

JACOBS UNIVERSITY

SEMESTER PROJECT

EARTH SCIENCE DATA ANALYSIS, ACCESS AND VISUALISATION

# Big Data and Cloud Computing

*Rahul Bhat*

supervised by

Prof. Dr. Peter Baumann

May 31, 2016

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Collaborators</b>	<b>2</b>
<b>3</b>	<b>Scientific goal</b>	<b>2</b>
<b>4</b>	<b>Dataset</b>	<b>3</b>
<b>5</b>	<b>Data Format</b>	<b>3</b>
<b>6</b>	<b>Data Access</b>	<b>3</b>
<b>7</b>	<b>Conclusion</b>	<b>7</b>

## ***Abstract***

Geospatial data or geographic information is the data with the spatial component, it has an information of a geographic location and boundaries on Earth, such as oceans, and more. This geographic data is in the form of coordinates and topology and it is the data that can be mapped. Spatial data can be accessed, manipulated or analyzed through Geographic Information Systems (GIS). The goal of this project is to get the data from "KNMI Data Centre" and then after careful visualization and after analyzing that data, ingest the valuable information from that data into Rasdaman.

## **1 Introduction**

In recent times, "Big Data" has been the talk in the analytic sector. The questions which one should ask is what is "Big Data"? The word "Big Data" came because of the huge amount of data is generating every second. Big Data is a massive volume of both structured or unstructured data, which moves so quickly and which exceeds the processing capacity of database systems. So it is very difficult to process this large amount of data using traditional techniques. Geospatial data is big in size and sometimes in TB (terabyte) and to store that huge amount of data we are using an array engine called Rasdaman which is a raster data manager which allows storing and querying a huge amount of multi-dimensional data.

## **2 Collaborators**

- **KNMI Data Centre**

Royal Netherlands Meteorological Institute(KNMI) provides access to KNMI data on weather, climate, and seismology. KNMI data is available on various topics such as Historical data, data on meteorological stations, modeling, earthquake data and satellite products etc. For each dataset a description (or metadata) is available. Most datasets in this portal are free to access, There are some which require a registration and for some datasets you must first request access rights [?].

- **EUDAT**

They are collaborative Pan-European infrastructure for research data services, training & consultancy. For a large-scale deployment, EUDAT provides hosting sites to migrate the data.

## **3 Scientific goal**

The goal was to work on the subsurface voxel models and a mentor had been assigned for this project "Alessandro Spinuso from KNMI". He was the guide throughout the project. He gave the information from where I will get that and I had to work under his guidelines.

- The primary aim was to establish sample services in support of and in line with the EUDAT activity.
- Work on the subsurface voxel models.
- Access the data from KNMI and then after observing the data, Ingest that data into Rasdaman.
- Run queries on that data. Data Analysis.

- Get useful information about that data.
- Ingest the valuable information from that data to Rasdaman.

## 4 Dataset

Data Set - MSG-CPP cloud, precipitation, and radiation products. The Cloud Physical Properties (CPP) algorithm is being developed at KNMI to derive cloud, precipitation, and radiation products from satellite instruments. The use of these products is granted to every interested user, free of charge.

Category : Climatology Meteorology Atmosphere

## 5 Data Format

The format of the data is NetCDF and the size of the data is 5.924.159.838 bytes (5,92 GB on disk). The network common data form (NetCDF) is a data abstraction for storing and retrieving multidimensional data. NetCDF is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. NetCDF is useful for the objects that contain different kinds of data in a heterogeneous network environment. NetCDF was developed and is maintained at Unidata. Unidata provides data and software tools for use in geoscience education and research. NetCDF is available from many different repositories. A development version will typically have a name such as "netcdf-devel" or "libnetcdf-dev".

## 6 Data Access

### Gdalinfo

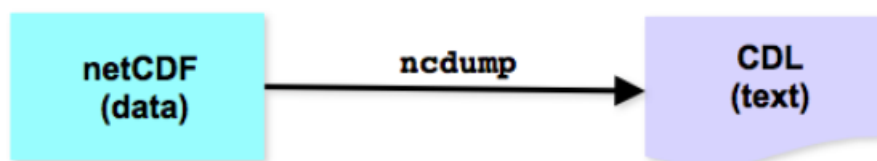
This utility allows to get info from the GDAL library, for instance, specific Driver capabilities and input Datasets and files properties.

