↪7124-40bd-8ae4-0961e399e9ec
http://esgf-data1.diasjp.net/thredds/fileServer/esg_dataroot/cmip5/output1/MRI/MRI-
↪CGCM3/amip/day/atmos/day/r1i1p1/v20120701/tas/tas_day_MRI-CGCM3_am
ip_r1i1p1_19790101-19881231.
↪nc|SHA256|804f3325a2b0e29bad14e5773a7216c2893a5200fba62ffa83db992b9765b283|e95dd229-
↪1e36-4e8a-9e50-8797e2a136a2
...
```

We consider the first path. It is constituted of two parts. The first part begins with `http://esgf-data1.ceda.ac.uk/...` and ends a the vertical line. This is a *wget* link. The second part, separated by vertical lines, are the checksum typw, checksum and tracking id, respectively. The checksum is as published on the EGSF website. The file found at the other end of the *wget* link can be expected to have the same checksum.

The string that precedes `/thredds/...` in the *wget* link is the *data node*. Here, we have two data nodes: `http://esgf-data1.ceda.ac.uk` and `http://esgf-data1.diasjp.net`. Retrieving two files from two different data nodes at the same time should not hinder the transfer of one another.

Hint: The command `cdb_query CMIP5 ask` does not guarantee that the simulations found satisfy ALL the requested criteria.

Validating the set of simulations

> **Warning:** From now on it is assumed that the user has installed properly resgistered with the ESGF. Using the `--openid` option combined with an ESGF account takes care of this. The first time this function is used, it might fail and ask you to register your kind of user. This has to be done only once.

To narrow down our results to the simulations that satisfy ALL the requested criteria, we can use

```
$ cdb_query CMIP5 validate \
            --openid=$OPENID \
            --Xdata_node=http://esgf2.dkrz.de \
            --num_procs=10 \
```

```
            tas_ONDJF_pointers.nc \
            tas_ONDJF_pointers.validate.nc
```

Here, we are removing data node `http://esgf2.dkrz.de` from the validation because on this data node, data sits on a tape archive and it can be very slow to recover it.

To output now has a time axis for each variable (except fx). It links every time index to a time index in a UNIQUE file (remote or local). Try looking at the resulting netCDF file using `ncdump`:

```
$ ncdump -h tas_ONDJF_pointers.validate.nc
```

Again, this file can be queried for simulations:

```
$ cdb_query CMIP5 list_fields -f institute -f model -f ensemble tas_ONDJF_pointers.
↪validate.nc
CCCMA,CanAM4,r0i0p0
CCCMA,CanAM4,r1i1p1
CCCMA,CanAM4,r2i1p1
CCCMA,CanAM4,r3i1p1
CCCMA,CanAM4,r4i1p1
MRI,MRI-CGCM3,r0i0p0
MRI,MRI-CGCM3,r1i1p1
MRI,MRI-CGCM3,r2i1p1
MRI,MRI-CGCM3,r3i1p1
NCAR,CCSM4,r0i0p0
NCAR,CCSM4,r1i1p1
NCAR,CCSM4,r2i1p1
NCAR,CCSM4,r3i1p1
NCAR,CCSM4,r4i1p1
NCAR,CCSM4,r5i1p1
NCAR,CCSM4,r7i1p1
```

Retrieving the data: *wget*

cdb_query CMIP5 includes built-in functionality for retrieving the paths. It is used as follows

```
$ cdb_query CMIP5 download_files \
                --download_all_files \
                --openid=$OPENID \
                --out_download_dir=./in/CMIP5/ \
                tas_ONDJF_pointers.validate.nc \
                tas_ONDJF_pointers.validate.downloaded.nc
```

It downloads the paths listed in `tas_ONDJF_pointers.validate.nc` to `./in/CMIP5/` and records the soft links to the local data in `tas_ONDJF_pointers.validate.downloaded.nc`.

Warning: The retrieved files are structured with the CMIP5 DRS. It is good practice not to change this directory structure. If the structure is kept then `cdb_query CMIP5 ask` will recognize the retrieved files as local if they were retrieved to a directory listed in the `Search_path`.

The downloaded paths are now discoverable by `cdb_query CMIP5 ask`.

Retrieving the data: *OPeNDAP*

cdb_query CMIP5 includes built-in functionality for retrieving a subset of the data.

To retrieve the first month of daily data:

```
$ cdb_query CMIP5 download_opendap \
                        --year=1979 \
                        --month=1 \
                        --openid=$OPENID \
                        tas_ONDJF_pointers.validate.nc \
                        tas_ONDJF_pointers.validate.197901.retrieved.nc
```

The file `tas_ONDJF_pointers.validate.197901.retrieved.nc` should now contain the first thirty days for all experiments! To check the daily surface temperature in the amip experiment from simulation NCAR,CCSM4,r1i1p1 *ncview* (if installed):

