Development of Data Cube for Uttarakhand Using Geospatial Technologies

Indian Institute of Remote Sensing (IIRS), ISRO

Abstract: The Australian Geoscience Data Cube (AGDC) is an analytical framework developed by Geoscience Australia in collaboration with National Computational Infrastructure (NCI) and Commonwealth Space and Industrial Research Organization (CSIRO). It is an effective storage and retrieval method of large earth observation datasets and it supports analysis in the observational (spectral), temporal and spatial domains in a wide range of permutations. Access to the AGDC data holdings is currently only available via the API.
Our aim in this project is to implement this Data Cube for the Indian state of Uttarakhand.

I. INTRODUCTION

Scientists across different countries have felt in the past few years that they lack the knowledge, infrastructure and resources to access and use the growing space-based data. Traditionally, earth observation data from different satellites was stored as small scenes in individual files which could be used on desktop computers and workstations. This data accumulated over time resulting in terabytes and even petabytes of data in some cases. The earth observation data from different satellites offers great potential for analysis and understanding of the physical environment, reduce risk from natural hazards, predicting climatic changes and assist in securing new resources. However, it proved difficult to analyze and extract useful information from such large datasets. A multi-dimensional (space, time, data layers) Data Cube is an efficient and effective solution.

A data cube is an n-dimensional (generally three-dimensional) array of values which is used to represent data along some measure of interest. They are generally used to explain the time sequence of an image's data. It can be viewed as a collection of identical 2-D tables stacked upon one another. Each dimension represents some attribute in the database and the cells in the data cube represent the measure of interest.

The basic concept involved in a datacube is to form a cube from earth observation datasets by stacking satellite image tiles over time covering the same area of ground. It arranges spatial (2D) data temporally and spatially which allows flexible and large scale analysis. For this, the data cube uses the "**Dice and Stack**" method. Under this method, the data is divided into spatially regular, time stamped tiles which are then stacked in a way that the latest tile is at the top of the stack and the oldest at the bottom.

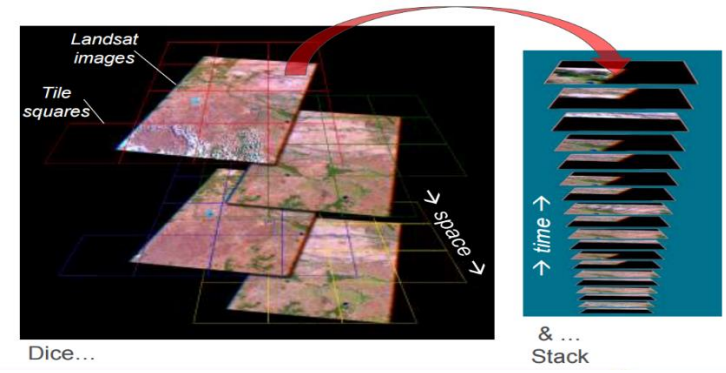

Figure 1 Dice and Stack method [1]

This makes temporal analysis and interpretation of data stored in the data cube much easier. The following are the characteristics of the data cube:

In contrast to other methods which are used to handle gridded data, every unique value is stored in a data cube. Thus, there is no loss of data.

There is calibration and standardization of data ingested in a data cube which makes it easier to access only the required data from a data cube. It also increases the value which can be derived from earth observation, and other sources of large gridded datasets.

II. DATA PREPROCESSING

A. Objective

To pre-process the Multispectral Remote Sensing (MRS) and Thermal Infrared Remote Sensing (TIRS) data of Landsat 4-5, Landsat 7 and Landsat 8 of Uttarakhand region required for the Data Cube.

Sub objective-

- Top of Atmospheric Reflectance (TOAR) and Normalized Difference Vegetation Index (NDVI) image generation for entire Uttarakhand region using Landsat 4-

5(TM), 7(ETM+), 8(OLI) Multispectral Remote Sensing data (Red, Green, Blue and Near Infrared bands).

- Land Surface Temperature (LST) image generation for entire Uttarakhand region using Landsat 4-5(TM), 7(ETM+), 8(OLI) Thermal Infrared Data.

