TECHNISCHE UNIVERSITÄT DRESDEN

DLR

Faculty of Environmental Sciences Institute for Cartography

WEB MAPPING APPLICATION FOR OPERATIVE FIRE AND WATER SERVICES

Thesis is submitted to the Technical University in Dresden
and the German Aerospace Center
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE
IN CARTOGRAPHY

MOSTAFA ELFOULY

Supervisors:
Prof. Dr. Manfred F. Buchroithner
Dipl. -Geogr. Stefan Plattner, DLR
Dr. Christian Strobl, DLR

Dresden, 2014

BIBLIOGRAPHIC-DOCUMENTALISTIC INFORMATION AND ABSTRACT

Author:	Mostafa ElFouly
Supervisor:	Prof. Dr. Buchroithner Manfred, TU Dresden
Co-advisor:	Stefan Plattner, Dipl.-Geogr., DLR; Dr. Christian Strobl, DLR
Title:	Web Mapping Application for Operative Fire and Water Services
Key words:	Web Mapping, MODIS, Fire Hotspot, Fire-affected areas, Water Services

Abstract

This thesis' scope describes requirements and implementation for web-based processing and workflow creation for the retrieval of remote sensing products like fire hotspots and water services based on National Aeronautics and Space Administration (NASA) Moderate Resolution Imaging Spectroradiometer (MODIS) data. Service chains to retrieve the remote sensing products for the desired regions is implemented where statistical analysis can be applied after.

The work is conducted in the context of the Environmental and Crisis Information System (UKIS) which is being developed at German Remote Sensing Data Center (DFD) as a modular framework and allows for the construction of information systems that rely both on remote sensing data produced at DFD and external web-enabled data services (e.g. water gauges), thus providing higher-level information for situation awareness and decision support.

A MODIS Web service which provides users with subsets of MODIS Land Products through standard based Simple Object Access Protocol (SOAP) Web service is called within a Python-based Representational State Transfer (REST) module to get the available dates for the selected MODIS product level. A separate Python-based REST module script is built to search for available satellite data that will match the selected tile numbers as well as the selected dates.

The retrieved MODIS data will get exposed to the process chain where the higher-level MODIS products will be generated.

An interface to retrieve and expose the retrieved archived data to the process chain was integrated; the user of this system can select interactively the desired region that he/she would like to get archived satellite data for according to the date period and product level and process it with the developed processing services and workflows.

This test application automates the retrieval of NASA - MODIS data through a given range of dates built in separate modules that can be reused in different applications. Visualization of the retrieved MODIS data should allow us to apply statistical analysis where it can be used for studies of processes and trends on local to global scales.

ACKNOWLEDGEMENT

I would like to express my appreciation to my advisory committee: Prof. Dr. Manfred Buchiroithner, Stefan Plattner, Dipl.-Geogr., and Dr. Christian Strobl. Thanks for giving me the opportunity to be part of the Earth Observation Center Research Team. Also thanks to Enrico for his help with the MODIS data archives. Stefan, it has been an honor to work with you. Also, thanks to Stefan Peters, Dipl.-Ing. the Cartography Masters program coordinator for his precious advices and supporting me through my studies.

My gratitude also goes to Tatjana Kutzner, Dipl.-Inf. who was always supporting me through my whole studying period giving me advices and giving me the chance on working on different tasks that helped me in improving my skills. My sincere thanks go to Dr. Bernhard Jenny and Dr. Helen Jenny for giving me the opportunity to join the Cartography and Geovisualization lab at Oregon State University and working on research projects which enhanced my technical skills. Thanks to my friends Abdullah, Abdulrahman, Bojan, Farid, Jillian, Johannes, Mahmoud, Omar, Pakinam and Sebastian for being great friends.

I would especially like to thank Aleksandra for her encouragement and moral support, this thesis would not have been accomplished without her.

Last but not least, I would like to thank my parents, Eman and Motie ElFouly, and my brother, Mohamed, who have genuinely contributed to the development of my personality.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

NASA	National Aeronautics and Space Administration
MODIS	Moderate Resolution Imaging Spectroradiometer
UKIS	Environmental and Crisist Information System
DFD	German Remote Sensing Data Center
SOAP	Simple Object Access Protocol
REST	Representational State Transfer
EOS	Earth Observation System
HTTP	Hyper Text Transfer Protocol
FTP	File Transfer Protocol
TDRSS	Tracking and Data Relay Satellite System
EDOS	EOS Data and Operations System
MODAPS	MODIS Adaptive Processing System
DAAC	Distributed Active Archive Center
OCDPS	Ocean Color Data Processing System
SWIR	Shortwave Infrared
NIR	Near Infrared
TIR	Thermal Infrared
JSON	JavaScript Object Notation
HDF	Hierarchical Data Format
GML	Geography Markup Language
GeoTIFF	Geospatial Tagged Image File Format
MVC	Model-View-Controller
LAADS	Level 1 and Atmosphere Archive and Distribution System
NSIDC	National Snow and Ice Data Center
LP DAAC	Land Processes Distributed Active Archive Center
EROS	Earth Resource Observation and Science
OCDPS	Ocean Color Data Processing System
GSFC	Goddard Space Flight Center
USGS	U.S Government office
DEM	Digital Elevation Model
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer
AIRS	Atmospheric Infrared Sounder
AMSR-E	Advanced Microwave Scanning Radiometer-EOS
AMSU-A	Advanced Microwave Sounding Unit
CERES	Cloud's and the Earth's Radiant Energy System
HSB	Humidity Sounder for Brazil
MOPITT	Measurements of Pollution in the Troposphere
MISR	Multi-angle Imaging SpectroRadiometer
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer

STATEMENT OF AUTORSHIP

Herewith I declare that the thesis entitled

Web Mapping Application for Operative Fire and Water Services

which has been submitted to the study commission of geosciences today is my own work. I have fully referenced the ideas and work of others, whether published or un-published. Literal or analogous citations are clearly marked as such.

Munich, January 28, 2014

Mostafa ElFouly

1 INTRODUCTION

Fire is a main element in the Earth system change and it occurs mainly in vegetation zones all over the world [1]. Hence Accurate monitoring of fire-affected areas in respect to time has became crucial part of understanding the changes that occur to the Earth system [2].

Recognition of the burned or fire-affected areas can also be treated as change detection issue. Variations in both the surface state and those imposed by the sensing system makes it difficult for the remote sensing algorithms developed to map large fire-affected areas [1].

The availability of robustly calibrated, atmospherically corrected, cloud-screened and geolocated data provided by the latest generation of moderate resolution remote sensing systems allows major advances in satellite mapping of fire-affected area. DFD provides operational, satellite-based crisis information including a wildfire hotspot service and a flooded areas service based on MODIS near real-time data for Europe.

In this thesis we designed a web-based system for the visualization of products generated by these services in conjunction with the adequate geo-basis information. Products include both vector and raster data of fire hotspots, flooded areas, receding water and cloud masks. A process of making use of radar data acquired by DLR-operated TerraSAR-X and TanDEM-X satellites is currently under development.

1.1 Motivation

Fire is an important ecosystem process affecting vegetation structure and composition [3] and in many land use systems is a proximate cause or indicator of land cover change [4]; [5]; [6]. It remains unclear if fire regimes will change as human population, their land use practices, and the climate change [7];[8];[9].

Granting reliable fire information to scientists, resource managers and policymakers becamse necessary because of such matter [1].

It is well established that for most fire regimes satellite active fire detections do not reliably define the fire affected area [1]. This is because the satellite may not overpass sufficiently frequently to capture the spatial details of how fires propagate across landscapes, and because clouds and optically thick smoke may preclude active fire detection [10];[11]. Usage of multitemporal satellite data, which provides several advantages over single date data for mapping fire-affected areas was the solution for decades [12]; [13].

The launch of NASA's Terra satellite in late 1999 and later Aqua satellite on May 4, 2002 marked a significant step forward in the ability to monitor fires from space [14]. The satellite's sensor payload includes the MODIS, an instrument having 1 km middle and long-wave infrared bands designed specifically for the observation of actively burning fires [15]. It has 36 spectral bands of image data [16]. In addition to offering enhanced fire detection, these bands permit, for example, low-intensity surface fires to be distinguished from higher-intensity crown forest fires [17];[18]. A second MODIS instrument on NASA's Aqua satellite, launched in mid-2002, provides an additional pair of observations (EOS AM) [19].

Hence, it became important to build an Archive/Catalog connector to enable, automate and ease the visualization of the data generated by the wildfire hotspot service and flooded areas. The retrieved data has to pass through the process chain, saved in the database and then visualized at the very front end over the map. Increased availability of real-time sensor data at the local scale could increase the understanding and detection of vegetation status of heterogeneous landscapes [20].

1.2 Problem overview and objective of the thesis

Higher levels of MODIS data take few days to be available. While MODIS Level 1 is retrieved on daily basis, the process of generation of MOD09A1 needs 8 days to be generated. The process chain introduced by Strobl [21] facilitates the generation of daily based MOD09A1 and fire hotspots. Where MODIS Level 1 products are available on daily basis, the desired ones get exposed to the process chain in which higher level products will be generated. Which allows us to get higher level products for any specific date.

Within the scope of the thesis, the web-based user interface for the visualization of products generated by these services in conjunction with the adequate geo-basis information is being built allowing the remote sensing scientists to easily download MODIS data of selected areas. Different MODIS products levels are available for being downloaded where they will get exposed after to the process chain and finally being visualized to the user.

1.3 Organisation of the thesis

This thesis consists of four main chapters. In the first chapter short introduction, motivation, problem overview and objective of the thesis is given. The second chapter describes the detailed theoretical background, definition of MODIS, MODIS fire, MODIS limitations, Terra and Aqua satellites and spatial data rertrieval. The thechnical implementation, the system architecture and the complete source code is given in the third chapter. Chapter four tackles the conclusion and summarizes the output of this thesis.

2 THEORY

The first section will give us a deep understanding of what MODIS sensor is. This section is divided into six subsections.