```
$ ncks -G : -g /NCAR/CCSM4/amip/day/atmos/day/r1i1p1/tas \
                tas_ONDJF_pointers.validate.197901.retrieved.nc \
                tas_ONDJF_pointers.validate.197901.retrieved.NCAR_CCSM4_r1i1p1.nc
$ ncview tas_ONDJF_pointers.validate.197901.retrieved.NCAR_CCSM4_r1i1p1.nc
```

Note: The `ncks` command can be slow. For some unknown reasons NCO version 4.5.3 and earlier with netCDF version 4.3.3.1 and earlier does not seem optimized for highly hierarchical files. At the moment, there are no indications that more recent versions have fixed this issue.

BASH script

This recipe is summarized in the following BASH script. The `--password_from_pipe` option is introduced to fully automatize the script:

```
#!/bin/bash

OPENID='your openid'
# Single quotes are necessary here:
PASSWORD='your ESGF password'
#Discover data:
cdb_query CMIP5 ask --ask_month=1,2,10,11,12 \
                    --ask_var=tas:day-atmos-day,orog:fx-atmos-fx \
                    --ask_experiment=amip:1979-2004 \
                    --model=CanAM4 --model=CCSM4 --model=GISS-E2-R --model=MRI-CGCM3 \
                    --num_procs=10 \
                    tas_ONDJF_pointers.nc

#List simulations:
cdb_query CMIP5 list_fields -f institute \
                            -f model \
                            -f ensemble \
                            tas_ONDJF_pointers.nc

#Validate simulations:
#Exclude data_node http://esgf2.dkrz.de because it is on a tape archive (slow)
#If you do not exclude it, it will likely be excluded because of its slow
#response time.
#
```

```
#The first time this function is used, it might fail and ask you to register your␣
↪kind of user.
#This has to be done only once.
echo $PASSWORD | cdb_query CMIP5 validate \
                           --openid=$OPENID \
                           --password_from_pipe \
                           --num_procs=10 \
                           --Xdata_node=http://esgf2.dkrz.de \
                           tas_ONDJF_pointers.nc \
                           tas_ONDJF_pointers.validate.nc

#List simulations:
cdb_query CMIP5 list_fields -f institute \
                           -f model \
                           -f ensemble \
                           tas_ONDJF_pointers.validate.nc

#CHOOSE:
    # *1* Retrieve files:
        #echo $PASSWORD | cdb_query CMIP5 download_files \
        #                   --download_all_files \
        #                   --openid=$OPENID \
        #                   --password_from_pipe \
        #                   --out_download_dir=./in/CMIP5/ \
        #                   tas_ONDJF_pointers.validate.nc \
        #                   tas_ONDJF_pointers.validate.downloaded.nc

    # *2* Retrieve to netCDF:
        #Retrieve the first month:
        echo $PASSWORD | cdb_query CMIP5 download_opendap --year=1979 --month=1 \
                           --openid=$OPENID \
                           --password_from_pipe \
                           tas_ONDJF_pointers.validate.nc \
                           tas_ONDJF_pointers.validate.197901.retrieved.nc

        #Pick one simulation:
        #Note: this can be VERY slow!
        ncks -G :8 -g /NCAR/CCSM4/amip/day/atmos/day/r1i1p1/tas \
           tas_ONDJF_pointers.validate.197901.retrieved.nc \
           tas_ONDJF_pointers.validate.197901.retrieved.NCAR_CCSM4_r1i1p1.nc
        #Remove soft_links informations:
        ncks -L 0 -O -x -g soft_links \
           tas_ONDJF_pointers.validate.197901.retrieved.NCAR_CCSM4_r1i1p1.nc \
           tas_ONDJF_pointers.validate.197901.retrieved.NCAR_CCSM4_r1i1p1.nc

        #Look at it:
        #When done, look at it. A good tool for that is ncview:
        #   ncview tas_ONDJF_pointers.validate.197901.retrieved.NCAR_CCSM4_r1i1p1.nc

        #Convert hierarchical file to files on filesystem (much faster than ncks):
        #Identity reduction simply copies the data to disk
        cdb_query CMIP5 reduce \
                           '' \
                           --out_destination=./out/CMIP5/ \
                           tas_ONDJF_pointers.validate.197901.retrieved.nc \
                           tas_ONDJF_pointers.validate.197901.retrieved.converted.nc

        #The files can be found in ./out/CMIP5/:
```

```
        #find ./out/CMIP5/ -name '*.nc'
```

2. Speeding things up

The `ask` and `validate` steps can be slow. They can be sped up by querying the archive simulation per simulation AND do it asynchronously.

Asynchronous discovery

The *ask* and *validate* commands provides a basic multi-processing implementation:

```
$ cdb_query CMIP5 ask --num_procs=5 \
                        ... \
                        tas_ONDJF_pointers.nc
$ cdb_query CMIP5 validate --num_procs=5 \
                        tas_ONDJF_pointers.nc \
                        tas_ONDJF_pointers.validate.nc
```

This command uses 5 processes and queries the archive simulation per simulation.

Asynchronous downloads

The *download_files* and *download_opendap* commands also provides a basic multi-processing implementation. By default, data from different data nodes is retrieved in parallel. One can allow more than one simultaneous download per data node

```
$ cdb_query CMIP5 download_files --num_dl=5 ...
$ cdb_query CMIP5 download_opendap --num_dl=5 ...
```

These commands now allow 5 simulataneous download per data node.

3. Retrieving precipitation for JJAS over France (CORDEX)

Specifying the discovery

This relies on the idea that all queries are for a experiment list and a variable list. The CORDEX project has however another important component that one might want to query: its domain. The first step is thus to find what domains are available