B. Datasets required for Data cube generation

Multispectral Remote Sensing (MRS) and Thermal Infrared Remote Sensing (TIRS) data of Landsat 4, 5, 7 and 8.In this study, MRS and TIRS data is used for generating Reflectance, NDVI and LST images. This data is downloaded from United State geological Survey (USGS) website. [2]

C. Software Used

- ENVI 5.1
- ERDAS Imagine 2014

ERDAS IMAGINE and ENVI are image processing software which can be used for pre-processing of satellite imagery. By manipulating imagery data values and positions, it is possible to see features that would normally be invisible and locate geo-positions of features that would otherwise be graphical.

D. Methodology:

1. Flow chart

The following flow charts represents the steps to generate Top of Atmosphere Reflectance, NDVI and Land Surface temperature from MRS and TIRS dataset.

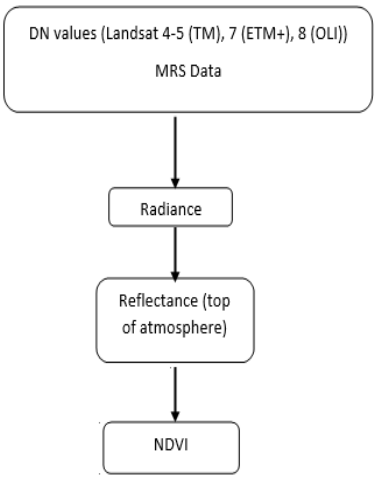

Figure 2 Flow chart: Steps to generate NDVI and TOAR

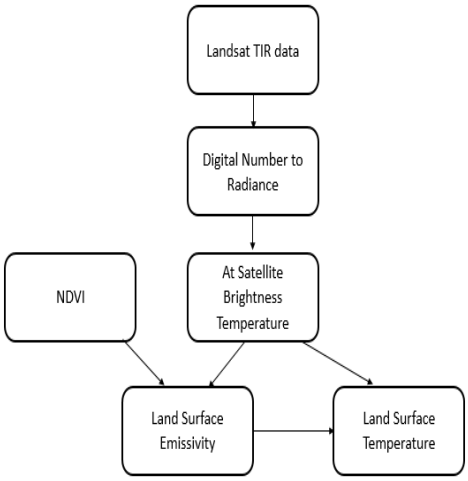

Figure 3 Flow chart: Steps to generate LST

2. DN to Radiance Conversion

Un-calibrated MRS Dataset in DN, obtained from Landsat 4-5(TM), 7(ETM+), 8(OLI) is converted to Radiance images using the Gain and bias method [3]. To derive a radiance image from an un-calibrated image, gain and offset values are retrieved from the image's metadata file, and radiance is calculated as:

$$L_\lambda = \text{gain} * DN + \text{bias}$$

3. Radiance to Top of Atmospheric Reflection (TOAR)

Radiance obtained from previous step is further used to generate TOAR using the following formula [4]:

$$R = \frac{\pi * L_\lambda * d^2}{\sin\theta * E_{sun\,\lambda}}$$

Where:

θ = solar elevation angle.

R =Top of atmosphere reflectance,

L_λ= spectral radiance,

d = Earth-Sun distance in astronomical units

$E_{sun\lambda}$ = mean solar exoatmospheric irradiances

4. Normalized difference vegetation index (NDVI)

TOAR (Red and NIR bands) images thus obtained in the previous step are further used to generate NDVI. It ranges from -1 to 1. It is calculated using the following formula [5]:

$$NDVI = (NIR-RED) / (NIR+RED)$$

5. Generation of Land Surface Temperature

a) Conversion to At-Satellite Brightness Temperature
TIRS band data can be converted from spectral radiance to brightness temperature using the thermal constants provided in the metadata file [6]:

$$T_B = \frac{k_2}{\ln\left(\frac{k_1}{L_\lambda} + 1\right)}$$

Where:

T_B = At - satellite Brightness temperature (K)

L_λ = TOA Spectral Radiance (Watts / m^2 * str * μm)

K_1 = Band – specific thermal conversion constant from the metadata (K1_CONSTANT_BAND_x, where x is the thermal band number)

K_2 = Band – specific thermal conversion constant from the metadata (K2_CONSTANT_BAND_x, where x is the thermal band number)

b) Calculate Emissivity using NDVI generated
NDVI generated previously is further used to calculate emissivity using the following formula [6]:

$$\varepsilon = \varepsilon_s (1 - P_V) + \varepsilon_v P_V + d\varepsilon$$

Where, $P_v = \left[\frac{NDVI - NDVI_{min}}{NDVI_{max} - NDVI_{min}}\right]^2$ **, and** $d\varepsilon = (1 - \varepsilon_s)(1 - P_V) F \varepsilon_v$.