In the first sub-section we talk about a unique feature that is provided by the MODIS instrument which is Direct Broadcast. While in the second sub-section we discuss the available MODIS products. The third sub-section will focus on which MODIS data level we will get the MODIS fire hotspots. The fourth sub-section we are going to show the MODIS limitations. The last two sub-sections will talk about the satellites in which the MODIS instrument is attached to.

In the second section we talk about how to retrieve the Spatial Data.

2.1 Moderate Resolution Imaging Spectroradiometer (MODIS)

MODIS is a sensor carried on both the Terra and Aqua satellites [22]. It allows to obtain images in the morning (Terra) and in the afternoon (Aqua) for any distinct location [16]. It achieves 2,330-km swath which provides near-daily global coverage and have transformed the ways we study and monitor the Earth [2]. It provides high radiometric sensitivity in 36 spectral bands ranging in wavelength from 0.4 µm to 14.4 µm [23] for measuring visible and infrared radiation [24].

One of the MODIS sensor's strong attributes is that its data spatial resolution are between 250 meters to 1 kilometer [25]. At a nominal resolution of 250 m at nadir two bands are visualized, five bands at 500 m, and the rest 29 bands at 1 km [23].

MODIS data will improve our understanding of measuring land cover, vegetation, land use change, fire occurrence, volcanoes, snow cover and cloud properties [24]. Global interactive Earth system models is partially developed by MODIS sensor. It is capable of guessing global changes precisely which helps policy makers in making decisions concerning the protection of the environment [22].

Tracking and Data Relay Satellite System (TDRSS) is used to transfer the MODIS data - as well as all the data from other remote sensors on board the Terra and Aqua spacecrats - to the ground stations in New Mexico. MODIS Adaptive Processing System (MODAPS) produces the Level1A, Level1B and the higher-level MODIS land and atmoshphere products. The output of the MODAPS is then handed over among three Distributed Active Archive Centers (DAAC) for distribution.[26].

MODIS data have proven useful in near real-time monitoring of natural disturbance at large scales such as fires [1], insect outbreaks [27] and drought, while it is difficult to monitor land changes with MODIS daily gridded products [28].

Even when the land surface is not changing, the spectral signatures associated with the same grid cell might change [2].

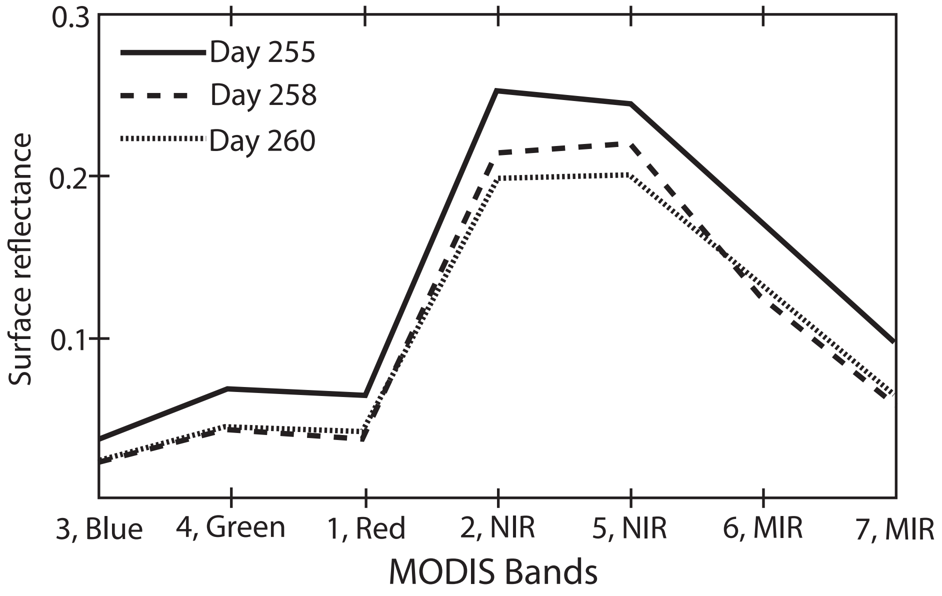

Figure 2.1: The spectral signatures of cloud-free observations (source:[2])

2.1.1 Direct Broadcast

Direct Broadcast capability is one of MODIS' sensor unique features. MODIS sensor is able to store data for later download at nominated intervals. Direct Broadcast allows MODIS to immediately broadcasts the raw data collected to any ground station having the right equipments. Terra was from the first satellites to apply the direct broadcast capability [29].

2.1.2 MODIS Data Products

MODIS products are divided into 44 data sets. These 44 data sets are placed into one of four categories. Lower-level products are used as an input and the higher-level ones are the output. The higher the product level, the more complicated the products get to be. The level 1A data product is produced from the MODIS detected bounced radiance beam that hits the earth and is reflected back from the earth and its atmosphere. This stream of beams are sent one after another, along the day, on daily basis. Geolocation and Calibration have to be performed in order to make use of these raw radiance numbers [30].

Calibration aims to improve the sensor performance by discarding the structural error and the known sources of interference. Where the structural error is the difference between the expected output and the measured output [31]. Geolocation is the process of detecting the location of the atmosphere or Earth's surface of a specific signal [30]. Calibration and Geolocation processes take place in the MOD02 Level 1B and MOD03 data sets where they form the footing for all the other MODIS products. Which is why Calibration is of extreme important because any minor error on the MOD02 or MOD03 all the subsequent products will have the same error [30].

The MOD01 (MODIS Level 1A) data set are used as input for calibration and geolocation for the MOD02 (MODIS Level 1B) generation. It contains 36 MODIS channels. Missing or bad pixels are indicated via quality indicators. During the daytime, Visible Shortwave Infrared (SWIR) and Near Infrared (NIR) measurements are made. While along the day and the night time radiances for the Thermal Infrared (TIR) are measured [32].

Terra file names for a complete file begin with MOD. As for Aqua file names begin with MYD [16]. Our focus in this thesis is:

- **MOD02/MYD02**
  Level-1B Calibrated Geolocation Data Set MOD02 (MODIS Level 1B) is the output of the processed MOD01 (MODIS Level 1A), hence it contains the calibrated and geolocated radiances for the 36 bands. By knowing the solar-diffuser data and the target illumination geometry reflectance for the solar reflective bands can be determined [33]. Some additional data are provided like error estimates and calibration data. Radiance unit is $W/(m^2 - \mu\text{m} - sr)$.

- **MOD03/MYD03**

  MOD03 - the Geolocation Data Set - is the Geolocation product which contains geodetic coordinates (latitude, longitude), sensor zenith angles, solar zenith angles, Elevation and Land/Water mask for each MODIS 1km sample [34]. The geolocation fields are found out using the instrument telemetry, spacecraft attitude and orbit, and DEM [35].

- **MOD09/MYD09**

  Surface Reflectance; Atmospheric Correction Algorithm Products

  The MODIS Surface Reflectance products provide an estimate of the surface spectral reflectance as it would be measured at ground level in the absence of atmospheric scattering or absorption.

  MOD09 products can be whether MOD09CQ, MOD09Q1, MOD09GA or MOD09A1. The scope of this thesis will be MOD09A1.

- **MOD09A1/MYD09A1**

  MODIS Surface Reflectance 8-Day L3 Global 500m file, provides Bands 1–7 and is composed of eight successive daily 500 m images. Since MODIS is an optical sensor and not a microwave one, so clouds might be misclassified as water or ice, so in order to eliminate clouds from the scene, Each MOD09A1 pixel contains the best possible observation in an 8-day period as selected on the basis of high observation coverage, low view angle, the absence of clouds or cloud shadow, and aerosol loading [36]. The best observation during eight day period is saved. MOD09A1 file contains 6 additional bands of information concerning quality control, solar zenith,relative azimuth, view zenith, surface reflectance 500 m state flags, and surface reflectance day of year in addition to the same seven spectral bands of data as the daily file [16].

2.1.3 MODIS Fire Products

Fire is an important ecosystem process affecting vegetation structure and composition [6] and in many land use systems is a proximate cause or indicator of land cover change [4]; [5]; [6]. Hence removal/alteration of vegetation structure is one of characteristics of fire hotspots or burned areas [13]; [37].

It remains unclear if fire regimes will change as human population, their land use practices, and the climate change [7];[8];[9]. Which is why reliable fire information is needed to be provided to scientists, resource managers and policymakers [1].

On regular basis, 1 km active fire product [15]; [12] as well as 500 m resolution burned area product that maps the approximate day and extent of burning is being generated by MODIS. MODIS algorithms map burned areas using the temporal and structural changes of vegetation [1]. The algorithm needs the regularly calibrated MODIS data [38].

The fire affected area may not be detected by the most fire regimes satellite active fire detections [1] because the satellite may not overpass frequently enough to aquire the spatial details of how fires spread across landscapes, and because clouds and optically thick smoke may hinder active fire detection [10]; [11].

In a study by Roy et al. it is stated that the unmapped areas reported by both MODIS fire products were primarily due to cloud with an average annual global unmapped area of 43% and 30% for the MODIS burned area and active fire products respectively, and up to 68% and 50% annual average unmapped respectively in Northern Eurasia [39].

Hence, Usage of multitemporal satellite data, which provides several advantages over single date data for mapping fire-affected areas was the solution for decades [40]; [13].

Significant advances in sattelite mapping of fire-affected areas is remarked by the latest generation of moderate resolution remote sensing systems which includes atmospherically corrected, geolocated, calibrated and cloud-screened data [1].

2.1.4 MODIS Limitations

Giving the fact that MODIS is an optical sensor and not a passive-microwave sensor which means that it can not penetrate clouds/smoke, hence cloud cover tends to be one of the biggest obstacles in detecting the land surface. The MODIS active fire product will only detect fires that are burning at the time of satellite overpass [40]. Cloud masks have been built into the data arrays of the MODIS products to avoid clouds from being miss-classified as ice or snow. Along the day time thermal bands are used - to detect the sea ice surface temperature by emittance - as well as visible bands of MODIS which are used when the reflectance is the main mechanism for observing snow and ice. While during the night time only thermal bands can be used [25].

