High Performance Data Analytics in eScience

Lab Tutorial

D. Elia\textsuperscript{1,2}

\textsuperscript{1} Fondazione Centro Euro-Mediterraneo sui Cambiamenti Climatici (CMCC), Lecce, Italy
\textsuperscript{2} University of Salento, Lecce, Italy

On behalf of the ECAS Team
Session outline

✔ Brief introduction to Jupyter Notebook

✔ PyOphidia modules and interface

✔ VMI environment for the Virtual Lab

✔ Overview of ECASLab @ CMCC

✔ PyOphidia notebook demo
“The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.”

1Jupyter Website: https://jupyter.org/
PyOphidia is a GPLv3-licensed Python module to interact with the Ophidia framework and it implements two main classes:

- **Client class**: supports the submissions of Ophidia commands and workflows, as well as the management of session from Python code (similar to the Ophidia Terminal)
  - It allows to run all the Ophidia operators, including massive tasks and workflows

- **Cube class**: provides the datacube type abstraction and the methods to manipulate, process and get information on cubes objects and it builds on the client class
  - Defines a object-oriented approach allowing to handle a datacube more naturally

While the cube module provides a user-friendly interface, the client module allows a finer specification of the operators.
The PyOphidia library: cube class

PyOphidia Cube class introduces the concept of **cube objects** and supports all the Ophidia operators as **methods**.

To this end, the class defines two types of methods according to the type of operator:

- **Class methods**: concerning the operators which do not refer to a particular cube object (e.g. the oph_list, the operators to manage the file system, etc.)

```python
cube.Cube.list(level=2)
```

- **Instance methods**: concern the operators applied directly on a cube object to access and manipulate it (by creating a new cube object)

```python
mycube.info()
mycube2 = mycube.reduce(operation='max', ncores=5)
```
The PyOphidia library: cube class

Example of **cube class** usage:

- Load the module and setup a connection to the server instance (similar to client class)

```python
from PyOphidia import cube

cube.Cube.setclient(username="oph-user", password="oph-passwd", server="127.0.0.1", port="11732")
```

- The arguments can be automatically inferred by the environment, if setup in the `.bashrc`

```python
cube.Cube.setclient(read_env=True)
```

- Once the connection has been setup all the operators can be executed remotely through the related method

```python
cube.Cube.list(level=2)
```
Example of **cube class** usage:

- A cube object can be created in multiple ways. In case of pre-existing cube (pid):
  ```python
  mycube = cube.Cube(pid='http://127.0.0.1/ophidia/1/1')
  ```

- A cube can be also created from a NetCDF file using the constructor function:
  ```python
  mycube = cube.Cube(exp_dim='lat|lon', imp_dim='time', ncores=2,
                     measure='tos', src_path='/path/tos.nc')
  ```

- Or directly using the import method (exactly the same as the previous one):
  ```python
  mycube = cube.Cube.importnc(exp_dim='lat|lon', imp_dim='time', ncores=2,
                               measure='tos', src_path='/path/tos.nc')
  ```

- After the processing, the cube can be deleted with the proper method:
  ```python
  mycube.delete()
  ```
Example of **cube class** usage:

- Once a cube is available in the python code, various operators can be executed to produce new datacubes:

  ```python
  mycube2 = mycube.reduce(operation='max', ncores=5)
  mycube3 = mycube2.subset2(subset_dims="lat|lon|time", ncores=5,
                             subset_filter="-80:30|30:120|151:240")
  mycube4 = mycube3.aggregate(operation='max', ncores=5)
  ```

- Methods can also be concatenated into a single command:

  ```python
  mycube5 = mycube.reduce(operation='max', ncores=5).subset2(
                        subset_dims="lat|lon|time", ncores=5,
                        subset_filter="-80:30|30:120|151:240").aggregate(
                        operation='max', ncores=5)
  ```
The PyOphidia library: client class

The client class allows to run the same commands of the cube class with a lower-level interface and supports the execution of massive operators (param. sweep)

- Commands follow the same structure as for the Oph_term (oph_operator param1=val1;)

```python
from PyOphidia import client
ophclient = client.Client(read_env=True)

ophclient.submit("oph_list level=1", display=True)
```

- Multiple files can be loaded in parallel by specifying a filter on the inputs

```python
ophclient.submit("oph_importnc exp_dim=lat|lon;imp_dim=time;ncores=2;measure=tos;src_path=[path=/path/*.*.nc]")
```

- The same operator can be run in parallel on multiple input cubes

```python
ophclient.submit("oph_reduce2 operation=avg;dim=time;cube=[*]")
```

Ophidia massive operators documentation: http://ophidia.cmcc.it/documentation/users/massive/index.html
Virtual Lab environment

The pre-installed VMI with the full Ophidia stack and other dependencies for the Virtual Lab is available at: https://download.ophidia.cmcc.it/vmi_desktop/training/OphidiaVMI.ova

Login and password are both ophidia. For additional information refer to the summer school virtual lab instructions.