```
$ cdb_query CORDEX ask --ask_experiment=historical:1979-2004 \
                       --ask_var=pr:day \
                       --ask_month=6,7,8,9 \
                       --list_only_field=domain \
                       pr_JJAS_France_pointers.nc
MNA-44
EAS-44
SAM-44
MNA-22
WAS-44i
ANT-44
```

```
EUR-44
CAM-44
EUR-11
ARC-44
AFR-44
WAS-44
NAM-44
```

Here the `--list_only_field=domain` option lists all available domains. The result is an (unsorted) list of domain identifiers. The European domains (`EUR-11` and `EUR-44`) are what we want. For the sake of this recipe, we are going to limit our discovery to the higher resolution data `EUR-11`

Discovering the data

The script is run using:

```
$ cdb_query CORDEX ask --ask_experiment=historical:1979-2004 \
                       --ask_var=pr:day \
                       --month=6,7,8,9 \
                       --domain=EUR-11 \
                       --driving_model=ICHEC-EC-EARTH \
                       --num_procs=10 \
                       pr_JJAS_France_pointers.nc
This is a list of simulations that COULD satisfy the query:
EUR-11,DMI,ICHEC-EC-EARTH,HIRHAM5,v1,r3i1p1,historical
EUR-11,CLMcom,ICHEC-EC-EARTH,CCLM4-8-17,v1,r12i1p1,historical
EUR-11,KNMI,ICHEC-EC-EARTH,RACMO22E,v1,r1i1p1,historical
EUR-11,SMHI,ICHEC-EC-EARTH,RCA4,v1,r12i1p1,historical
cdb_query will now attempt to confirm that these simulations have all the requested␣
↪variables.
This can take some time. Please abort if there are not enough simulations for your␣
↪needs.
```

Obtaining the tentative list of simulations can take a few minutes but confirming that these simulations have all the requested variables should take a few minutes, depending on your connection to the ESGF IPSL node. It returns a self-descriptive netCDF file with pointers to the data. Try looking at the resulting netCDF file using `ncdump`:

```
$ ncdump -h pr_JJAS_France_pointers.nc
```

As you can see, it generates a hierarchical netCDF4 file. `cdb_query CORDEX list_fields` offer a tool to query this file.

Querying the discovered data

For example, if we want to know how many different simulations were made available, we would run

```
$ cdb_query CORDEX list_fields -f domain -f driving_model -f institute \
                               -f rcm_model -f rcm_version -f ensemble pr_JJAS_France_
↪pointers.nc
EUR-11,ICHEC-EC-EARTH,CLMcom,CCLM4-8-17,v1,r12i1p1
EUR-11,ICHEC-EC-EARTH,DMI,HIRHAM5,v1,r3i1p1
EUR-11,ICHEC-EC-EARTH,KNMI,RACMO22E,v1,r1i1p1
EUR-11,ICHEC-EC-EARTH,SMHI,RCA4,v1,r12i1p1
```

This test was run on June 23, 2016 and these results represent the data presented by the ESGF on that day.

If this list of models in satisfying, we next check the paths

```
$ cdb_query CORDEX list_fields -f path pr_JJAS_France_pointers.nc
http://cordexesg.dmi.dk/thredds/dodsC/cordex_general/cordex/output/EUR-11/DMI/ICHEC-
↪EC-EARTH/historical/r3i1p1/DMI-HIRHAM5/v1/day/pr/v20131119/pr_EUR-11_ICHEC-EC-EARTH_
↪historical_r3i1p1_DMI-HIRHAM5_v1_day_19510101-19551231.
↪nc|SHA256|d172a848bfaa24db89c5f550046c8dfc789e61f5b81c6d9ea21709c70b17eff7|d2d75739-
↪4023-446a-a834-c111daf6d970
...
```

We consider the first path. It is constituted of two parts. The first part begins with `http://esgf-node.ipsl.fr/...` and ends a the vertical line. This is an *OPENDAP* link. The second part, at the right of the vertical line, is the checksum type, the checksum and the tracking id.

Hint: The command `cdb_query CORDEX ask` does not guarantee that the simulations found satisfy ALL the requested criteria.

Validating the simulations

Warning: From now on it is assumed that the user has installed properly resgistered with the ESGF. Using the `--openid` option combined with an ESGF account takes care of this. The first time this function is used, it might fail and ask you to register your kind of user. This has to be done only once.

To narrow down our results to the simulations that satisfy ALL the requested criteria, we can use

```
$ cdb_query CORDEX validate \
                --openid=$OPENID \
                --num_procs=10 \
                pr_JJAS_France_pointers.nc \
                pr_JJAS_France_pointers.validate.nc
```

To output now has a time axis for each variable (except fx). It links every time index to a time index in a UNIQUE file (remote or local). Try looking at the resulting netCDF file using `ncdump`:

```
$ ncdump -h pr_JJAS_France_pointers.validate.nc
```

Again, this file can be queried for simulations:

```
$ cdb_query CORDEX list_fields -f domain -f driving_model -f institute \
                               -f rcm_model -f rcm_version -f ensemble pr_JJAS_France_
↪pointers.validate.nc
EUR-11,ICHEC-EC-EARTH,CLMcom,CCLM4-8-17,v1,r12i1p1
EUR-11,ICHEC-EC-EARTH,DMI,HIRHAM5,v1,r3i1p1
EUR-11,ICHEC-EC-EARTH,KNMI,RACMO22E,v1,r1i1p1
EUR-11,ICHEC-EC-EARTH,SMHI,RCA4,v1,r12i1p1
```

We can see that no simulations were excluded. This means that they had ALL the variables for ALL the months of ALL the years for the historical experiment.

Retrieving the data: *wget*

cdb_query CORDEX includes built-in functionality for retrieving the paths. It is used as follows

```
$ cdb_query CORDEX download_files --out_download_dir=./in/CMIP5/ \
                                  --openid=$OPENID \
                                  --download_all_files \
                                  pr_JJAS_France_pointers.validate.nc \
                                  pr_JJAS_France_pointers.validate.files.nc
```

It downloads the paths listed in `pr_JJAS_France_pointers.validate.nc` and create a new soft links file `pr_JJAS_France_pointers.validate.files.nc` with the downloaded path registered.

Warning: The retrieved files are structured with the CORDEX DRS. It is good practice not to change this directory structure. If the structure is kept then `cdb_query CORDEX ask` will recognized the retrieved files as local if they were retrieved to a directory listed in the `--Search_path`.

The downloaded paths are now discoverable by `cdb_query CORDEX ask`.

Retrieving the data: *OPeNDAP*

We retrieve the first month:

```
$ cdb_query CORDEX download_opendap --year=1979 --month=6 \
                                --openid=$OPENID \
                                pr_JJAS_France_pointers.validate.nc \
                                pr_JJAS_France_pointers.validate.197906.retrieved.nc
```

This step took about 4 minutes from the University of Toronto on June 23, 2016. Next, we extract precipitation for the simulation with the EUR-11 domain:

```
$ ncks -G :9 -g /EUR-11/DMI/ICHEC-EC-EARTH/historical/r3i1p1/HIRHAM5/v1/day/pr \
               pr_JJAS_France_pointers.validate.197906.retrieved.nc \
               pr_JJAS_France_pointers.validate.197906.retrieved.EUR-11.nc
$ ncview pr_JJAS_France_pointers.validate.197906.retrieved.EUR-11.nc
```

Hint: This file contains a `soft_links` subgroup that contains full traceability informations for the accompyning data.

This data is projected onto a rotated pole grid, making it difficult to zoom in onto France by using slices along dimensions. Sever tools can be used to zoom in even with a rotated pole grid. With *CDO*, one would do:

```
$ cdo -f nc -sellonlatbox,-5.0,10.0,40.0,53.0 -selgrid,curvilinear,gaussian,lonlat \
                        pr_JJAS_France_pointers.validate.197906.retrieved.EUR-11.nc \
                        pr_JJAS_France_pointers.validate.197906.retrieved.EUR-11_
↪France.nc
```

Alternatively, bundled with `cdb_query` there is a simple tool that can accomplish this:

```
$ nc4sl subset --lonlatbox -5.0 10.0 40.0 53.0 \
                        pr_JJAS_France_pointers.validate.197906.retrieved.EUR-11.nc \
                        pr_JJAS_France_pointers.validate.197906.retrieved.EUR-11_
↪France.nc
```

We can make sure that our subsetting was ok:

```
$ ncview pr_JJAS_France_pointers.validate.197906.retrieved.EUR-11_France.nc
```

Subsetting the data BEFORE the *OPENDAP* retrieval

We can subset the soft link file before using `download_opendap` and `cdb_query` will only download the requested data:

```
$ nc4sl subset --lonlatbox -5.0 10.0 40.0 53.0 \
                        pr_JJAS_France_pointers.validate.nc \
                        pr_JJAS_France_pointers.validate.France.nc
```

or, using `reduce_soft_links`:

```
$ cdb_query CORDEX reduce_soft_links \
                            --num_procs=10 \
                            'nc4sl subset --lonlatbox -5.0 10.0 40.0 53.0' \
                            pr_JJAS_France_pointers.validate.nc \
                            pr_JJAS_France_pointers.validate.France.nc
```

In the second method, the subsetting can be performed asynchronously (`--num_procs=10`). Finally, we retrieve the subsetted data:

```
$ cdb_query CORDEX download_opendap --year=1979 --month=6 \
                                --openid=$OPENID \
                                pr_JJAS_France_pointers.validate.France.nc \
                                pr_JJAS_France_pointers.validate.France.197906.
↪retrieved.nc
```

This step took about 3m40s from the University of Toronto. It retrieves all models but only over France. We can then check the variables:

```
$ ncks -G :9 -g /EUR-11/DMI/ICHEC-EC-EARTH/historical/r3i1p1/HIRHAM5/v1/day/pr \
               pr_JJAS_France_pointers.validate.France.197906.retrieved.nc \
               pr_JJAS_France_pointers.validate.France.197906.retrieved.EUR-11.nc
$ ncview pr_JJAS_France_pointers.validate.France.197906.retrieved.EUR-11.nc
```

Should show precipitation over France in June 1979.

The amount of time required for the download is not substantially improved for single month but they are for longer retrievals:

```
$ time cdb_query CORDEX download_opendap --month=6  \
                                         --openid=$OPENID\
                                         pr_JJAS_France_pointers.validate.France.nc \
                                         pr_JJAS_France_pointers.validate.France.June.
↪retrieved.nc
real    25m28.268s
user    14m25.368s
sys 3m18.299s
$ time cdb_query CORDEX download_opendap --month=6  \
                                         --openid=$OPENID\
                                         pr_JJAS_France_pointers.validate.nc \
                                         pr_JJAS_France_pointers.validate.June.
↪retrieved.nc
real    43m45.656s
```

```
user    21m59.345s
sys 8m53.251s
```

BASH script

This recipe is summarized in the following BASH script:

```
#!/bin/bash
#Change to set number of processes to use:
NUM_PROCS=10
#Specify your OPENID
OPENID='your openid'
# Single quotes are necessary here:
PASSWORD='your ESGF password'

#Discover data:
cdb_query CORDEX ask --ask_experiment=historical:1979-2004 \
                     --ask_var=pr:day \
                     --domain=EUR-11 \
                     --num_procs=${NUM_PROCS} \
                     pr_JJAS_France_pointers.nc

#List simulations:
cdb_query CORDEX list_fields -f domain -f driving_model -f institute \
                          -f rcm_model -f rcm_version -f ensemble pr_JJAS_France_
↪pointers.nc

#Validate simulations:
#Exclude data_node http://esgf2.dkrz.de because it is on a tape archive (slow)
#If you do not exclude it, it will likely be excluded because of its slow
#
#The first time this function is used, it might fail and ask you to register your␣
↪kind of user.
#This has to be done only once.
echo $PASSWORD | cdb_query CORDEX validate \
            --openid=$OPENID \
            --password_from_pipe \
            --num_procs=${NUM_PROCS} \
            --Xdata_node=http://esgf2.dkrz.de \
            pr_JJAS_France_pointers.nc \
            pr_JJAS_France_pointers.validate.nc
#CHOOSE:
    # *1* Retrieve files:
        #echo $PASSWORD | cdb_query CORDEX download_files \
        #                    --out_download_dir=./in/CMIP5/ \
        #                    --openid=$OPENID \
        #                    --download_all_files \
        #                    --password_from_pipe \
        #                    pr_JJAS_France_pointers.validate.nc \
        #                    pr_JJAS_France_pointers.validate.files.nc

    # *2* Retrieve to netCDF:
        #Retrieve one month:
        echo $PASSWORD | cdb_query CORDEX download_opendap --year=1979 --month=6 \
                          --openid=$OPENID \
                          --password_from_pipe \
```

```
                          pr_JJAS_France_pointers.validate.nc \
                          pr_JJAS_France_pointers.validate.197906.retrieved.nc

       #Convert to filesystem:
       cdb_query CORDEX reduce --out_destination=./out/CORDEX/ '' \
                               pr_JJAS_France_pointers.validate.197906.retrieved.nc \
                               pr_JJAS_France_pointers.validate.197906.retrieved.
↪converted.nc

       #Subset France on soft_links:
       cdb_query CORDEX reduce_soft_links \
                       --num_procs=${NUM_PROCS} \
                       'nc4sl subset --lonlatbox -5.0 10.0 40.0 53.0' \
                       pr_JJAS_France_pointers.validate.nc \
                       pr_JJAS_France_pointers.validate.France.nc

       #We then retrieve the whole time series over France:
       echo $PASSWORD | cdb_query CORDEX download_opendap \
                           --openid=$OPENID \
                           --password_from_pipe \
                           pr_JJAS_France_pointers.validate.France.nc \
                           pr_JJAS_France_pointers.validate.France.retrieved.nc

       #Convert to filesystem:
       cdb_query CORDEX reduce --out_destination=./out_France/CORDEX/ \
                               --num_procs=${NUM_PROCS} \
                                '' \
                                pr_JJAS_France_pointers.validate.France.retrieved.nc␣
↪\
                                pr_JJAS_France_pointers.validate.France.retrieved.
↪converted.nc
```

4. Operator chaining

The real purpose of `cdb_query` is to perform all of the steps asynchronously. The `ask`, `validate`, `reduce_soft_links` and `download_opendap` operations can be chained and applied to each simulation.

CMIP5

In *CMIP5*, simulations are `institute, model, ensemble` triples. Chaining operators will first determine the simulations triples and chain the operators for every triples. Advanced options allow to bypass this default setting. This will be covered in later recipes. This is the internal mechanics but for the user it is fairly transparent (except when there is an error message).

With operator chaining, recipe 1 could be written:

```
$ OPENID="your openid
$ cdb_query CMIP5 ask validate record_validate download_opendap reduce \
                  --ask_month=1,2,10,11,12 \
                  --ask_var=tas:day-atmos-day,orog:fx-atmos-fx \
                  --ask_experiment=amip:1979-2004 \
                  --Xdata_node=http://esgf2.dkrz.de \
                  --openid=$OPENID \
                  --year=1979 --month=1 \
```