2.1.5 Terra Satellite

Terra, named for Earth, Satellite was launched on December 18, 1999 by NASA's Earth Observing Systems (EOS) for monitoring how is the Earth changing, collecting data about the Earth's changing climate and what are the consequences of these changes [41]. It passes from north to south in the morning. Along with the Aqua Satellite they cover the entire Earth's surface every 1 to 2 days gaining data in 36 spectral bands [42].

The five Terra on board sensors that are designed to monitor the state of Earth's environment and the continuing changes in its climate system are: Moderate-resolution Imaging Spectroradiometer (MODIS), Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) for creating definite maps of land surface reflectance, temperature and elevation [43], Clouds and the Earth's Radiant Energy System (CERES) to measure the Earth's thermal radiation and used with MODIS to get accurate information about clouds [44], Multi-angle Imaging SpectroRadiometer (MISR) determining the amount of sunlight that is scattered from the Earth's surface so as to determine the amount, height and types of clouds [45], Measurements of Pollution in the Troposphere (MOPITT) by measuring the emitted and reflected radiances of carbon monoxide that are generated by cars and factories [46].

2.1.6 Aqua Satellite

Aqua, named for Water, Satellite was launched on May 04, 2002 by NASA's Earth Observing Systems (EOS) for collection large masses of information about water changes in the Earth surface. The collection of the water includes water in its different life cycle stages, solid, liquid, soil moisture, sea ice, land ice and vapor state [47]; [48]. It passes from south to north in the afternoon [49].

There are six operating sensors or instruments on the Aqua satellite: MODIS, Atmospheric Infrared Sounder (AIRS) to allow measuring of atmospheric temperature and humidity, land and ocean surface conditions [50], Advanced Microwave Scanning Radiometer-EOS (AMSR-E) observing atmospheric, land, oceanic, and cryospheric parameters, including precipitation, sea surface temperatures, ice concentrations, snow water equivalent, surface wetness, wind speed, atmospheric cloud water, and water vapor [51], CERES, Advanced Microwave Sounding Unit (AMSU-A) provides atmospheric temperature measurements [52], Humidity Sounder for Brazil

(HSB) for collection humidity profiles throughout the atmosphere [53]. The AMSR-E rotation rate is reduced for calibration purposed rather than for science [48].

2.2 Spatial Data Retrieval

Spatial Data are distributed on the web with specifications like Web Map Service, Web Feature Service and Web Coverage Service [21]. So the further step is to process them.

A research paper published by C.Strobl and J.Eberle [21] was discussing what requirements for web-based processing of remote sensing data do exist, whether they can be achieved with actual standards, how the combination of OGC standards and processing of remote sensing data can be realised and what requirements need to be solved for the web-based client. And in order to achieve these aims and answer these questions near real-time and archived remote sensing data from the NASA MODIS sensor was processed. Information like fire hotspots (MOD14/MYD14) [14] and land surface temperature (MOD11) [54] was derived and published within a web-based information system.

3 METHODOLOGY AND TECHNICAL IMPLEMENTATION

This chapter gives a detailed explanation on how the application modules were built, shows the system architecture and how these modules interact with each other. The first part describes the design of the system to show the big picture. In the second part we dive deeper into the application modules and the role of each one of them. The third part is showing the technical implementation briefly and how models and tasks were implemented and handeled technical wise. As for the fourth part we show the data sources that we relied on. In the fifth part we demostrate how to configure the application to get it running. The sixth and the last parts are showing the used framework and language and why did we decide on chosing these ones.

3.1 Design

The concept of this web mapping application heavily relies on information exchange via Representational State Transfer (RESTful) web services. Separate modules are built that communicate through REST interface. JavaScript Object Notation (JSON) requests are sent from one module to another where a certain operation will be processed and the result will be sent back as a JSON response where it will be parsed back.

A system architecture and the services for each module were defined to build up a loosely coupled web-based modules following the REST framework as shown in Figure 2. For the interaction between these modules rules for data exchange and data management were specified. The system architecture proposed in this thesis consists of three layers:

- **Data Layer**
  It describes different data sources available for the near real-time and archived remote sensing data from the NASA MODIS. Each of the MODIS Level has different data source and they have to be accessed differently [21].
- **Service Layer**
  This brings us to the Service Layer which communicates with the different data sources and parse the user input from the client side to the matched MODIS data at the data source. There are two different modules in this service layer that are implemented: The Archive Connector and The Fetcher. Where the Archive Connector gets the user input via Hyper Text Transfer Protocol (HTTP) as a JSON request, parse it and pulls the available

information of the selected area based on what exist in the data source as linked files via File Transfer Protocol (FTP).

While the Fetcher pulls the selected data from the MODIS data pool and saves it in the pickup folder where the process chain checks periodically for processing the MODIS data. The output data in this module is in the form of the Hierarchical Data Format (HDF). This is used from NASA as output data format of their Earth Observation System.

- **Client/Front End Layer**
  Is where the user selects the desired area of interest, the MODIS Level and the Satellite he would like to get the data from.

These different modules or services are exposed over the REST interface so they can be used by any other application.

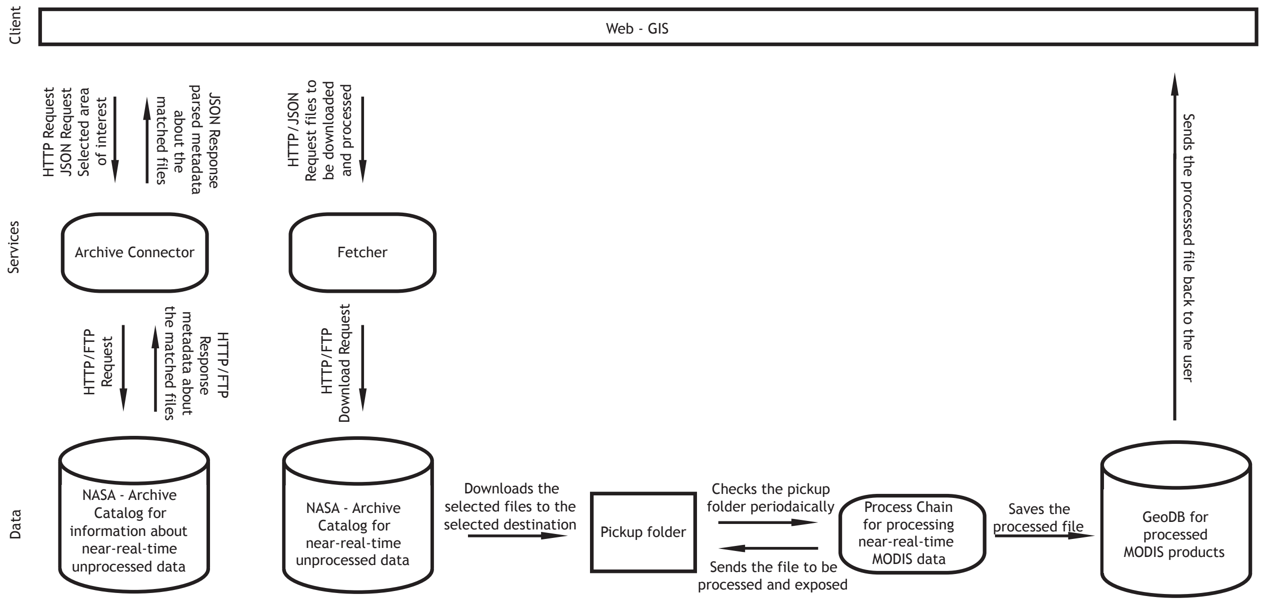
Client
Web - GIS
HTTP Request JSON Request Selected area of interest
JSON Response parsed metadata about the matched files
HTTP/JSON Request files to be downloaded and processed
Sends the processed file back to the user
Services
Archive Connector
Fetcher
HTTP/FTP Request
HTTP/FTP Response metadata about the matched files
HTTP/FTP Download Request
Data
NASA - Archive Catalog for information about near-real-time unprocessed data
NASA - Archive Catalog for near-real-time unprocessed data
Downloads the selected files to the selected destination
Pickup folder
Checks the pickup folder periodaically
Sends the file to be processed and exposed
Process Chain for processing near-real-time MODIS data
Saves the processed file
GeoDB for processed MODIS products

Figure 3.1: The system workflow

The main goal of this thesis was to pull the unprocessed MODIS files, Georectified Calibrated Geolocated (MOD02 Level 1B) Dataset and Geolocated Data Set (MOD03) derived from near real time MODIS data from different data sources and get them to be exposed to our process chain in which we can make use of its output for different applications in order to extract further information, such as fire hotspot points (MOD 14).

To download the unprocessed MODIS data from the NASA servers, the module "fetcher" was developed. If a HTTP/FTP URL is given, the data from the URL is downloaded as HDF file to the specified pickup folder. As output, an unprocessed MODIS file downloaded from the NASA servers is available for further processing.