Where,

ε_s and ε_v are emissivity of pure soil and pure vegetation respectively,

F is shape factor which is taken as 0.55.

$$\varepsilon = 0.004 P_V + 0.986.$$

c) LST Generation using Emissivity
At-Satellite Brightness Temperature and emissivity calculated is further used to generate LST using the following formul [6]:

$$S_t = \frac{T_B}{1 + (\lambda X T_B / \rho) \ln \epsilon}$$

Where, $\rho = \frac{hc}{\sigma}$, where h, c and σ are Plank's constant, speed of light and Boltzmann constant respectively.

And $\lambda = Wavelength\ of\ emitted\ radiation = 11.335\ \mu m$

And ϵ is emissivity, and $T_B =$ Brightness temperature.

E. Mosaicking and Sub-setting of images:
Small images of different tiles need to be mosaicked, combined into a single image in GCS co-ordinate system.

Steps for Mosaicking:

- Select the input files
- Select the data values to ignore (background pixels)
- Set the Scene order
- Feathering
- Colour Correction

Sub-setting
Cut the mosaicked image with shape file of Uttarakhand.

F. Pre-processing Results

Figure 4 TOAR of Uttrakhand region

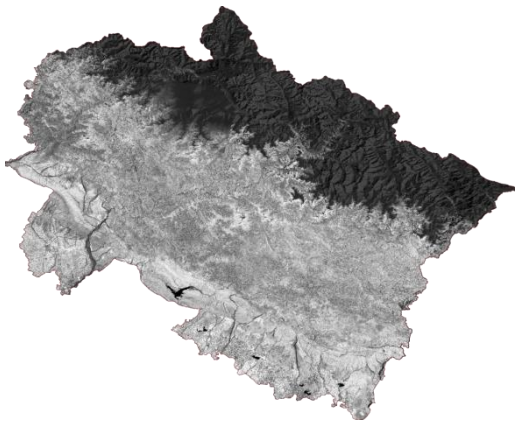

Figure 5 NDVI image of Uttarakhand region

G. Pre-processing conclusion

- Stored top of atmospheric reflectance file in '.dat' format.
- Stored top of atmospheric reflectance UTM (float32) BAND WISE in '.Gtiff' format.
- Stored NDVI file in '.dat' format.
- Stored NDVI UTM (float32) file in '.Gtiff' format.
- Stored Brightness Temperature in '.dat' format.
- Stored Land Surface Temperature in '.dat' format.
- Stored LST UTM (float32) file in '.Gtiff' format.

III. Data Storage Formats

Some of the problems currently being faced are that the data we have is very large (~100GB for only Uttarakhand) and its organization is complex. Moreover, the data is heterogeneous (Reflectance, NDVI, LST). Therefore, we can't use any legacy image/file formats to store our data, since they don't provide efficient enough access capabilities. The formats suitable for storing the data for the datacube are NetCDF (Network Common Data Form), HDF5 (Hierarchical Data Form 5) and ECW (Enhanced Compression Wavelet).

A. NetCDF (Network Common Data Form)

The biggest advantage of NetCDF is that it supports efficient sub-setting [7] and Array oriented data access (direct access), creation and storing. NetCDF Data also becomes Self-describing by having a 'header' to store the metadata. NetCDF is also not machine specific and therefore sees a wide adoption rate in the industry. NetCDF is great also because it supports concurrent readers.

The only problem with legacy NetCDF (version 3 and lower) is that there is a limit on external numeric data types and the maximum possible file size (2GB). Also only one unlimited dimension is permitted for each data set. [8] [9]

B. NetCDF-4

NetCDF4 is the updated version of Legacy NetCDF. It is basically an abstraction over a HDF5 file system i.e., the base of NetCDF4 file is a HDF5 file. It overcomes most of the problems of the legacy NetCDF by combining features of HDF5 with NetCDF(<4). It supports more numeric data types and has no limit on size of dataset. Introduction of hierarchical groups allows users new ways to organize data. It also supports parallel read/write access to netCDF4/HDF5 files. (using the HDF5 libraries).