```
                  --out_destination=./out/CMIP5/ \
                  --num_procs=10 \
                  '' \
                  tas_ONDJF_pointers.validate.197901.retrieved.converted.nc
```

It does:

1. Finds ONDJF `tas` and fixed variable `orog` for `amip`.
2. Excludes (`--Xdata_node=http://esgf2.dkrz.de`) data node `http://esgf2.dkrz.de` because it is a tape archive and tends to be slow.
3. Retrieve credentials (`--openid=$OPENID`). It will prompt for your password.
4. Record the result (`record_validate`) of `validate` to `tas_ONDJF_pointers.validate.197901.retrieved.converted.nc.validate`.
5. Does this using 10 processes `--num_procs=10`.
6. Download only January 1979 (`--year=1979 --month=1`).
7. Converts (the empty script `''` passed to `reduce`) the data to the CMIP5 DRS to directory `./out/CMIP5/`.

CORDEX

In *CORDEX*, simulations are `domain,driving_model,institute,rcm_model,rcm_version,ensemble` sextuples. Chaining operators will first determine the simulations triples and chain the operators for every sextuple. Advanced options allow to bypass this default setting. This will be covered in later recipes. This is the internal mechanics but for the user it is fairly transparent (except when there is an error message).

With operator chaining, recipe 3 could be written:

```
$ OPENID="your openid"
$ cdb_query CORDEX ask validate record_validate reduce_soft_links download_opendap␣
↪reduce \
                  --ask_experiment=historical:1979-2004 --ask_var=pr:day --ask_
↪month=6,7,8,9 \
                  --openid=$OPENID \
                  --year=1979 --month=6 \
                  --domain=EUR-11 \
                  --out_destination=./out_France/CORDEX/ \
                  --Xdata_node=http://esgf2.dkrz.de \
                  --num_procs=10 \
                  --reduce_soft_links_script='nc4sl subset --lonlatbox -5.0 10.0 40.0␣
↪53.0' \
                  '' \
                  pr_JJAS_France_pointers.validate.France.retrieved.converted.nc
```

It does:

1. Finds JJAS `pr` for `historical`.
2. Excludes (`--Xdata_node=http://esgf2.dkrz.de`) data node `http://esgf2.dkrz.de` because it is a tape archive and tends to be slow.
3. Retrieve certificates (`--openid=$OPENID`). It will prompt for your password.
4. Record the result (`record_validate`) of `validate` to `pr_JJAS_France_pointers.validate.France.retrieved.converted.nc.validate`.
5. Does this using 10 processes `--num_procs=10`.

6. Download only June 1979 (`--year=1979 --month=6`).
7. Converts (the empty script `''` passed to `reduce`) the data to the CMIP5 DRS to directory `./out_France/CORDEX/`.

Note: From now on, recipes will be presented as chained operators.

5. Remap MAM precipitation and temperature to US (CMIP5)

Discovery and analyzing a sample

With operator chaining, in a BASH script:

```
#!\bin\bash
#Create new cdo grid:
cat >> newgrid_atmos.cdo <<EndOfGrid
gridtype  = lonlat
gridsize  = 55296
xname     = lon
xlongname = longitude
xunits    = degrees_east
yname     = lat
ylongname = latitude
yunits    = degrees_north
xsize     = 288
ysize     = 192
xfirst    = 0
xinc      = 1.25
yfirst    = -90
yinc      = 0.94240837696
EndOfGrid

OPENID='your openid'
# Single quotes are necessary here:
PASSWORD='your ESGF password'
#latlon box -124.78 -66.95 24.74 49.34 is continental us
echo $PASSWORD | cdb_query CMIP5 ask validate record_validate reduce_soft_links␣
↪download_opendap reduce \
                  --ask_month=3,4,5 \
                  --ask_var=tas:mon-atmos-Amon,pr:mon-atmos-Amon \
                  --ask_experiment=historical:1950-2005,rcp85:2006-2050 \
                  --related_experiments \
                  --Xdata_node=http://esgf2.dkrz.de \
                  --openid=$OPENID \
                  --password_from_pipe \
                  --out_destination=./out_sample/CMIP5/ \
                  --num_procs=10 \
                  --year=2000 --month=3 \
                  --reduce_soft_links_script='nc4sl subset --lonlatbox -150.0 -50.0␣
↪20.0 55.0' \
                  'cdo -f nc \
                       -sellonlatbox,-124.78,-66.95,24.74,49.34 \
                       -remapbil,newgrid_atmos.cdo \
                       -selgrid,lonlat,curvilinear,gaussian,unstructured ' \
```