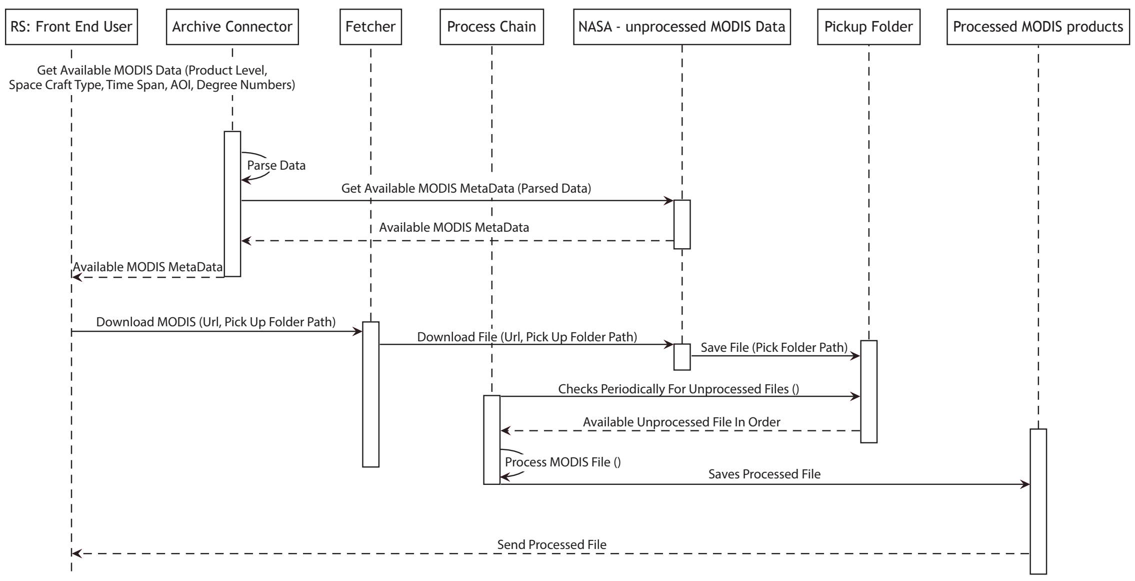
RS: Front End User
Archive Connector
Fetcher
Process Chain
NASA - unprocessed MODIS Data
Pickup Folder
Processed MODIS products
Get Available MODIS Data (Product Level, Space Craft Type, Time Span, AOI, Degree Numbers)
Parse Data
Get Available MODIS MetaData (Parsed Data)
Available MODIS MetaData
Available MODIS MetaData
Download MODIS (Url, Pick Up Folder Path)
Download File (Url, Pick Up Folder Path)
Save File (Pick Folder Path)
Checks Periodically For Unprocessed Files ()
Available Unprocessed File In Order
Process MODIS File ()
Saves Processed File
Send Processed File

Figure 3.2: Sequence diagram for an overview of how the whole system works

As for the process chain which is already implemented by Strobl: In order to acheive the remote sensing producs or the final products several steps have to be carried on. Figure 4 shows an example workflow to derive fire hotspot points [21]. Each of the workflows can work with in-house data from the data archive or extraneous data as HTTP URL. The eventual output of the workflow shown in Figure 4 are a Geospatial Tagged Image File Format (GeoTIFF) file visualising the satellite scene, a Geography Markup Language (GML) and a Shapefile representing the fire points as well as a GeoTIFF file displaying the classification mask achieved for distinguishing fire points [21].

Similar workflows were generated for the derivation of land surface temperature, for the visualisation of the given MODIS satellite scene and for the fire-service, automatically running on near real-time data. For this service, the derived fire hotspot points are deployed into a PostgreSQL/PostGIS database and the output data is published to an external web server for website integration.

This process chain was developed at the German Aerospace Center to derive products based on MODIS satellite data, integrate own processing workflows, monitor started processes and visualise the output data. Based on the literature work the potential to provide web-based processing services with workflows is very visible [21].

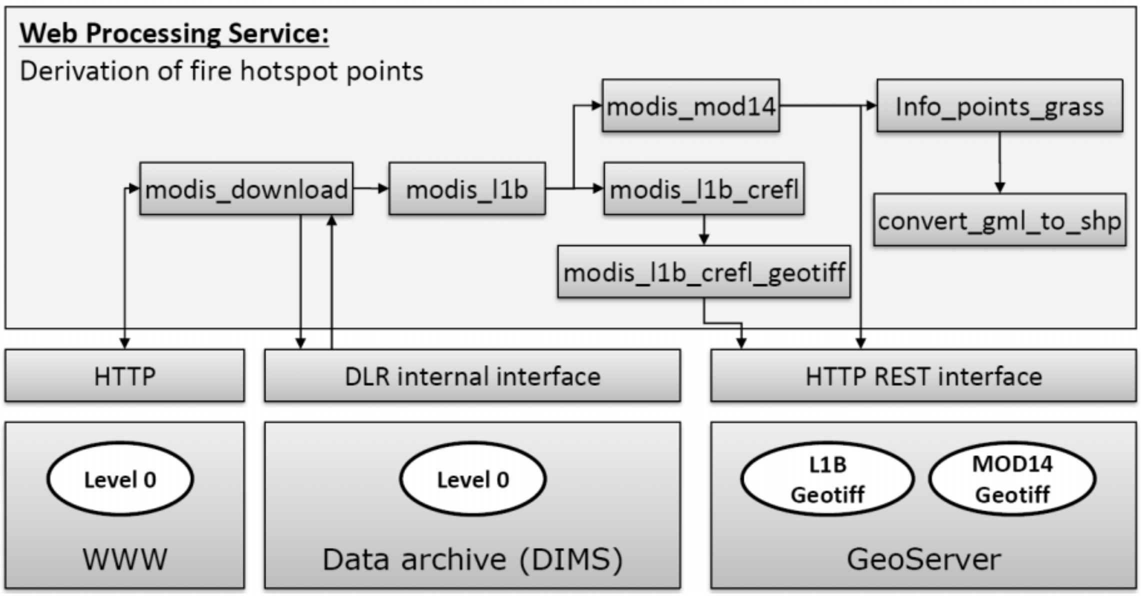

Figure 3.3: Sequence of processes called for derivation of fire points with interfaces for input and output data(source:[21])

The idea from Michael and Ames [55] is to implement a mechanism for listing all available data from the processing server which leads to an efficient data management without the need of download them immediately.

3.2 Application modules

Four different modules were built as follow:

- **Archive Connector:**
  Mapping the user input selections (Product Level, Spacecraft Type, Timespan, Swath Vs Grid) to NASA-MODIS Data Archive retrieving the metadata for the matched files.
- **MODIS Available Dates:**
  Retrieving MODIS Surface Reflectance (MOD09, MYD09) available dates.

- **Fetcher:**

  Is where a selected MODIS data set is fetched from the NASA MODIS Data Archive to the Web Back-end and retrieved.

- **Admin module:**

  Controlling and checking the status of the processes in the queues.

The system has to be loosely coupled where each module can be treated as a standalone application sending and receiving requests over HTTP.

3.3 Implementation

Each of the aforementioned modules is implemented in Python and built on the Django framework, which is a free open source web application framework written in Python which follows the model-view-controller (MVC) design pattern. Django ensures the reusability and pluggability of the components of the code [56].

Eight models were built where complicated nested relationships were avoided. One-to-many relationship is used below between the Surface Reflectance Connector and the retrieved data.

```
class MODISConnector(models.Model):
        product_level = models.CharField(max_length=7,
                                 choices=product_levels_choices,
                                 default='MOD09A1')
        spacecraft_type = models.CharField(max_length=5,
                                 choices=spacecraft_type_choices,
                                 default='Terra')
        start_timespan = models.DateField()
        end_timespan = models.DateField()
        aoi = models.TextField()
        degree_numbers = models.TextField()

        class Meta:
                    ordering = ('product_level',)
        def __str__(self):
              return self.product_level



class SurfaceReflectanceFileProperties(models.Model):
        hdf_url = models.URLField()
        metadata_url = models.URLField()
        image_url = models.URLField()
        collection_metadata = models.CharField(blank=True,
                                               max_length=1000)
        range_date_time = models.CharField(blank=True,
                                           max_length=1000)
        boundary = models.CharField(blank=True,
                                    max_length=1000)
        surface_reflectance_connector =
                        models.ForeignKey(MODISConnector)

        class Meta:
                        ordering = ('hdf_url',)
        def __str__(self):
                        return self.hdf_url


class MODISLevel1FileProperties(models.Model):
        hdf_url = models.URLField()
        file_name = models.CharField(blank=True,
                                           max_length=1000)
        file_id = models.CharField(blank=True,
                             max_length=1000)
        file_size = models.CharField(blank=True,
                                max_length=1000)
        updated = models.CharField(blank=True,
                                 max_length=1000)
        geobox = models.CharField(blank=True,
                                max_length=1000)
        modis_level1_connector =
                        models.ForeignKey(MODISConnector)

        class Meta:
                        ordering = ('file_name',)

        def __str__(self):
                        return self.file_name
```

In order to build the Web browsable APIs Django REST framework had to be used. The models are serialized and deserialized into JSON representations. Serialization classes are built for that purpose.

The Django REST framework API views are written using class based views, rather than function based views. It allows us to reuse common functionality and follow the 'don't repeat yourself' approach which is a principle of software development.

REST framework provides ModelSerializers, that was used to avoid information replication that is also contained in the Model itself.

```

class SurfaceReflectanceFilePropertiesSerializer(serializers.ModelSerializer):

    class Meta:
        model = SurfaceReflectanceFileProperties
        fields = ('id', 'hdf_url', 'metada_url', 'image_url',
                'collection_metadata', 'range_date_time', 'boundary')

class MODISLevel1FilePropertiesSerializer(serializers.ModelSerializer):

    class Meta:
        model = MODISLevel1FileProperties
        fields = ('id', 'file_name', 'hdf_url', 'file_id',
                'file_size', 'updated', 'geobox')

class MODISConnectorSerializer(serializers.ModelSerializer):

    class Meta:
        model = MODISConnector

        fields = ('id', 'product_level', 'spacecraft_type',
                'star_timespan', 'end_timespan', 'aoi', 'degree_numbers')
```

The Archive Connector gets the user input selection (product_level, satellite_type, start_timespam, end_timespam, Swath Vs Grid, degree_numbers) as a JSON request.

Then, it deserializes the user input JSON request to complex datatypes, request the NASA - MODIS Data Archive and check for available MODIS data for the selected region, time period and product level using FTP and then serialize the result back to native Python code which is easily rendered to JSON with the available URLs and metadata matched.

```
class MODISProductList(APIView):
    """
    List all MODIS Data, or create a new one.
    """
    def get(self, request, format=None):
        surface_reflectance_connector =
                SurfaceReflectanceConnector.objects.all()
        modis_level1_connector =
                MODISLevel1Connector.objects.all()
        result = list()

        for item in surface_reflectance_connector:
            dictionary = model_to_dict(item)
            modis09_metadata = SurfaceReflectanceFileProperties.objects
```

```
                .filter(surface_reflectance_connector=item.id)
            dictionary['modis09_Data'] = list()
            for file_property in modis09_metadata:
                data = {'url': file_property.url,
                        'tile_number':
                                file_property.tile_number,
                        'collection_metadata':
                                file_property.collection_metadata,
                        'range_date_time':
                                file_property.range_date_time,
                        'boundary':
                                file_property.boundary,}
                dictionary['modis09_Data'].append(data)
            result.append(dictionary)

        for item in modis_level1_connector:
            dictionary = model_to_dict(item)
            all_file_properties = MODISLevel1FileProperties.objects
                .filter(modis_level1_connector=item.id)
            dictionary['file_properties'] = list()
            for file_property in all_file_properties:
                data = {'file_name': file_property.file_name,
                        'file_id': file_property.file_id,
                        'file_size': file_property.file_size,
                        'updated': file_property.updated,
                        'geobox': file_property.geobox, }
                dictionary['file_properties'].append(data)
            result.append(dictionary)

        return Response(result)

    def post(self, request, format=None):
        data = request.DATA[0]
        product_level = data['product_level']
        dictionary = list()

        for case in switch(product_level):
            if case('MOD09A1'):
                dictionary =
                        tasks.modis_surface_reflectance_list(data)
                break

            if case('MOD03'):
                dictionary =
                        tasks.modis_level1_list(data)
                break

            if case():
                print "something else!"

        return Response(dictionary, status=status.HTTP_201_CREATED)
```

The user at the web back-end then checks the JSON response and checks the available URLs' metadata and picks the desired ones that he/she would like to retrieve.