NetCDF performance analysis

The paper "NetCDF-4 Performance Report" by HDF-Group, written by Choonghwan Lee, MuQun Yang and Ruth Aydt provides detailed performance analysis of Legacy NetCDF and NetCDF4 and suggests various benchmarks, performance comparisons and optimizations and tunings possible in netCDF4. [8]

The summary of the NetCDF performance comparison is given in figure 7.

Storage Layout	System Caching	Figure	Test Array	Write				Read			
				Rate (MB/s)		Best	% Faster	(Rate MB/s)		Best	% Faster
				netCDF3	netCDF4			netCDF3	netCDF4		
Contiguous	Default	1	Large	351	279	netCDF3	26%	334	424	netCDF4	27%
			Small	317	255	netCDF3	25%	321	385	netCDF4	20%
	Override	2	Large	41	42	netCDF4	3%	40	27	netCDF3	47%
			Small	11	11	netCDF4	4%	38	42	netCDF4	9%
Chunked	Default	3	Large	339	153	netCDF3	121%	331	431	netCDF4	30%
			Small	343	321	netCDF3	7%	315	223	netCDF3	41%
	Override	4	Large	37	42	netCDF4	13%	42	44	netCDF4	5%
			Small	10	10	netCDF3	9%	40	39	netCDF3	2%

Figure 7 – Summary of NetCDF performance comparison

As it is clear from the figure, NetCDF4 performs better overall during reading of data.

C. HDF5 (Hierarchical Data Form 5)

HDF5 simplifies the file structure to include only two major types of objects - Datasets, which are multidimensional arrays of a homogeneous type and Groups, which are container structures which can hold datasets and other groups. NetCDF4 is basically an abstraction over HDF5. So all the benefits of NetCDF4 are available in HDF5 also. It is widely in use and has an expanding in the industry. User can also provide information about the data (metadata) in the form of headers. It also supports chunking of data. HDF5 also has a widely used parallel implementation that can support non-rectangular data structure [10]. The only downside is that it only has a single C implementation in the open standard.

1. *Performance Analysis with NetCDF.*

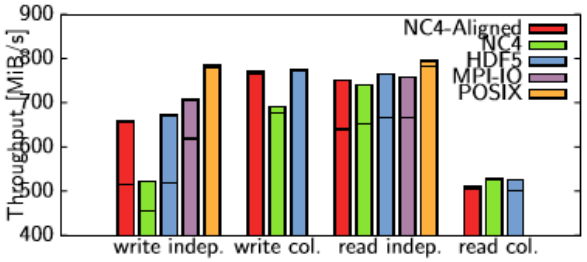

Figure 8- Performance analysis with disjoint pattern (higher is better) [11]

As the Figure shows, HDF5 and NetCDF4 are very similar in reading performance when the disjoint pattern of HDF5 is taken. The high variation in

independent I/O vs collective I/O is due to lack of synchronization.

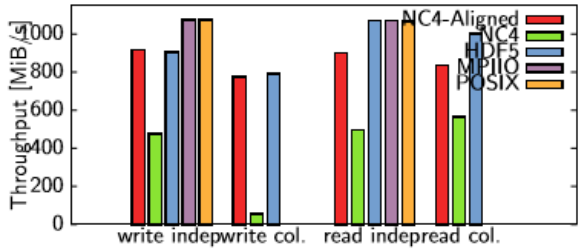

Figure 9- Performance analysis with OST pattern

As given in Figure 9, the performance is much more stable as compared to Disjoint pattern. Clearly HDF5 performs better than NetCDF4. NetCDF4 API without the alignment patch is much worse than the other APIs because of unaligned access.

The following figure shows the comparison between Parallel-NetCDF and HDF5.