```
us_pr_tas_MAM_pointers.validate.200003.retrieved.converted.nc
```

It does:

1. Finds MAM `pr` and `tas` for 1950 to 2050 `historical`, followed by `rcp85`.
2. Drops simulations (`institute`, `model`, `ensemble`) triples that are not found in both `historical` and `rcp85` for ALL requested years.
3. Excludes (`--Xdata_node=http://esgf2.dkrz.de`) data node `http://esgf2.dkrz.de` because it is a tape archive and tends to be slow.
4. Retrieves certificates (`--openid=$OPENID`). Password read from the pipe (`--password_from_pipe`).
5. Records the result (`record_validate`) of `validate` to `us_pr_tas_MAM_pointers.validate.200003.retrieved.converted.nc.validate`.
6. Selects a slightly larger area than continental US for download (`--reduce_soft_links_script='nc4sl subset --lonlatbox -150.0 -50.0 20.0 55.0'`)
7. Downloads only March 2000 (`--year=2000 --month=3`).
8. Uses a bilinear remapping and focuses on the continental US (`'cdo ... '`).
9. Does this using 10 processes `--num_procs=10`.
10. Converts the data to the CMIP5 DRS to directory `./out_sample/CMIP5/`.
11. Writes a full description of downloaded data (pointers to it) in file `us_pr_tas_MAM_pointers.validate.200003.retrieved.converted.nc`.

Scaling up to the whole dataset

If the data looks OK, then one can use the validate file to bypass the `ask` and `validate` steps:

```
#!\bin\bash
OPENID="your openid"
PASSWORD="your ESGF password"
#latlon box -124.78 -66.95 24.74 49.34 is continental us

echo $PASSWORD | cdb_query CMIP5 reduce_soft_links download_opendap reduce \
                  --openid=$OPENID \
                  --password_from_pipe \
                  --out_destination=./out/CMIP5/ \
                  --num_procs=10 \
                  --reduce_soft_links_script='nc4sl subset --lonlatbox -150.0 -50.0␣
↪20.0 55.0' \
                  'cdo -f nc \
                       -sellonlatbox,-124.78,-66.95,24.74,49.34  \
                       -remapbil,newgrid_atmos.cdo \
                       -selgrid,lonlat,curvilinear,gaussian,unstructured ' \
                  us_pr_tas_MAM_pointers.validate.200003.retrieved.converted.nc.
↪validate \
                  us_pr_tas_MAM_pointers.validate.retrieved.converted.nc
```

This will download all the data!

Hint: It is good practice to first download a small subset to ensure that everything outputs as expected. Because we record the validate step, this two-parts process comes at a very small price.

6. Retrieving DJF monthly atmospheric data over a latitude band (CMIP5)

The following BASH script recovers several variables over a latitude band:

```
#!/bin/bash

#This script discovers and retrieves the geopotential height (zg), meridional wind␣
↪(va) and
#atmospheric temperature (ta) at the monthly frequency (mon) from the atmospheric␣
↪realm (atmos)
#and from monthly atmospheric mean CMOR table (Amon) for years 1979 to 2005 of␣
↪experiment
#historical and years 2006 to 2015 for experiment rcp85.
#
#A ramdisk (/dev/shm/) swap directory is used (--swap_dir option)
#Data node http://esgf2.dkrz.de is excluded because it is a tape archive
#(and therefore too slow for the type of multiple concurrent requests that are␣
↪required)
#
#The data is reduced to a latitude band (55.0 to 60.0) using the
#--reduce_soft_links_script='ncrcat -d lat 55.0 60.0' option and the reduce_soft_
↪links command.

#The results are stored in:
#   1) a validate file (${OUT_FILE}.validate),
#   2) a directory tree under ${OUT_DIR} and
#   3) a pointer file ${OUT_FILE} that can be used in a further reduce step.

#Use 5 processors:
NUM_PROCS=5
OPENID='your openid'
# Single quotes are necessary here:
PASSWORD='your ESGF password'

SWAP_DIR="/dev/shm/lat_band/"
OUT_FILE="DJF_lat_band.nc"
OUT_DIR="out_lat_band/"

#Create swap directory:
mkdir ${SWAP_DIR}
echo $PASSWORD | cdb_query CMIP5 ask validate record_validate reduce_soft_links␣
↪download_opendap reduce \
      --openid=$OPENID \
      --password_from_pipe \
      --swap_dir=${SWAP_DIR} \
      --num_procs=$NUM_PROCS \
      --ask_experiment=historical:1979-2005,rcp85:2006-2015 \
      --ask_var=zg:mon-atmos-Amon,va:mon-atmos-Amon,ta:mon-atmos-Amon \
      --ask_month=1,2,12 \
      --related_experiments \
```

```
      --Xdata_node=http://esgf2.dkrz.de \
      --reduce_soft_links_script='ncrcat -d lat,55.0,65.0' \
      '' \
       --out_destination=${OUT_DIR} \
       ${OUT_FILE}
#Remove swap directory:
rm -r ${SWAP_DIR}
```

CHAPTER 4

BASH script to compile and install NCO on CentOS

The following script has been used on CentOS 5 to successfully build *NCO*. It should compile and install *NCO* sucessfully but it should fail at building the *NCO* documentation. This is not a problem. It is a modified version of a script found on http://idolinux.blogspot.ca/2011/02/nco-netcdf-operators-build-log.html:

```
#!/bin/bash
#This script compiles NCO assuming that
#netCDF4 and HDF5 libraries are already compiled,
#tested and installed on the current system.

