Integrating Machine Learning Algorithms and HPDA Frameworks to Run Predictive Analytics on Large-Scale Climate and Weather Datasets

Alessandro D’Anca
Sandro Fiore, Giovanni Aloisio

PASC18 Conference
Basel, 2-4 July 2018
Outline

• Analytics requirements and needs in the climate context

• Ophidia
  – Architecture
    • Primitives
    • Data and metadata operators
    • Workflow support
    • End-user view: PyOphidia, Ophidia Terminal, OphidiaLab

• Time series prediction and LSTM Network
  – Integration in Ophidia as “primitives”

• Case studies
  – WFR over the Brazilian region of Curitiba
  – SANIFS over South of Italy
Data analytics requirements and use cases

Requirements and needs focus on:

- Time series analysis
- Data subsetting
- Model intercomparison
- Multimodel means
- Massive data reduction
- Data transformation (through array-based primitives)
- Param. Sweep experiments (same task applied on a set of data)
- Maps generation
- Ensemble analysis
- Data analytics workflow support

But also…

- Performance
- re-usability
- extensibility
The Ophidia project

**Ophidia** ([http://ophidia.cmcc.it](http://ophidia.cmcc.it)) is a CMCC Foundation research project addressing big data challenges for eScience.

It provides support for declarative, parallel, server-side data analysis exploiting parallel computing techniques and database approaches.

Exploits a multidimensional data model providing the data cube abstraction for access and analysis of scientific n-dimensional data.
Ophidia in a nutshell

✔ Big data stack for scientific data analysis

✔ **Features**: time series analysis (array-based analysis), data subsetting (by value/index), data aggregation, model intercomparison, OLAP, etc.

✔ Use of parallel operators and parallel I/O

✔ **Support for complex workflows / operational chains**

✔ Extensible: **simple API** to support framework extensions like new operators and array-based primitives
  ✔ currently 50+ operators and 100+ primitives provided

✔ **Multiple interfaces** available (WS-I, GSI/VOMS, OGC-WPS).

✔ Programmatic access via C and **Python APIs**

✔ Support for both **batch & interactive** data analysis

✔ Command line interpreter for submitting operators.
Ophidia architecture

Multi-interface server front-end & job/workflow management

Analytics framework for the execution of parallel MPI-based (data cube) operators

Multiple I/O servers (MySQL or native in-memory) run array-based primitives on data

Distributed hardware resources to manage data storage

OphidiaDB maps data fragmentation and tracks metadata
Array based primitives (about 100)

- Ophidia provides a **wide set of array-based primitives** to perform data summarization, sub-setting, predicates evaluation, statistical analysis, compression, etc.

- Primitives come as plugins and are applied on a single datacube chunk (fragment)

- **Primitives can be nested** to get more complex functionalities

- **Compression is a primitive too!**

- New primitives can be easily integrated as additional plugins
<table>
<thead>
<tr>
<th>OPERATOR NAME</th>
<th>OPERATOR DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPH_APPLY(datacube_in, datacube_out, array_based_primitive)</td>
<td>Creates the datacube_out by applying the array-based primitive to the datacube_in</td>
</tr>
<tr>
<td>OPH_DUPLICATE(datacube_in, datacube_out)</td>
<td>Creates a copy of the datacube_in in the datacube_out</td>
</tr>
<tr>
<td>OPH_SUBSET(datacube_in, subset_string, datacube_out)</td>
<td>Creates the datacube_out by doing a sub-setting of the datacube_in by applying the subset_string</td>
</tr>
<tr>
<td>OPH_MERGE(datacube_in, merge_param, datacube_out)</td>
<td>Creates the datacube_out by merging groups of merge_param fragments from datacube_in</td>
</tr>
<tr>
<td>OPH_SPLIT(datacube_in, split_param, datacube_out)</td>
<td>Creates the datacube_out by splitting into groups of split_param fragments each fragment of the datacube_in</td>
</tr>
<tr>
<td>OPH_INTERCOMPARISON(datacube_in1, datacube_in2, datacube_out)</td>
<td>Creates the datacube_out which is the element-wise difference between datacube_in1 and datacube_in2</td>
</tr>
<tr>
<td>OPH_DELETE(datacube_in)</td>
<td>Removes the datacube_in</td>
</tr>
</tbody>
</table>