Then the user sends back a JSON request with the selected URLs to the Fetcher which should connect to NASA - MODIS Data Archive and download the desired hdf files to the designated pickup point.

Distributed Task Queue, Celery, is being used during the hierarchical data format files download process, allowing the download process for multiple files to be executed asynchronously. The user gets a response back once the download order is submitted where Celery schedules the download order in the background.

```
{
    "url":"http://e4ftl01.cr.usgs.gov/MOLT/MOD09A1.005/2013.09.14/
            MOD09A1.A2013257.h19v06.005.2013266130027.hdf",
    "pickup_folder_path": "D:/MODISData/",
    "status": "your order has been sent to the task queue and will be
        downloaded soon...",
    "id": 1,
    "product_level": "MOD09A1"
}
```

As soon as the file is downloaded successfully the user will get notified and the download status will change accordingly. A mass amount of tasks are being processed everyday using Celery [57].

```
{
    "url": "http://e4ftl01.cr.usgs.gov/MOLT/MOD09A1.005/2013.09.14/
            MOD09A1.A2013257.h19v06.005.2013266130027.hdf",
    "pickup_folder_path": "D:/MODISData/",
    "status": "your data is ready to pick up...",
    "id": 1,
    "product_level": "MOD09A1"
}
```

Celery requires a solution to send and receive messages. Usually this comes in the form of a separate service called a message broker. In our case we used Redis [57].

Message broker handles the information about new tasks for Celery [58]. Redis was selected in our case because of its lightweight, speed and its high scalability for data storage shared by multiple processes. To be mentioned, Redis does not guarantee the delivery in case of system termination.

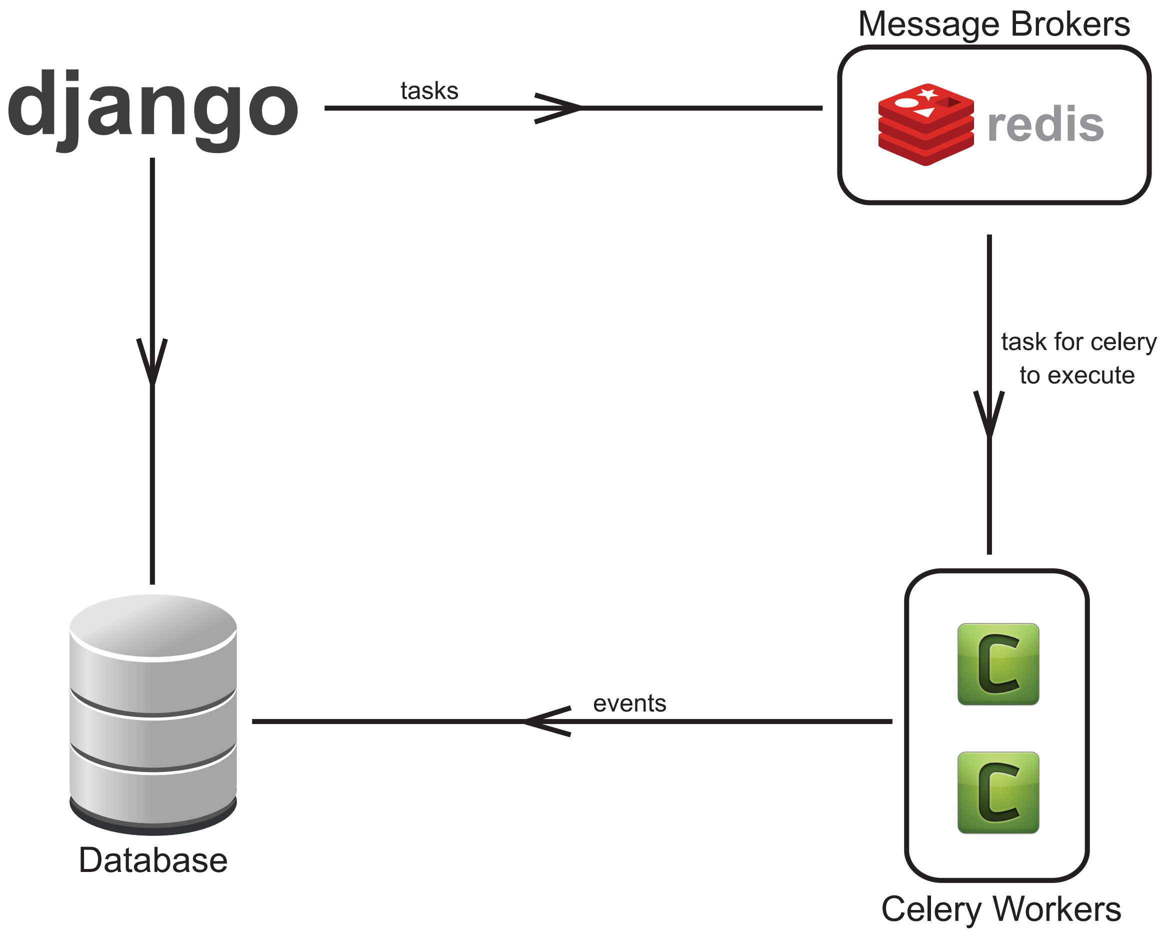

Figure 3.4: Celery Architecture (source:[2])

Process chain in its place checks on the pickup folder regularly and gets the hdf files from there where they get exposed to the process chain and the fire hotspots are extracted (MOD14).

These modules are loosely coupled plug and play ones, which means they do not depend on one another they just communicate through JSON requests. The developed modules can be used in different applications and are platform independent.

The idea behind serializing is that each of the REST modules can accept a JSON request, serializing JSON request to the model's understandable language and save the data in the database. Deserializing is the other way around, where the data from the storage medium is retrieved and deserialized to JSON response to the user.

Archive Connector

It is used for mapping the user input selections (Product Level, Spacecraft Type, Start_Timespan, End_Timespan, Swath Vs Grid) to the parser module.

Input parameters:

- "Product Level", whether it is MOD09 or MYD09
- "Spacecraft Type", whether it is Terra or Aqua
- "Start_Timespan"
- "End_Timespan"
- "Degree_Numbers", Lat/Lng of the selected points in which the matched geolocation MODIS data should be retrieved.

```
{
    "product_level": "MOD09A1",
    "spacecraft_type": "Terra",
    "start_timespan": "2013-09-14",
    "end_timespan": "2013-04-28",
    "aoi": "Swath",
    "degree_numbers": "POINT(22.5,28.34) POINT(28.64,28.34) POINT(29.4,19.16)
        POINT(19.19,19.3067)"
}
```

Output:

- Matched .hdf files' references
- Metadata url for each of the matched .hdf
- Sample image url for each of the files
- Collection Metadata
- Range Date Time

- Boundary

```
{
    "id": 1,
    "product_level": "MOD09A1"
    "spacecraft_type": "Terra",
    "start_timespan": "2013-09-14",
    "end_timespan": "2013-04-28",
    "aoi": "Swath",
    "degree_numbers": "[u'22.5,28.34', u'28.64,28.34', u'29.4,19.16',
        u'19.19,19.3067']",

    "modis09_Data":
        [
            {
             "spacecraft_type": "Terra",
                     "degree_numbers": "POINT(22.5,28.34) POINT(28.64,28.34)
                         POINT(29.4,19.16) POINT(19.19,19.3067)",
             "file_properties": [
                {
                    "hdf_url": "
                        http://e4ftl01.cr.usgs.gov/MOLT/MOD09A1.005/2013.09.14/
                        MOD09A1.A2013257.h19v06.005.2013266130027.hdf",
                    "range_date_time": "
                        <RangeEndingTime>23:59:59.000000</RangeEndingTime>
                        <RangeEndingDate>2013-09-21</RangeEndingDate>
                        <RangeBeginningTime>00:00:00.000000</RangeBeginningTime>
                        <RangeBeginningDate>2013-09-14</RangeBeginningDate>",
                    "collection_metadata": "<ShortName>MOD09A1</ShortName>
                        <VersionID>5</VersionID>",
                    "metadata_url": "
                        http://e4ftl01.cr.usgs.gov/MOLT/MOD09A1.005/2013.09.14/
                        MOD09A1.A2013257.h19v06.005.2013266130027.hdf.xml",
                    "image_url": "
            http://e4ftl01.cr.usgs.gov/MOLT/MOD09A1.005/2013.09.14/
            BROWSE.MOD09A1.A2013257.h19v06.005.2013266130028.1.jpg",
        "boundary":
            "<Point>
                    <PointLongitude>10.5996129612972</PointLongitude>
                    <PointLatitude>19.9237405201567</PointLatitude>
            </Point>
            <Point>
                    <PointLongitude>11.498652399037</PointLongitude>
                    <PointLatitude>30.0080231274941</PointLatitude>
            </Point>
            <Point>
                    <PointLongitude>23.1051739635034</PointLongitude>
                    <PointLatitude>29.9995009438162</PointLatitude>
            </Point>
            <Point>
                    <PointLongitude>21.2899725552125</PointLongitude>
                    <PointLatitude>19.9159339927733</PointLatitude>
            </Point>"
        },
       {
        "hdf_url": "
            http://e4ftl01.cr.usgs.gov/MOLT/MOD09A1.005/2013.09.14/
            MOD09A1.A2013257.h19v07.005.2013266123041.hdf",
        "range_date_time": "
            <RangeEndingTime>23:59:59.000000</RangeEndingTime>
            <RangeEndingDate>2013-09-21</RangeEndingDate>
            <RangeBeginningTime>00:00:00.000000</RangeBeginningTime>
            <RangeBeginningDate>2013-09-14</RangeBeginningDate>",
        "collection_metadata":"<ShortName>MOD09A1</ShortName>
            <VersionID>5</VersionID>",
        "metadata_url": "
            http://e4ftl01.cr.usgs.gov/MOLT/MOD09A1.005/2013.09.14/
```

```
                   MOD09A1.A2013257.h19v07.005.2013266123041.hdf.xml",
               "image_url": "
                   http://e4ftl01.cr.usgs.gov/MOLT/MOD09A1.005/2013.09.14/
                   BROWSE.MOD09A1.A2013257.h19v07.005.2013266123041.1.jpg",
               "boundary":
                   "<Point>
                           <PointLongitude>10.1164887877562</PointLongitude>
                           <PointLatitude>9.96346738992328</PointLatitude>
                   </Point>
                   <Point>
                           <PointLongitude>10.5987660250161</PointLongitude>
                           <PointLatitude>20.004527943914</PointLatitude>
                   </Point>
                   <Point>
                           <PointLongitude>21.2932451415814</PointLongitude>
                           <PointLatitude>19.9997043834619</PointLatitude>
                   </Point>
                   <Point>
                           <PointLongitude>20.3192706375588</PointLongitude>
                           <PointLatitude>9.95883147552263</PointLatitude>
                   </Point>"
                }
               "collection_metadata": "
                    <ShortName>MOD09A1</ShortName>
                    <VersionID>5</VersionID>",
               "range_date_time": "
                    <RangeEndingTime>23:59:59.000000</RangeEndingTime>
                    <RangeEndingDate>2013-09-21</RangeEndingDate>
                    <RangeBeginningTime>00:00:00.000000</RangeBeginningTime>
                    <RangeBeginningDate>2013-09-14</RangeBeginningDate>",
         }
    ]
}
```