#######################################
# You MUST modify the following lines:
#######################################
INSTALL_PATH=$HOME/local/nco-4.4.3

#set these to 0 if you have to re-run this
#script and some of these libraries
#compiled succesfully:
INSTALL_ANTLR=1
INSTALL_UDUNITS=1
INSTALL_GSL=1

NETCDF4_DIR="$HOME/anaconda"
HDF5_DIR="$HOME/anaconda"
###############################################
# Please do not modify anything past this line
# before trying the script!
###############################################

mkdir -p $INSTALL_PATH/src

# ANTLR2
# Do not change the version of ANTLR
# NCO works ONLY with versions 2.7.x and
# NOT with newer versions:
    APP=antlr-2.7.7
```

```
    ANTLR_PATH=$INSTALL_PATH/$APP
    if [ "$INSTALL_ANTLR" -eq "1" ]; then
        rm -rf $ANTLR_PATH
        cd $INSTALL_PATH/src

        rm -rf ${APP}
        wget http://www.antlr2.org/download/${APP}.tar.gz
        tar xzf ${APP}.tar.gz ; cd ${APP}

        CC=gcc CXX='' ./configure \
        --prefix=$ANTLR_PATH \
        --disable-csharp \
        --disable-java \
        --disable-python 2>&1 | tee $APP.config
        make 2>&1 | tee $APP.make
        make install 2>&1 | tee $APP.install
    fi


# UDUNITS
# Here, you can try to change the version number if
# a newer version if avaiable.
    APP=udunits-2.2.13
    UDUNITS_PATH=$INSTALL_PATH/$APP
    if [ "$INSTALL_UDUNITS" -eq "1" ]; then
        rm -rf $UDUNITS_PATH
        cd $INSTALL_PATH/src

        rm -rf ${APP}*
        wget ftp://ftp.unidata.ucar.edu/pub/udunits/${APP}.tar.gz
        tar xzf ${APP}.tar.gz ; cd ${APP}

        CC=gcc CXX='' F77=gfortran ./configure \
        --prefix=$UDUNITS_PATH 2>&1 | tee $APP.config
        make 2>&1 | tee $APP.make
        make install 2>&1 | tee $APP.install
    fi

#GSL
# Here, you can try to change the version number if
# a newer version if avaiable.
    APP=gsl-1.16
    GSL_PATH=$INSTALL_PATH/$APP
    if [ "$INSTALL_GSL" -eq "1" ]; then
        rm -rf $GSL_PATH
        cd $INSTALL_PATH/src

        rm -rf ${APP}*
        wget ftp://ftp.gnu.org/gnu/gsl/${APP}.tar.gz
        tar xzf ${APP}.tar.gz ; cd ${APP}

        ./configure \
        --prefix=$GSL_PATH \
        CFLAGS="-fexceptions" | tee $APP.config
        make 2>&1 | tee $APP.make
        make install 2>&1 | tee $APP.install
    fi
```

```
# NCO
# Here, you can try to change the version number if
# a newer version if available.
    APP=nco-4.4.3
    NCO_PATH=$INSTALL_PATH/$APP
    rm -rf $NCO_PATH
    cd $INSTALL_PATH/src

    rm -rf ${APP}*
    wget http://nco.sourceforge.net/src/${APP}.tar.gz
    tar xzf ${APP}.tar.gz ; cd ${APP}

    export LD_LIBRARY_PATH=$HDF5_DIR/lib:$LD_LIBRARY_PATH
    export PATH=$HDF5_DIR/bin:$PATH
    export LD_LIBRARY_PATH=$NETCDF4_DIR/lib:$LD_LIBRARY_PATH
    export PATH=$NETCDF4_DIR/bin:$PATH
    export LD_LIBRARY_PATH=$ANTLR_PATH/lib:$LD_LIBRARY_PATH
    export PATH=$ANTLR_PATH/bin:$PATH
    export LD_LIBRARY_PATH=$UDUNITS_PATH/lib:$LD_LIBRARY_PATH
    export PATH=$UDUNITS_PATH/bin:$PATH
    export LD_LIBRARY_PATH=$GSL_PATH/lib:$LD_LIBRARY_PATH
    export PATH=$GSL_PATH/bin:$PATH

    CC=gcc CXX='' \
    NETCDF_INC=$NETCDF4_DIR/include \
    NETCDF_LIB=$NETCDF4_DIR/lib \
    NETCDF4_ROOT=$NETCDF4_DIR \
    HDF5_LIB_DIR=$HDF5_DIR/lib \
    UDUNITS2_PATH=$UDUNITS_PATH \
    LDFLAGS="-L$ANTLR_PATH/lib -lantlr \
    -lhdf5_hl -lhdf5 -L$NETCDF4_DIR/lib -lnetcdf" \
    CFLAGS="-I$HDF5_DIR/include \
    -L$HDF5_DIR/lib \
    -I$ANTLR_PATH/include \
    -L$ANTLR_PATH/lib" \
    CPPFLAGS="-I$HDF5_DIR/include \
    -L$HDF5_DIR/lib \
    -I$ANTLR_PATH/include \
    -L$ANTLR_PATH/lib" \
    ./configure \
    --prefix=$NCO_PATH \
    --enable-optimize-custom \
    --enable-netcdf-4 2>&1 | tee $APP.config
    echo "#define ENABLE_NETCDF4 1" >> config.h
    make 2>&1 | tee $APP.make
make install 2>&1 | tee $APP.install
```

Once this script is completed, add `$INSTALL_PATH/nco-4.4.0/bin` to your path.