**Data processing**

**Import/Export**

**Data Access**

**Metadata management**

<table>
<thead>
<tr>
<th>OPERATOR NAME</th>
<th>OPERATOR DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPH_EXPORT_NC(datacube_in, file_out)</td>
<td>Exports the datacube_in data into the file_out NetCDF file.</td>
</tr>
<tr>
<td>OPH_IMPORT_NC(file_in, datacube_out)</td>
<td>Imports the data stored into the file_in NetCDF file into the new datacube_in datacube</td>
</tr>
<tr>
<td>OPH_INSPECT_FRAG(datacube_in, fragment_in)</td>
<td>Inspects the data stored in the fragment_in from the datacube_in</td>
</tr>
<tr>
<td>OPH_PUBLISH(datacube_in)</td>
<td>Publishes the datacube_in fragments into HTML pages</td>
</tr>
</tbody>
</table>

**Operators “Metadata”**

<table>
<thead>
<tr>
<th>OPERATOR NAME</th>
<th>OPERATOR DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPH_CUBE_ELEMENTS(datacube_in)</td>
<td>Provides the total number of the elements in the datacube_in</td>
</tr>
<tr>
<td>OPH_CUBE_SIZE(datacube_in)</td>
<td>Provides the disk space occupied by the datacube_in</td>
</tr>
<tr>
<td>OPH_LIST(void)</td>
<td>Provides the list of available datacubes.</td>
</tr>
<tr>
<td>OPH_CUBEIO(datacube_in)</td>
<td>Provides the provenance information related to the datacube_in</td>
</tr>
<tr>
<td>OPH_FIND(search_param)</td>
<td>Provides the list of datacubes matching the search_param criteria</td>
</tr>
</tbody>
</table>
The analytics framework: “data” operators

INPUT DATA CUBE

OUTPUT DATA CUBE

REDUCE ALL MAX

AGGREGATE ALL MAX

OUTPUT (INPUT) DATA CUBE

INPUT (OUTPUT) DATA CUBE

OUTPUT (INPUT) DATA CUBE

SPLIT by 10 FRAG

(MERGE by 10 FRAG)
The analytics framework: "metadata" operators
The workflow runtime engine is the core component of the Ophidia Server:
• it formats the commands for the analytics framework;
• submits the tasks to the resource manager;
• checks for task status updates in the runtime environment and
• provides the proper response messages.
Workflow submission

[Response]
Workflow Status

OPH STATUS COMPLETED

Workflow Progress

<table>
<thead>
<tr>
<th>NUMBER OF COMPLETED TASKS</th>
<th>TOTAL NUMBER OF TASKS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

Workflow Task List

<table>
<thead>
<tr>
<th>OPH JOB ID</th>
<th>SESSION CODE</th>
<th>WORKFLO ID</th>
<th>WORKFLOW ID</th>
<th>TASK NAME</th>
<th>TYPE</th>
<th>EXIT STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><a href="http://193.204.199.174/ophidia/sessions/37669923831130223511449455166146380/experiment?247#3144">http://193.204.199.174/ophidia/sessions/37669923831130223511449455166146380/experiment?247#3144</a></td>
<td>376699238311302232511449455166146380</td>
<td>247</td>
<td>3145</td>
<td>3144</td>
<td>Loop on tasmin and tasmax cubes</td>
</tr>
<tr>
<td></td>
<td><a href="http://193.204.199.174/ophidia/sessions/37669923831130223511449455166146380/experiment?247#3146">http://193.204.199.174/ophidia/sessions/37669923831130223511449455166146380/experiment?247#3146</a></td>
<td>376699238311302232511449455166146380</td>
<td>247</td>
<td>3146</td>
<td>3144</td>
<td>Compute operation over time (1)</td>
</tr>
<tr>
<td></td>
<td><a href="http://193.204.199.174/ophidia/sessions/37669923831130223511449455166146380/experiment?247#3147">http://193.204.199.174/ophidia/sessions/37669923831130223511449455166146380/experiment?247#3147</a></td>
<td>376699238311302232511449455166146380</td>
<td>247</td>
<td>3147</td>
<td>3144</td>
<td>Compute operation over time (2)</td>
</tr>
<tr>
<td></td>
<td><a href="http://193.204.199.174/ophidia/sessions/37669923831130223511449455166146380/experiment?247#3148">http://193.204.199.174/ophidia/sessions/37669923831130223511449455166146380/experiment?247#3148</a></td>
<td>376699238311302232511449455166146380</td>
<td>247</td>
<td>3148</td>
<td>3144</td>
<td>Conversion from Kelvin to Celsius degrees (1)</td>
</tr>
<tr>
<td></td>
<td><a href="http://193.204.199.174/ophidia/sessions/37669923831130223511449455166146380/experiment?247#3149">http://193.204.199.174/ophidia/sessions/37669923831130223511449455166146380/experiment?247#3149</a></td>
<td>376699238311302232511449455166146380</td>
<td>247</td>
<td>3149</td>
<td>3144</td>
<td>Conversion from Kelvin to Celsius degrees (2)</td>
</tr>
</tbody>
</table>
**End-user view**