MODISAvailableDates

Get available dates for specific MODIS product level, Lat/Lng.

Input parameters:

- ”Product Level”, whether it is MOD09 or MYD09
- Latitude
- Longitude

```
{
    "product_level": "MOD09A1",
    "latitude": "26",
    "longitude": "30"
}
```

Out Parameters:

- Available Dates

```
{
    "latitude": 26.0,
    "available_dates":
        [
            {
                "gregorian_date": "2000-02-18",
                "julian_date": "A2000049"
            },
            {
                "gregorian_date": "2000-02-26",
                "julian_date": "A2000057"
            },
            {
                "gregorian_date": "2000-03-05",
                "julian_date": "A2000065"
            },
        ]
}
```

Fetcher

Is where a selected MODIS data set is fetched from the NASA MODIS Data Archive to the Web Back-end and retrieved.

Input parameters:

- ”URL”, the hdf file to be downloaded,
- ”PickUpFolderPath”, the destination folder,
- ”Product Level”

```
{
    "url": "http://e4ftl01.cr.usgs.gov/MOLT/MOD09A1.005/2013.09.14/
            MOD09A1.A2013257.h19v06.005.2013266130027.hdf",
    "pickup_folder_path": "D:/testMODIS/",
    "product_level": "MOD09A1"
}
```

Out parameters, before being downloaded:

- ”Status”, the hdf file download status

```
{
    "url": "http://e4ftl01.cr.usgs.gov/MOLT/MOD09A1.005/2013.09.14/
            MOD09A1.A2013257.h19v06.005.2013266130027.hdf",
    "pickup_folder_path": "D:/MODISData/",
    "status": "your order has been sent to the task queue and will be
        downloaded soon...",
    "id": 1,
    "product_level": "MOD09A1"
}
```

After the file being downloaded the status changes as following:

```
{
    "url": "http://e4ftl01.cr.usgs.gov/MOLT/MOD09A1.005/2013.09.14/
            MOD09A1.A2013257.h19v06.005.2013266130027.hdf",
    "pickup_folder_path": "D:/MODISData/",
    "status": "your data is ready to pick up...",
    "id": 1,
    "product_level": "MOD09A1"
}
```

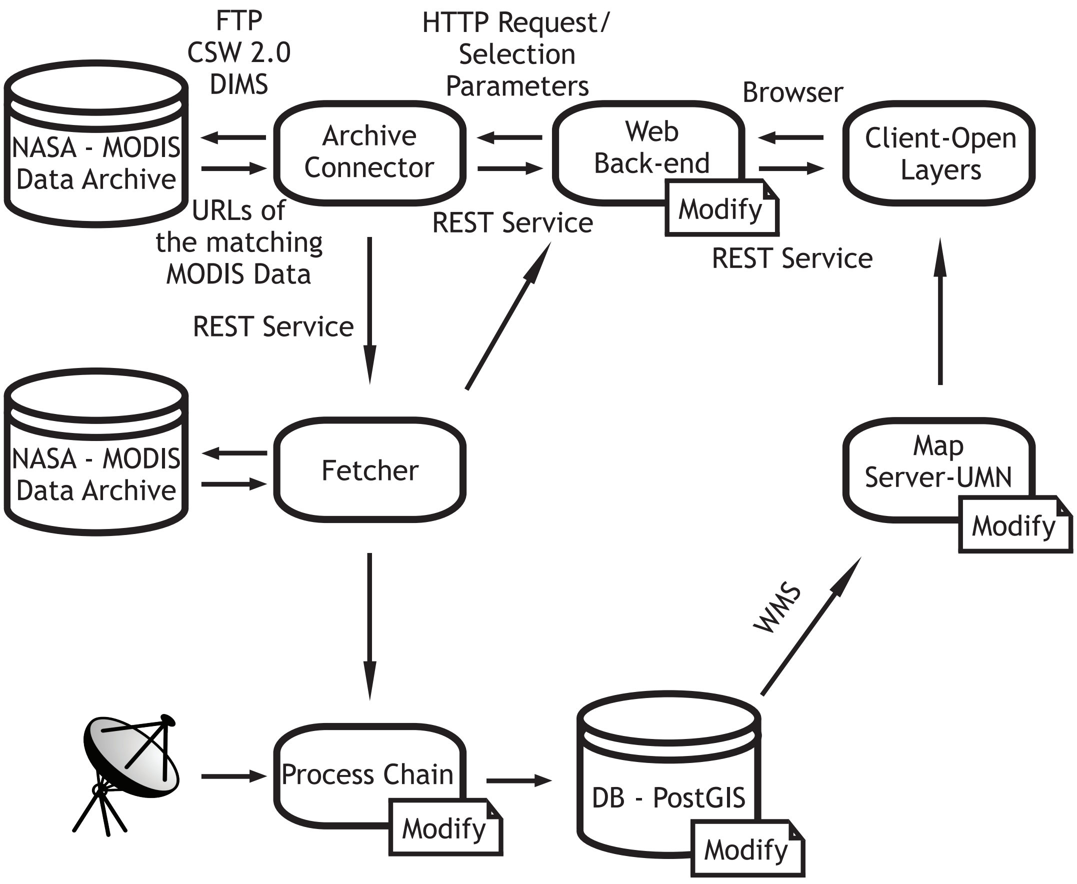

Figure 3.5: System Architecture

3.4 Data sources

Features of the atmosphere, land and oceans can be described from the MODIS observation data products. There are few MODIS products' data pools. Level 1 and Atmosphere Archive and Distribution System (LAADS) provides MODIS Level 1 and atmosphere products. Land Processes Distributed Active Archive Center (LP DAAC) at the U. S. Geological Survey Earth Resource Observation and Science (EROS) Data Center (EDC) provides Level 2 and Level 3 Land products [59]. Whereas National Snow and Ice Data Center (NSIDC) distributes Cryosphere data products [60]. Ocean Color Data Processing System (OCDPS) at Goddard Space Flight Center (GSFC) provides ocean color and sea surface temperature products [61]. Benefiting from MODIS Direct Broadcast signal, receivers with a suitable x-band receiving system can aquire and receive territorial data directly from Terra or Aqua spacecraft [26].

Many data products derived from MODIS observations describe features of the land, oceans and the atmosphere that can be used for studies of processes and trends on local to global scales.

The MODIS surface reflectance products (MOD09 and MYD09) are downloaded from the United States Government office (USGS) Land Process DAAC data pool. The data pool provides direct ftp access to the most recent MODIS products available at: `ftp://e4ftl01u.ecs.nasa.gov/MOLA/MYD09A1.005`.

For the MOD02 and MOD03, LAADS web APIs were used. Where the mission of LAADS is to provide quick and easy access to MODIS Level 1, Atmosphere and Land data product through this direct ftp access: `ftp://ladsftp.nascom.nasa.gov/` [62].

LAADS web service REST interface was used for the MODIS Level 1 products metadata available at: `http://ladsweb.nascom.nasa.gov/data/web_services.html`

The Process Chain is used right after the download process is completed to clip, reproject the images and to convert the images to GML and GeoTiff format.

3.5 Configuring the application

Giving that we are using Django for our application, creating a project will need some Django-specific options and application-specific settings.

Since SQLite is included in Python and it is the default, the configuration uses SQLite, we do not need to change the configuration file MODIS/settings.py:

```
DATABASES = {
    'default': {
        # Add 'postgresql_psycopg2', 'mysql',
        # 'sqlite3' or 'oracle'.
        'ENGINE': 'django.db.backends.sqlite3',
        # Or path to database file if using sqlite3.
        'NAME': 'tmp.db',
        # The following settings are not used with sqlite3:
        'USER': '',
        'PASSWORD': '',
        # Empty for localhost through domain sockets
        #or '127.0.0.1' for localhost through TCP.
        'HOST': '',
        # Set to empty string for default.
        'PORT': '',
    }
}
```

After configuring our application, we have to create the Models in which we will use in our application.