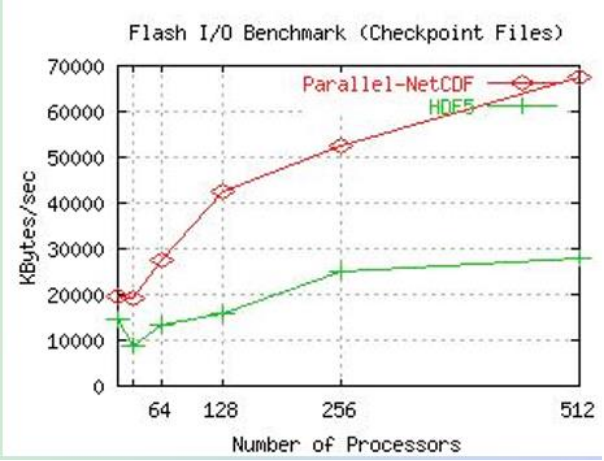

Figure 10 – Performance analysis of P-NetCDF with HDF5 [12]

D. ECW (Enhanced Compression Wavelet)

It is a proprietary wavelet compression image format optimized for aerial and satellite imagery. It is a lossy compression format. Very efficient compression. But lossy compression of irreversible compression uses inexact approximations and partial data discarding to represent the content, or in other words ECW is not lossless. But it has extremely high compression rations from 1:2 to 1:100.

E. Data Formats Conclusion

Preliminary assessment of papers and applications related to Data-Cube suggests that NetCDF4 should be the most suitable format for our needs as it incorporates the advantages of both NetCDF and HDF5. HDF5 browser plugins can be used to efficiently display the images on the internet.

IV. DATACUBE ARCHITECTURE

A. Overview

The Datacube is intending to manage thousands of billions of observations, and allow applications to access and analyse each individual observation through its application programming interface (API). Key requirements include:

- Efficient processing and delivery of large-scale, temporally-deep data collections
- A platform to manage all forms of multidimensional, regular gridded data across diverse domains (Spatial and Temporal)
- Avoid duplication of data in different structures for different purposes
- Endpoint access APIs according to different usage criteria

B. Architecture

1. Data Cube Core:

The core components of the Data Cube are:

- Analytical Engine- This engine is responsible for the creating an execution plan to perform analysis on a specified volume of information using the available computational resources.
- Execution Engine-This engine executes the plans created by analytical engine and produces outputs of various requested analyses.
- Ingester- This component ingests the indexed analysis ready data into the datacube by forming NetCDF files and updating the datacube index.
- Visualisation- The visualization module plots the data and provides a graphical visualisation of the data using 3rd party libraries like matplotlib and glue.
- N-Dimensional Array Interface- This is an abstraction for data handling in the datacube as the entire data is a large multi-sensor, spatio-temporal and other attribute array in n-dimensions.
- Data Management and Access Module- It consists of three main components:
  - Index Store – the database contains query able indexing and metadata information across all Storage Units.
  - Storage Units – the actual files containing the EO data

- Data Collections – a combination of Index Store and Storage Units.

- Reporting- This module is used to report about the data contained and the various products available in the datacube.
- Discovery and Delivery- This component provides query, filter and data delivery functionality for applications.

2. UI & Application Layer

This layer represents all the third party developer applications supplied to users. For example, the web interface that was built on top of the datacube to access and view the data.

C. External Interfaces to Datacube

1. Ingest interface

- Used to ingest analysis-ready-data to the Datacube. Interacts with the Data Management and Access Module to populate the index store with metadata and store the multidimensional Storage Units.
- **Data flow:** The interface is required to support the storage of terabytes of data and millions of metadata record updates for a complete replacement or new Data Collection. In addition the interface provides methods for appending or updating individual Storage Units and associated metadata records.

2. Datacube Application Programming Interface (API)

- A Python programming interface to the Datacube for Application Developers. Provides an n-dimensional array abstraction and access to the Analytics Engine and Data Management and Access Module.
- **Data Flow:** Generally the Datacube API is designed to work with the data and analytics stored on the same high performance cluster. The only external to Datacube data flow occurs when the API is used to obtain data or results and store this outside of the Datacube. It is the application developer's responsibility to manage all external data flow under these circumstances. Internal to Datacube data flows can be extensive and is covered later in this document.

3. Analytics and Exploratory Data Analysis API

- Available to both Users and Application Developers provides a set of functions suitable for high level analysis and exploration of the Datacube data built upon the Datacube API.
- **Data Flow:** Similar to the Datacube API, Analytics and EDA is performed on the Datacube Cluster. External data flow occurs when the API is used to obtain data or results and store them outside the Datacube. It is the users' responsibility to manage all external data flow.