**Oph_Term**: a commands interpreter with no GUI (like bash), serving as a client for the Ophidia framework

**Ophidia framework**: Server-side processing

*Through the oph_term we “send” commands to the framework*
PyOphidia provides a Python interface to submit commands to the Ophidia Server and to retrieve/deserialize the results.

Two classes implemented:

- **Client class**: connect to the server, navigate into the ophidia file system, submit workflows, manage sessions, etc.

- **Cube class**: manipulate cubes (reduce, subset, operations between cubes, intercomparison, etc.), get information on cubes (schema, dimensions, metadata, etc.)
OphidiaLab: Jupyter notebooks

Import PyOphidia and connect to server instance

```python
from PyOphidia import cube, client
cube.Cube.setclient(read_env=True)
```

Import data and extract a single time series

```python
mycube2 = mycube.subset2(subset_dims='lat|lon', subset_filter='0:1|0:1', ncores=5)
data = mycube2.export_array()
```

Plot time series

```python
import matplotlib.pyplot as plt
y = data['measure'][0]['values'][0][::]
x = data['dimension'][2]['values'][::]
plt.figure(figsize=(11, 3), dpi=100)
plt.plot(x, y)

plt.ylabel(data['measure'][0]['name'] + " (degK)"
plt.xlabel("Days since 2001/01/01")
plt.title('Sea Surface Temperature (point 0.5, 1)')
plt.show()
```

Convert from Kelvin to Celsius degrees

```python
mycube3 = mycube2.apply(query="oph_sum_scalar('OPH_FLOAT','OPH_FLOAT',measure,-273.15", description="celsius")
data = mycube3.export_array()
```

Plot time series

```python
y = data['measure'][0]['values'][0][::]
x = data['dimension'][2]['values'][::]
plt.figure(figsize=(11, 3), dpi=100)
plt.plot(x, y)

plt.ylabel(data['measure'][0]['name'] + " (degC)"
plt.xlabel("Days since 2001/01/01")
plt.title('Sea Surface Temperature (point 0.5, 1)')
plt.show()
```

https://ophidialab.cmcc.it
Climate indicators processing (I)

✔ In the CLIPC project, processing chains for data analysis are being implemented with Ophidia to compute **climate indicators**

✔ **Parallel approach**
  ✔ Inter-parallelism: Multiple branches are executed in parallel
  ✔ Intra-parallelism: data analysis operators have been parallelized too (e.g. MPI)

✔ **First set of indicators included**: TNn, TNx, TXn, TXx

✔ **Input files**: 12GBs (TasMin & TasMax)
Snow on/off – Length of snow season

- **Dataset time range**: 1979-2012
- **6341 nc files**
- **50 GB of input data**
- **599 tasks performed**
- **99 NetCDF output files (6MB each)**
- **21 tasks in the exp. description**
ESIWACE project

ESDM - Earth System Data Middleware

Provides an interface between the commonly used HDF library and storage which addresses both the performance of POSIX and the usability of object stores

ESDM - enabled "HPDA application"

- We are compiling Ophidia with the ESDM interface (I/O operators) on CMCC HPC infrastructure (Athena) to run a multi-model use case on CMIP data

The ESIWACE project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 675191 http://www.esiwace.eu
The **ENES Climate Analytics Service (ECAS)**, proposed by CMCC & DKRZ in EOSC-hub supports climate data analysis.