Models.py

```
class SurfaceReflectanceConnector(models.Model):
    product_level = models.CharField(max_length=7,
                                     choices=product_levels_choices,
                                     default='MOD09A1')
    spacecraft_type = models.CharField(max_length=5,
                                       choices=spacecraft_type_choices,
                                       default='Terra')
    timespan = models.DateField()
    aoi = models.TextField()
    degree_numbers = models.TextField()
```

```
class SurfaceReflectanceFileProperties(models.Model):
    url = models.URLField()
    tile_number = models.CharField(max_length=6)
    collection_metadata = models.CharField(blank=True, max_length=1000)
    range_date_time = models.CharField(blank=True, max_length=1000)
    boundary = models.CharField(blank=True, max_length=1000)
    surface_reflectance_connector = models.ForeignKey(
   SurfaceReflectanceConnector)

    class Meta:
        ordering = ('url',)
    def __str__(self):
        return self.url
```

The code above is straightforward. Each of these models represents a class in the database, and each of its class variables represents database field in the model.

Edit MODIS/settings.py and change the INSTALLED_APPS to include 'spacecrafts', 'rest_framework', 'djcelery' and ‘kombu.transport.django’

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'spacecrafts',
    'djcelery',
    'kombu.transport.django',
)
```

Now by running the syncdb command the tables will be created in the database.

```
$ python manage.py syncdb
```

The syncdb command looks at the INSTALLED_APPS setting and creates any necessary database tables according to the database settings in your MODIS/settings.py file 15 reference.

After running this command line Django REST framework is installed as well, so we introduced our serializers.py class that is used for allowing complex data such as model instances to be converted to native Python datatypes in which it can be rendered to JSON. Deserialization is also available allowing parsed data to be converted back into complex types after first validating the incoming request 16 references.

```
from rest_framework import serializers
from spacecrafts.models import *

class SurfaceReflectanceFilePropertiesSerializer(serializers.ModelSerializer):

    class Meta:
        model = SurfaceReflectanceFileProperties
        fields = ('id', 'tile_number', 'url', 'collection_metadata',
                  'range_date_time', 'boundary')


class SurfaceReflectanceConnectorSerializer(serializers.ModelSerializer):

    class Meta:
        model = SurfaceReflectanceConnector
        fields = ('id', 'product_level', 'spacecraft_type', 'timespan',
                  'aoi', 'degree_numbers')
```

The next step is to point the root URLconf at the MODIS/urls.py

```
urlpatterns = patterns('',
        url(r'^', include('spacecrafts.urls')),
)
```

spacecrafts/urls.py

```
from django.conf.urls import patterns, url
from rest_framework.urlpatterns import format_suffix_patterns
from spacecrafts import views

urlpatterns = patterns('',
        url(r'^MODIS/$',
                 views.MODISProductList.as_view()),
        url(r'^downloadOrder/$',
                views.DownloadOrder.as_view()),
        url(r'^MODISAvailableDates/$',
                views.MODISAvailableDates.as_view()),
        url(r'^MODISSurfaceReflectance/(?P<pk>[0-9]+)/$',
                views.MODISSurfaceReflectanceDetail.as_view()),
        url(r'^parseXML/$',
                views.ParseXML.as_view()),
)

urlpatterns = format_suffix_patterns(urlpatterns)
```

The operations/methods are separated from the views. Views can be found in spacecrafts/views.py

```
class MODISProductList(APIView):
    """
    List all MODIS Data, or create a new one.
    """
    def get(self, request, format=None):
```

```
    def post(self, request, format=None):


class DownloadOrder(APIView):
    """
    Downloads MODIS Data
    """
    def get(self, request, format=None):


    def post(self, request, format=None):


class MODISAvailableDates(APIView):
    """
    List all available dates for specific MOD09
    """
    def get(self, request, format=None):


    def post(self, request, format=None):


class ParseXML(APIView):
    """
    Parse incoming xml and extract the nodes
    """


    def get(self, request, format=None):


    def post(self, request, format=None):
```

Basically the logic for the operations is inserted in the tasks.py class, the views.py is used for the get/post request, serialize/deserialize and saving data in the model.

The following scenario will show how the workflow goes:

Getting a downloadOrder request for the user will be redirected to the DownloadOrder class view as mentioned in the urls.py

```
url(r'^downloadOrder/$', views.DownloadOrder.as_view()),
```

The incoming JSON request is expected to be as follow:

```
[
    {
        "url": "http://e4ftl01.cr.usgs.gov/MOLT/MOD09A1.005/2013.09.14/
                MOD09A1.A2013257.h19v06.005.2013266130027.hdf",
        "pickup_folder_path": "D:/MODISArchive/",
        "product_level": "MOD09A1"
    }
]
```

First thing to be done is to serialize the JSON parameters to native Python code, this is done using the following code:

```
        data = request.DATA[0]
        product_level = data['product_level']
        task = ''
        modis_data_fetcher = MODISDataFetcher()
        modis_data_fetcher.url = data['url']
        modis_data_fetcher.pickup_folder_path = data['pickup_folder_path']
        modis_data_fetcher.product_level = data['product_level']
        modis_data_fetcher.status = "your order will be downloaded soon..."
        modis_data_fetcher.save()

        for case in switch(product_level):
            if case('MOD09A1'):
                task = tasks.download_surface_reflectance_modis
                        .delay(data, modis_data_fetcher.pk)
                break
            if case('MOD03'):
                task = tasks.download_lance_modis
                        .delay(data, modis_data_fetcher.pk)
                break

        tasks.get_task_status(task.id)

        dictionary = list()
        dictionary = model_to_dict(modis_data_fetcher)