Check the README on the Desktop to startup the environment services

The notebooks for the Virtual Lab are located on the Desktop under the ecas-training/notebooks folder

ECAS_Basics is the notebook shown in the lab tutorial video
ECASLab is a scientific data analytics environment built on top of ECAS (the ENES Climate Analytics Service), one of the thematic services included in the EOSC-hub service portfolio.

It provides a scientific environment exploiting a server-side approach and integrating both data and analysis tools to support data scientists in their daily research activities.

ECASLab starts from a previous effort (OphidiaLab, developed at CMCC Foundation) with the main aim of providing a virtualized research environment for researchers. It represents the entry point for users that want to test, train, exploit the ECAS Thematic Service.

It consists of several components like an ECAS cluster, a JupyterHub instance jointly with a large set of pre-installed Python libraries for running data manipulation, analysis, and visualization, a data publication service and a tool for the infrastructure monitoring (mainly intended for the administrators).

In order to get started with ECASLab please have a look at the Quick Start section and register here to get an account.

https://ecaslab.cmcc.it/
ECASLab Registration form @CMCC

ECASLab Registration Form

Sign Up!

First Name *

Last Name *

E-mail *

Affiliation *

Country *

IS-ENES3

Training

Specify motivation for requesting access. You can simply type “Training”

Select IS-ENES3 as project

https://ecaslab.cmcc.it/web/registration.php
ECASLab JupyterHub service

Demo notebooks

ECAS_Basics Notebook
Demo: ECAS/Ophidia simple commands examples

First of all import PyOphidia modules and connect to server (connection details are inferred from the ECAS environment)

```
In [ ]: from PyOphidia import cube, client
cube.Cube.setclient(read_env=True)
```

Create a datacube from the NetCDF file:

- The file is `data/ecas_training/tasmax_day_CMCC-CESM_rcp85_r1i1p1_20960101-21001231.nc`
- The variable to be imported is `tasmax`
- Data should be arranged in order to operate on time series (time dimension)

**Note: We are not directly reading the file from the Notebook**

```
In [ ]: mycube = cube.Cube.importnc(
           src_path='data/ecas_training/tasmax_day_CMCC-CESM_rcp85_r1i1p1_20960101-21001231.nc',
           measure='tos',
           imp_dim='time',
           ioserver='ophidiaio_memory',
           ncores=2,
           description="Imported cube"
)
```

Check the datacubes available in the virtual file system

```
In [ ]: cube.Cube.list(level=2)
```
Links and references

**Virtual Lab**
- Ophidia Virtual Machine Image: [https://download.ophidia.cmcc.it/vmi_desktop/training/OphidiaVMI.ova](https://download.ophidia.cmcc.it/vmi_desktop/training/OphidiaVMI.ova)
- Updated training material: [https://github.com/ECAS-Lab/ecas-training/tree/ESiWACE2_SummerSchool_2020](https://github.com/ECAS-Lab/ecas-training/tree/ESiWACE2_SummerSchool_2020)

**Ophidia**
- Ophidia Website: [http://ophidia.cmcc.it](http://ophidia.cmcc.it)
- Ophidia Doc: [http://ophidia.cmcc.it/documentation](http://ophidia.cmcc.it/documentation)

**ECASLab**
- CMCC ECASLab instance: [https://ecaslab.cmcc.it/](https://ecaslab.cmcc.it/)
- ECASLab registration form @ CMCC: [https://ecaslab.cmcc.it/web/registration.php](https://ecaslab.cmcc.it/web/registration.php)

**PyOphidia**
- PyOphidia Doc: [http://ophidia.cmcc.it/documentation/users/pyophidia/](http://ophidia.cmcc.it/documentation/users/pyophidia/)
- PyOphidia repository: [https://github.com/OphidiaBigData/PyOphidia](https://github.com/OphidiaBigData/PyOphidia)

**Other software/Python modules used in the examples**
- Cartopy Doc: [https://scitools.org.uk/cartopy/docs/latest/](https://scitools.org.uk/cartopy/docs/latest/)
- Matplotlib User’s Guides: [https://matplotlib.org/users/index.html](https://matplotlib.org/users/index.html)
Thank you!

These activities are supported in part by ESiWACE2, EOSC-Hub and IS-ENES3 projects:

ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 823988.

EOSC-hub receives funding from the EU’s Horizon 2020 research and innovation programme under grant agreement No. 777536.

IS-ENES3 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 824084.

Contact us at: ecas-support@cmcc.it