Command : `gdalinfo "filename"`

### Ncdump

The ncdump command-line utility converts netCDF data to Ncdump human-readable text form.

command : `ncdump "filename"`

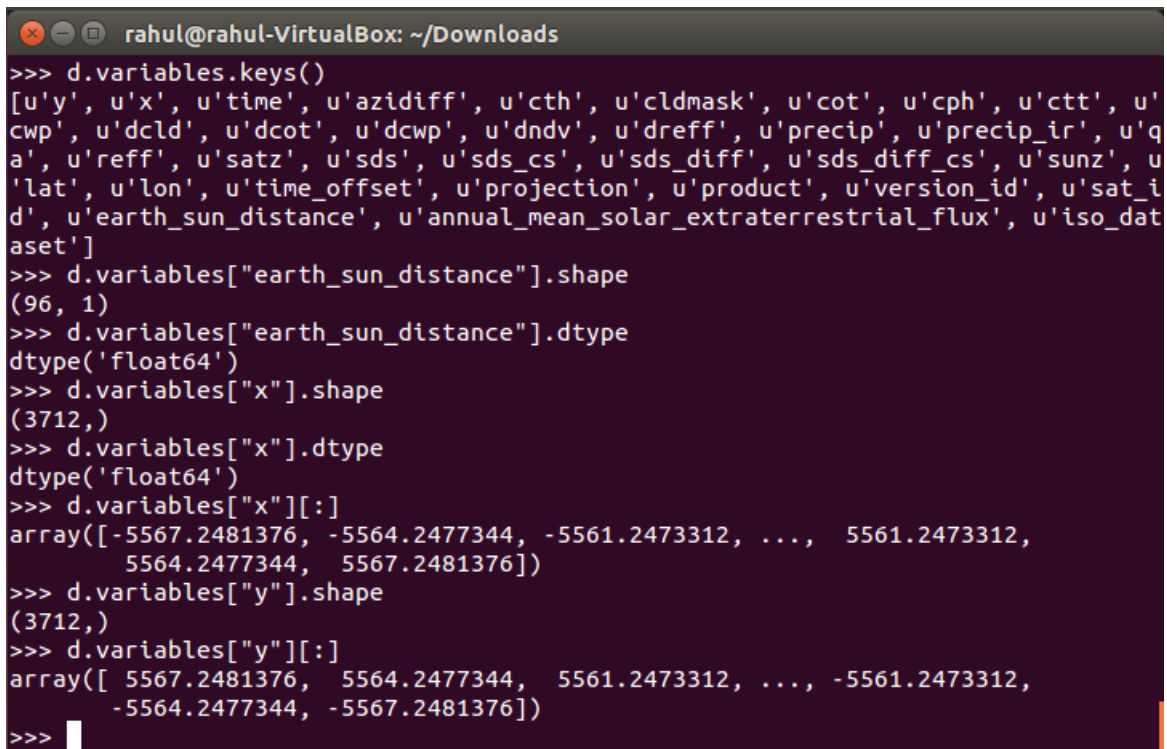


After accessing the data it would be difficult to ingest the data into Rasdaman because of the format of the file that is netCDF. Rasdaman reported an error when tried to ingest the data. Rasdaman was killing the process in between. I tried the same thing with tiff file and it successfully got ingested.

So the option was to convert the netCDF into tiff format. This became possible with the use of Python. There is a library in python 'netCDF4' which reads the netCDF data.

**command :**

```
import numpy as np
from netCDF4 import Dataset
d = Dataset("filename")
```



```
rahul@rahul-VirtualBox: ~/Downloads
>>> d.variables.keys()
[u'y', u'x', u'time', u'azidiff', u'cth', u'cldmask', u'cot', u'cph', u'ctt', u'
cwp', u'dcld', u'dcot', u'dcwp', u'dndv', u'dreff', u'precip', u'precip_ir', u'q
a', u'reff', u'satz', u'sds', u'sds_cs', u'sds_diff', u'sds_diff_cs', u'sunz', u
'lat', u'lon', u'time_offset', u'projection', u'product', u'version_id', u'sat_i
d', u'earth_sun_distance', u'annual_mean_solar_extraterrestrial_flux', u'iso_dat
aset']
>>> d.variables["earth_sun_distance"].shape
(96, 1)
>>> d.variables["earth_sun_distance"].dtype
dtype('float64')
>>> d.variables["x"].shape
(3712,)
>>> d.variables["x"].dtype
dtype('float64')
>>> d.variables["x"][:]
array([-5567.2481376, -5564.2477344, -5561.2473312, ..., 5561.2473312,
       5564.2477344, 5567.2481376])
>>> d.variables["y"].shape
(3712,)
>>> d.variables["y"][:]
array([ 5567.2481376, 5564.2477344, 5561.2473312, ..., -5561.2473312,
       -5564.2477344, -5567.2481376])
>>> █
```

After importing the data, the user can access and look at the variables in the data and can access them individually. Codes above shows that the number of variables in the data and user can check the shape and size of the variables with the codes shown above and then after close observation ingest that data which is valuable.

After this, the next step is to ingest the variables into Rasdaman and to do so, the file should be in the "tiff" format and to do that user has to convert the netCDF file into "tiff" format and this can be done by using "tiff.py" and to retrieve that one can use 'wget <http://www.lfd.uci.edu/~gohlke/code/tiff.py>'.