V. DATACUBE IMPLEMENTATION

Requirements:

1. Analysis Ready Data (ARD) (Band Separated)
2. Various Metadata (Reference examples in appendix)
3. Datacube and dependencies installed

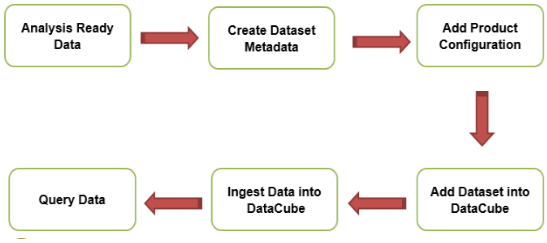

Figure 11 Working of a Datacube

Step - 1: Create Dataset Metadata

Using the preparation script in the utilities, the metadata (yaml file) that is required for adding the dataset into the Datacube is generated from the corresponding MTL text file of every scene by picking up the required data about satellite and sensors from text file and header information from the Tiff files.

Step - 2: Adding Product Configuration

According to the product that is required to add into the datacube, the corresponding configuration file (.yaml) should be fed into the datacube. This creates indexes in the database of various products that the datacube will be containing.

Step - 3: Adding Dataset into Datacube

The dataset needs to be added into the datacube. What this step does at the back-end is creation of indexes of the tiff files corresponding to that dataset in the database of the Datacube.

Step - 4: Ingesting Data into Datacube

This is a very crucial step in the process as this is where the core function of datacube gets executed. The dataset added (indexed) needs to be ingested using the ingestion metadata. During ingestion, the

bands (Presently tiff files) get aggregated and split into spatial tiles and saved as separate NetCDF files. The ingestion metadata consists of the bounds for ingestion, tile size etc. These NetCDF files are indexed into the database along with their tile bounds so that when a query is performed with a specific region the corresponding tiles are accessed (or a subset of a tile, if the bounds are within one tile). After formation the archival system stores these files in the archive.

Step - 5: Query and Analysis

Once the data is ingested the datacube is ready for queries and analysis. Data query can be done using the API provided in python. In order to perform analyses on the data, analysis and execution engines can be used. This is further elaborated in the analysis API section of the report.

B. Decompression Time Analysis of multi-threaded compression techniques

Table 1 Decompression time-Multithreaded-HPC-.nc

	Decompression time(in sec)		
Compression technique	**C Index: 1**	**C Index: 5**	**C Index: 9**
lbzip2	1.52	1.48	1.59
Pigz	0.85	0.78	0.81
pigz_zlib	0.84	0.81	0.81
pbzip2	0.47	0.55	1.59

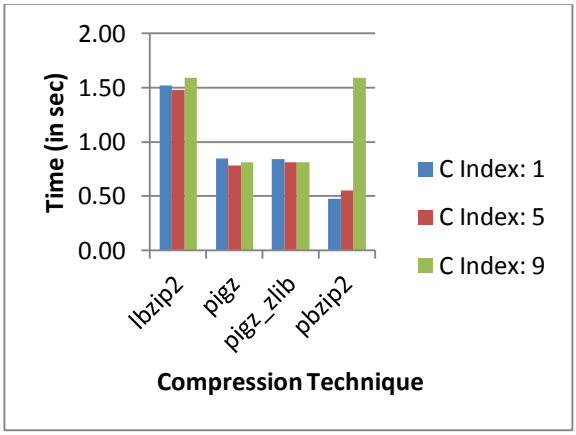

Figure 12 Decompression time-Multithreaded-HPC-.nc

Pbzip2 (index 1 and 5) take the least time for decompression, which will best serve the purpose of quick data retrieval. Therefore, Pbzip2 (index 5) is used in the Archival system for compression and decompression.

C. Archival System

A module of archival system is added on top of the ingestion framework of the datacube. As per the results of the performance analysis of compression algorithms, pbzip2 is used to compress the files. This Archival System mainly does 3 tasks:

1. Compression
2. File Lookup
3. Decompression

1. Compression

As soon as the ingestion process is completed the NetCDF files formed are compressed and stored into an archived directory (As shown in **Error! Reference source not found.**). This directory consists of individual NetCDF files compressed using pbzip2 with a compression index of 5.