- It is one of the **EOSC-Hub Thematic Services** and has been ranked as the **1st out of 64** Thematic Service proposals.

- ECAS builds on top of the **Ophidia big data analytics framework** with components from INDIGO-DataCloud, EUDAT and EGI in order to provide a **multi-model environment** for CMIP-based analytics experiments in ESGF.
We modeled the time series as a supervised learning problem, that is, as a sequence of inputs and outputs.

At each stage, the network receives as input the $n$ values in the past from a time $t$. The output represents $h$ values in the future.

The goal of the network is to learn the mapping from the input to the output.
**Long Short-Term Memory Network (LSTM)**

- RNNs are a class of Neural Networks tailored for sequential data.
- Standard RNNs are not able to capture dependencies over long ranges because of the vanishing or exploding gradient.
- **LSTMs** are designed to overcome the problem. They are capable of remembering information over long periods of time.

Ophidia has been integrated with a LSTM algorithm

A new couple of “primitives” have been developed for
- Training the Network
- Performing the validation phase
Ophidia Primitives For LSTM: Training

• The algorithm has been divided in two phases: one for training and one for test/prediction.

• The primitive for the training task is the following:

```python
oph_lstm(input_OPH_TYPE, output_OPH_TYPE, measure, 
dim_in, dim_out, n_h_layers, n_h_neurons, [dropout], 
[learning_rate], [unrolled_len], [minibatch_size], 
[max_epoch])
```

• It can be run in a SQL statement or in the OPH_APPLY operator.

• After the training phase, the resulting neural network with updated parameters is saved as a binary array in a datacube. It can then be reused in the test phase.
The primitive for the test/prediction phase is the following:

```python
oph_lstm_predict(input_OPH_TYPE, output_OPH_TYPE,
measure_a, measure_b, test)
```

- In test mode, our goal is to compare the network predictions with the actual values. We can evaluate if the model generalizes well or there is overfitting.
- In prediction mode, we pass a new input and ask the LSTM to generate a forecast for the specified time steps.
- Currently, such primitive can only be run in SQL statements, since it does not exist an operator that supports it yet.
LSTM In Ophidia: Use Case (I)

Weather Research and Forecasting (WRF) Model running over the Brazilian region of Curitiba

- 14 days of data for training (672 samples)
- 2 days of predictions (96 output values)
The SANIFS model for sea level forecast

The Southern Adriatic Northen Ionian coastal Forecasting System (SANIFS) is a physic-driven model, providing short-term forecasts about sea levels.

- LSTM tested with the sea level observations acquired from tide gauge devices.
- The observations refer to the period between
  - 31 December 2012
  - 31 December 2016
- 80% of data used for training
- 20% of data used for testing
Conclusions and future work

- **Ophidia** is a big data analytics framework for eScience
- Multiple use cases for data analysis have been implemented
- It provides access via **CLI** (end-users) and **API** (devel users)
- Strong **workflow support** and **in-memory** analytics

- **First integration of Machine Learning** capabilities into Ophidia
  - Test on bigger datasets and with longer forecasts
  - Study the spatial correlation between different points
- Evaluate in depth ML opportunities and build **concrete use cases** for our scientists
- **Co-design approach** (computational scientists, climate scientists, vendors)
Ophidia – Useful Resources

- Website: http://ophidia.cmcc.it
- Doc: http://ophidia.cmcc.it/documentation
- The Ophidia code is available on GitHub under GPLv3 license at https://github.com/OphidiaBigData
- RPMs are also available for CentOS7 and Ubuntu14 at the following repo: http://download.ophidia.cmcc.it/
- Youtube Channel https://www.youtube.com/user/OphidiaBigData/
- A Virtual Machine Image (OVA format) is also available at https://download.ophidia.cmcc.it/vmi_desktop/ to get started in a few minutes with Ophidia
Thanks

http://ophidia.cmcc.it

@OphidiaBigData

www.youtube.com/user/OphidiaBigData