The main concept of the code is to make slices of the arrays and after making the slices convert the file into "tiff" format and then import that tiff file into Rasdaman.

```
rahul@rahul-VirtualBox: ~/Downloads
# STEP 2: Create empty marray
rasql $RASQL_ARGS -q "insert into $RASQL_COLL values marray it in [0:0,0:0] values 0f" """)

# load the file
d = Dataset(fn_in, 'r')

# the variables
variables = ['time', 'time_offset']

# read in each variable in one row as a float
out = np.array([
    d.variables[v][:].astype(float) for v in variables
])

# write the tiff file out
tiff.imwrite(fn_out, out)

print("""
rasql $RASQL_ARGS -q "update $RASQL_COLL as c set c[:,*] assign inv_tiff(\$1
)" --file %s && rm %s
""") % (fn_out, fn_out)
```

In the above code, it shows that the data is loaded and it takes the variables which are going to be ingested and the "data type" is also in float and after taking variable it converts the file into "tiff" format. After converting into the tiff, the file is ingested into Rasdaman.

**To ingesting the data into Rasdaman first step is to create a collection**

```
$ RASQL_ARGS -q "create collection $RASQL_COLL FloatSet"
```

**After creating a collection, an empty array is inserted into Rasdaman**

```
rasql $ RASQL _ ARGS -q insert into values marray it in [0:0,0:0] values 0f" """)
```

## Data Ingestion into Rasdaman

```
rahul@rahul-VirtualBox: ~/Downloads
rasql done.
rahul@rahul-VirtualBox:~/Downloads$ python script.py data.nc | RASQL_ARGS="--use
r rasadmin --passwd rasadmin" RASQL_COLL="test" bash
Traceback (most recent call last):
  File "script.py", line 80, in <module>
    tifffile.imsave(fn_out, out)
  File "/home/rahul/Downloads/tifffile.py", line 320, in imsave
    tif.save(data, **kwargs)
  File "/home/rahul/Downloads/tifffile.py", line 695, in save
    assert len(data.shape) in (5, 6)
AssertionError
rasql: rasdaman query tool v1.0, rasdaman v9.2.1-gd703cf7 -- generated on 31.03.
2016 00:18:46.
opening database RASBASE at localhost:7001...ok
Executing update query...rasdaman error 955: Update error 955 in line 1, column
1, near token create: Collection name exists already.
aborting transaction...ok
rasql done.
rasql: rasdaman query tool v1.0, rasdaman v9.2.1-gd703cf7 -- generated on 31.03.
2016 00:18:46.
opening database RASBASE at localhost:7001...ok
Executing insert query...ok
rasql done.
rahul@rahul-VirtualBox:~/Downloads$
```

Check that the data is inserted properly

```
rahul@rahul-VirtualBox: ~/Downloads
27026e-25,3.36919},{2.32049e+36,-3.35747},{2.32049e+36,3.35747},{-1.07655e+20,-3
.34575},{-1.07655e+20,3.34575},{4801.53,-3.33403},{4801.53,3.33403},{-2.14357e-1
3,-3.32231},{-2.14357e-13,3.32231},{1.00896e-29,-3.31059},{1.00896e-29,3.31059},
{5.32216e+31,-3.29887},{5.32216e+31,3.29887},{-2.3087e+15,-3.28715},{-2.3087e+15
,3.28715},{0.108658,-3.27543},{0.108658,3.27543},{-5.03686e-18,-3.26371},{-5.036
86e-18,3.26371},{2.24377e-34,-3.25199},{2.24377e-34,3.25199},{-1.97772e+15,-3.23
053},{-1.97772e+15,3.23053},{-4.10463e-18,-3.20709},{-4.10463e-18,3.20709},{9.96
286e+26,-3.18365},{9.96286e+26,3.18365},{1.9762e-06,-3.16021},{1.9762e-06,3.1602
1},{-nan,-3.13677},{-nan,3.13677},{-990440,-3.11333},{-990440,3.11333},{-2.1255e
-27,-3.08989},{-2.1255e-27,3.08989},{5.02135e+17,-3.06645},{5.02135e+17,3.06645},
{1.03796e-15,-3.04301},{1.03796e-15,3.04301},{-2.52761e+29,-3.01957},{-2.52761e
+29,3.01957},{4.23219e+31,-2.99225},{4.23219e+31,2.99225},{1.8261e-34,-2.94537},
{1.8261e-34,2.94537},{-9.26098e-23,-2.89849},{-9.26098e-23,2.89849},{4.6635e-11,
-2.85161},{4.6635e-11,2.85161},{-23.2862,-2.80473},{-23.2862,2.80473},{1.15088e+
13,-2.75785},{1.15088e+13,2.75785},{-4.54525e+10,-2.67193},{-4.54525e+10,2.67193},
{-1.06845e+34,-2.57817},{-1.06845e+34,2.57817},{0.091864,-2.46881},{0.091864,2
.46881},{-4.63923e-32,-2.28129},{-4.63923e-32,2.28129},{2.22114e+22,-1.93753},{2
.22114e+22,1.93753},{2.22114e+22,1.93753}
```

## 7 Conclusion

We couldn't be able to do more queries because of the limited time schedule, but we would love to work on this in future. This project was very challenging and motivating and we learned a lot of things from this, which would be very helpful in the future. In this project, we worked closely with EUDAT and KNMI Data Centre and this could be very helpful for us in making a base of our future as a "Data Scientist".