2. File Lookup

As and when a user queries data, the archival module looks in the cache (As shown in **Error! Reference source not found.**) for that particular NetCDF file. If it exists, it returns it to the data processing module to create an array out of it. Else, it indicates to the archival system of the need to extract the file from the archive.

3. De-compression

The files stored in the archive are decompressed and moved into cache as and when required by the users. After a query by the user, file lookup happens as explained above. If file lookup requests an extraction then it goes to the archive and extracts the required file and moves it into the cache and then returns the file. In this way, if the same query is made again the file now stored in cache is returned and hence the response will be quicker.

D. Web Interface

The web interface for Datacube provides the users with a GUI that is accessible from any browser to interact with the Datacube. The back-end of this website is built on Django framework connects the inputs received from the UI (Front-end) to the Datacube and performs the query with the given parameters. Then the data output is converted into image using plots of matplotlib and then this image is served on the front end.

The web portal created to access the data cube is shown below in Figure 13.

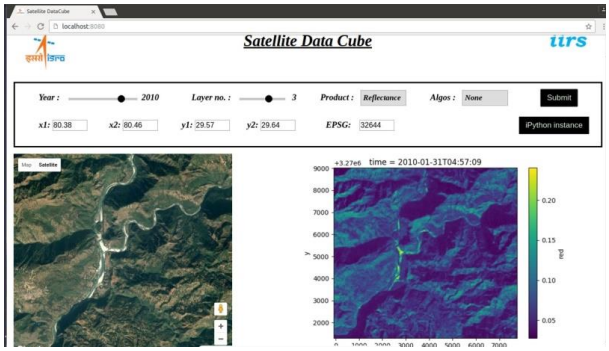

Figure 13 Web Portal

If the user wants to access the advanced features or perform other complex analysis, he/she can click the iPython button to access an instance of iPython running on the server in the Datacube environment. The available notebooks are displayed when the iPython instance button is selected by the user in the web portal. These notebooks enriches the user with examples of advanced usage, so that the user can learn to perform other different complex analysis by creating similar notebooks.

REFERENCES

[1] "datacube.org," 2016. [Online]. Available: http://www.datacube.org.au. [Accessed 5 July 2016].

[2] "EarthExplorer," 2016. [Online]. Available: http://earthexplorer.usgs.gov/. [Accessed 5 July 2016].

[3] "Landsat DNs to ToAR conversion," 2016. [Online]. Available: http://yceo.yale.edu/how-convert-landsat-dns-top-atmosphere-toa-reflectance. [Accessed 5 July 2016].

[4] "How to convert Landsat DNs to Top of Atmosphere (ToA) Reflectance," 2016. [Online]. Available: http://yceo.yale.edu/how-to-convert-landsat-dns-top-atmosphere-toa-reflectance. [Accessed 8 July 2016].

[5] "Normalized Difference Vegetation Index," 2016. [Online]. Available: https://en.wikipedia.org/wiki/Normalized_Difference_Vegetation_Index. [Accessed 8 July 2016].

[6] Q. Weng, D. Lu and J. Schubring, "Estimation of land surface temperature-vegetation abundance relationship for urban heat island studies.," *Remote Sensing of Environment*, vol. 89, no. 4, pp. 467-483, 2004.

[7] "unidata.ucar.edu performance," 5 July 2016. [Online]. Available: https://www.unidata.ucar.edu/software/netcdf/docs/netcdf_introduction.html#performance. [Accessed 5 July 2016].

[8] C. Lee, M. Yang and R. Aydt, "NetCDF-4 Performance Report," 2008.

[9] "unidata.ucar.edu limitations," 5 July 2016. [Online]. Available: https://www.unidata.ucar.edu/software/netcdf/docs/netcdf_introduction.html#limitations. [Accessed 5 July 2016].

[10] hdf5group, "hdfgroup.org/HDF5/," 5 July 2016. [Online]. Available: https://www.hdfgroup.org/HDF5/. [Accessed 5 July 2016].

[11] B. Christopher, C. Konstantinos and L. Thomas , "A Best Practice Analysis of HDF5 and NetCDF-4 Using Lustre," 2015.

[12] R. Ross, "Parallel NetCDF: A Scientific High-Performance I/O Interface," 2003.

[13] "Satellite-derived land surface temperature:Current status and perspectives," 2016. [Online]. Available: http://www.sciencedirect.com/science/a. [Accessed 5 July 2016].