        return Response(dictionary, status=status.HTTP_201_CREATED)
```

First thing we did is that we created an instance of the MODISDataFetcher model and saved the serialized json parameters to this model. Then after depending on whether the data to be downloaded is MODIS09 or MODIS03 different task will be called.

A response will be sent back to the user right after, without waiting for the data to be downloaded, and the task queue, Celery, will be responsible for the downloading process.

3.6 Used framework

Django which is a Python-based web framework was used. It allows rapid and easy development without the need to do lots of configurations. It is a complete framework that support multiple databases like MySql, SQLite and PostgreSQL [56]. Coding in Django tends to be built in phases. First thing we built is the database, which is implemented in the code as an abstract representation and is called models. Serializers are used for the serializing/deserializing of the incoming and outgoing messages from and to the database. Each of the features is coded individually through the views. The front-end displays are designed as a separate module and then finally wrapped up together through the URL maps. This way makes the code and the modules more loosely coupled that change in any of them does not affect all the other parts of code. Expanding the code became easier as the modules are loosely coupled and do not rely solely on each other [63].

3.7 Used language

Python is a high level programming language, used in the scientific community on large scale. It emphasizes code readability and can be easily extended.

It is a multi-paradigm which supports object oriented programming and structured programming. Python is platform independent which assures the code usability over different operating systems.

A huge amount of libraries are available, and are increasing everyday depending on daily needs. It is commonly cited as one of Python's greatest strengths [64]. HTTP and MIME protocols for example are supported for web-based applications and other modules and tools are available for Geospatial programming such as Shapely, Fiona, ArcPy's, and GeoAlchemy.

4 CONCLUSION

A web-based user interface for the retrieval, processing and exposing of products generated by satellite-based crisis information including wildfire hotspot service and flooded areas service based on MODIS near realtime data in conjunction with the adequate geo basis information to the process chain is being built.

Archive Connector mapping/parsing the user input parameters to NASA-MODIS Data Archive retrieving the metadata for the matched files is being built. Then the Fetcher where the selected MODIS data set is fetched from the NASA MODIS Data Archive to the Web Back-end is being established. The Fetcher module uses the Distributed Task Queue, Celery, during the hierarchical data format files download process, allowing the download process for multiple files to be executed asynchronously. A web-user interface is built so as to demonstrate the developed modules and to wrap it up together.

The retrieved MODIS data will get exposed to the process chain where the higher-level MODIS products will be generated. Products include both vector and raster data of water and flooded areas, receding water, cloud masks and fire hotspots. Service chains to retrieve the remote sensing products for the desired regions is implemented where statistical analysis can be applied after.

Wildfire hotspots as well as flooded areas based on MODIS near realtime data shall be possible to be retrieved on daily basis and not every 8 days.

Instead of letting all the downloaded HDF files to be in a queue so as to get processed by the process chain one after another, we can make use of the Symmetric Multiprocessing or shared memory environment. HDF files is to be distributed across many machines in a cluster or cloud, which will result in a faster processing of the files. Each of the HDF files can be divided into chunks where each of them is to be processed seprately leading to reaching higher-level MODIS products in a faster way. Performing statisctical analysis on the retrieved data as well as derivation and visualization of statistical information can be performed on downloaded data.

The tool should be able to fetch/retrieve HDF files from multiple archives depending on the availablity of the archive servers. A more generic parsing or archive connector module is to be built to be able to connect to multiple archives and checking the status of each.

Test application has been implemented by wrapping up all the create REST modules to automate the retrieval of NASA - MODIS Data through a given range of dates built in separate modules that can be reused in different applications. The user of this system can select interactively the desired region that he/she would like to get archived satellite data for, according to the date period and product level, and process it with the developed processing services and workflows. Visualization of the retrieved MODIS data should allow us to apply statistical analysis where it can be used for studies of processes and trends on local to global scales.

REFERENCES

[1] Roy, D. P., Jin, Y., Lewis, P. E., Justice, C. and O. (2005). 'Prototyping a global algorithm for systematic fire-affected area mapping using MODIS time series data', Remote Sensing of Environment, 97, pp. 137–162.

[2] Xin Q., Olofsson P., Zhu Z., Tan B. and Woodcock C. E. (2013). 'Toward near real-time monitoring of forest disturbance by fusion of MODIS and Landsat data', International Journal of Remote Sensing, Volume 135, pp. 234–247.

[3] Johnson, E. A., and Miyanishi, K. (1997). Forest fires: Behavior and ecological effects. San Diego, CA' Academic Press.

[4] Bucini, G., and Lambin, E. F. (2002). 'Fire impacts on vegetation in Central Africa: A remote-sensing-based statistical analysis', Applied Geography, 22, pp. 27–48.

[5] Cochrane, M. A. (2003). Fire science for rainforests. Nature, 27, 913–919.

[6] Janetos, A. C., and Justice, C. O. (2000). 'Land cover and global productivity: A measurement strategy for the NASA program', International Journal of Remote Sensing, 21, pp. 1491–1512.

[7] Stocks, B. J. (1998). 'Climate change and forest fire potential in Russian and Canadian boreal forest', Climate Change, 38, pp. 1 – 13.

[8] Murphy, P. J., Stocks, B. J., and Kasischke, E. S. (1999). Historical fire records in North American boreal forest. In E. S. Kasischke, and B. J. Stocks (Eds.), Fire, climate change and carbon cycling in boreal forest. Ecological studies series, pp. 274– 288. New York' Springer.

[9] UNEP. (2002). 'Global Environment Outlook 3', Earthscan Publications, London, 446, pp. 351–359.

[10] Justice, C. O., Giglio, L., Korontzi, S., Owens, J., Morisette, J., Roy, D. P., et al. (2002b). The 'MODIS fire products', Remote Sensing of Environment, 83, pp. 244–262.

[11] Robinson, J. M. (1991). 'Fire from space: Global fire evaluation using infrared remote sensing', International Journal of Remote Sensing, 12(1), pp. 3– 24.

[12] Eva, H., and Lambin, E. F. (1998b). 'Remote sensing of biomass burning in tropical regions: Sampling issues and multisensor approach', Remote Sensing of Environment, 64, pp. 292–315.

[13] Pereira, J. M. C., Chuvieco, E., Beaudoin, A., and Desbois, N. (1997). Remote sensing of burned areas: A review. In E. Chuvieco (Ed.), A review of remote sensing methods for the study of large wildland fires. Report of the Megafires Project ENV-CT96-0256, August 1997 (pp. 127–183). Alcala de Henares, Spain' Universidad de Alcala.

[14] Giglio, L., I. Csiszar, and C. O. Justice. (2006). Global distribution and seasonality of active fires as observed with the Terra and Aqua Moderate Resolution Imaging Spectroradiometer (MODIS) sensors, J. Geophys. Res., 111, G02016, doi:10.1029/2005JG000142.

[15] Kaufman, Y. J., C. O. Justice, L. P. Flynn, J. D. Kendall, E. M. Prins, L. Giglio, D. E. Ward, W. P. Menzel, and A. W. Setzer. (1998). Potential global fire monitoring from EOS-MODIS, J. Geophys. Res., 103(D24), 32,315–32,338.

[16] Yale University. (2010). 'The Center for Earth Observation', http://www.yale.edu/ceo (accessed 17 July 2013).

[17] Kaufman, Y., C. Ichoku, L. Giglio, S. Korontzi, D. A. Chu, W. M. Hao, R.-R. Li, and C. O. Justice. (2003). Fires and smoke observed from the Earth Observing System MODIS instrument—Products, validation, and operational use, International Journal of Remote Sensing, 24, 1765–1781.

[18] Wooster, M. J., and Y. H. Zhang. (2004). Boreal forest fires burn less intensely in Russia than in North America, Geophys. Res. Lett.,31, L20505, doi:10.1029/2004GL020805.

[19] Online Journal of Space Communication. (2013). 'ASTER', http://spacejournal.ohio.edu/issue3/remote_sats2.html (accessed 16 December 2013).

[20] Lammert Kooistra, Aldo Bergsma, Beatus Chuma and Sytze de Bruin, Development of a Dynamic Web Mapping Service for Vegetation Productivity Using Earth Observation and in situ Sensors in a Sensor Web Based Approach [http://edepot.wur.nl/10876]

[21] Eberle, J. and Strobl C. (2012). 'Web-Based Geoprocessing and Workflow Creation for Generating and Providing Remote Sensing Products' - Geomatica Vol. 66, No. 1, pp. 13 to 26

[22] Brandon Macchero. (2013). 'About MODIS' [online], http://modis.gsfc.nasa.gov/about/ (accessed 11 November 2013).

[23] Brandon Maccherone. 'Design Concept' [online], http://modis.gsfc.nasa.gov/about/design.php (accessed 1 August 2013).

[24] Steven Graham. (2011). 'Moderate Resolution Imaging Spectroradiometer (MODIS)' [online], http://aqua.nasa.gov/about/instrument_modis.php (accessed 20 September 2013).

[25] NASA Distributed Active Archive Center (DAAC) at NSIDC. (2012). 'Frequently Asked Questions' [online], http://nsidc.org/data/modis/faq.html (accessed 14 September 2013).

[26] Brandon Maccherone. (2013). 'MODIS Data' [online], http://modis.gsfc.nasa.gov/data/ (accessed 8 October 2013).

[27] Spruce, J. P., Sader, S., Ryan, R. E., Smoot, J., Kuper, P., Ross, K., et al. (2011). Assessment of MODIS NDVI time series data products for detecting forest defoliation by gypsy moth outbreaks. Remote Sensing of Environment, 115, 427–437.

[28] Verbesselt, J., Zeileis, A., and Herold, M. (2012). 'Near real-time disturbance detection using satellite image time series', Remote Sensing of Environment, 123, pp. 98–108.

[29] Brandon Maccherone. 'Direct Broadcast' [online], http://modis.gsfc.nasa.gov/data/directbrod/index.php (accessed 4 August 2013).

[30] Brandon Maccherone. 'MODIS Data Product Non-Technical Description - MOD 02, 03' [online], http://modis.gsfc.nasa.gov/data/dataprod/nontech/MOD0203.php (accessed 7 August 2013).

[31] VectorNav Technologies. 'Importance of Industrial Grade Sensor Calibration' [online], http://www.vectornav.com/support/library?id=86 (accessed 18 October 2013)

[32] Brandon Maccherone. 'MOD 01 - Level-1A Radiance Counts' [online], http://modis.gsfc.nasa.gov/data/dataprod/dataproducts.php?MOD_NUMBER=01 (accessed 20 September 2013).

[33] Brandon Maccherone. 'MOD 02 - Level-1B Calibrated Geolocation Data Set' [online], http://modis.gsfc.nasa.gov/data/dataprod/dataproducts.php?MOD_NUMBER=02 (accessed 20 September 2013).

[34] Zhengming Wan. 'MOD11_L2 LST Product' [online], http://www.icess.ucsb.edu/modis/LstUsrGuide/usrguide_mod11.html (accessed 20 September 2013).

[35] Brandon Maccherone. 'MOD 03 - Geolocation Data Set' [online], http://modis.gsfc.nasa.gov/data/dataprod/dataproducts.php?MOD_NUMBER=03 (accessed 20 September 2013).

[36] U.S. Geological Survey. (2013). 'Surface Reflectance 8-Day L3 Global 500m' [online], https://lpdaac.usgs.gov/products/modis_products_table/mod09a1 (accessed 20 September 2013).

[37] Roy, D. P., Giglio, L., Kendall, J. D., and Justice, C. O. (1999). 'A multitemporal active-fire based burn scar detection algorithm', International Journal of Remote Sensing, 20, pp. 1031–1038.

[38] Justice, C., Townshend, J., Vermote, E., Masuoka, E., Wolfe, R., Saleous, N., et al. (2002b). 'An overview of MODIS Land data processing and product status', Remote Sensing of Environment, 83, pp. 3–15.

[39] D.P. Roy et al. (2008). Remote Sensing of Environment. 112 3690–3707

[40] Roy, D. P., Lewis, P., and Justice, C. (2002a). 'Burned area mapping using multi-temporal moderate spatial resolution data – a bi-directional reflectance model-based expectation approach', Remote Sensing of Environment, 83, pp. 263–286.

[41] Ruth Netting. (2013). 'Terra' [online], http://science.nasa.gov/missions/terra/ (accessed 4 August 2013).

[42] Brian Dunbar. (2013). 'Terra' [online], http://www.nasa.gov/mission_pages/terra/ (accessed 4 August 2013).

[43] Tassia Owen. (2011). 'Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER)' [online], http://terra.nasa.gov/ASTER/ (accessed 10 December 2013).

[44] Steven Graham. (2011). 'Cloud's and the Earth's Radiant Energy System (CERES)' [online], http://aqua.nasa.gov/about/instrument_ceres.php (accessed 10 December 2013).

[45] Tassia Owen. (2012). 'Multi-angle Imaging SpectroRadiometer (MISR)' [online], http://terra.nasa.gov/MISR/ (accessed 10 December 2013).

[46] Tassia Owen. (2012). 'Measurement of Pollution in the Troposphere (MOPITT)' [online], http://terra.nasa.gov/MOPITT/ (accessed 10 December 2013).

[47] Brian Dunbar. (2014). 'Aqua' [online], http://www.nasa.gov/mission_pages/aqua/ (accessed 5 January 2014).

[48] Steven Graham. (2013). 'Aqua' [online], http://aqua.nasa.gov/ (accessed 20 December 2013).

[49] Holli Riebeek. (2009). 'Earth Observatory' [online], http://earthobservatory.nasa.gov/Features/OrbitsM
(accessed 30 September 2013).

[50] Steven Kempler. (2011). 'Atmospheric Infrared Sounder (AIRS) Instrument Guide' [online], http://disc.sci.gsfc.nasa.gov/AIRS/documentation/airs_instrument_guide.shtml (accessed 30 November 2013).

[51] Dawn Conway. (2007). 'Advanced Microwave Scanning Radiometer - Earth Observing System' [online], http://wwwghcc.msfc.nasa.gov/AMSR/ (accessed 13 December 2013).

[52] Steven Graham. (2011). 'Advanced Microwave Sounding Unit (AMSU-A)' [online], http://aqua.nasa.gov/about/instrument_amsu.php (accessed 20 September 2013).

[53] Steven Graham. (2011). 'Humidity Sounder for Brazil (HSB)' [online], http://aqua.nasa.gov/about/instrument_hsb.php (accessed 12 December 2013)

[54] Wan, Z. 1999. MODIS Land-Surface Temperature. Algorithm Theoretical Basis Document. Version 3.3.

[55] Michael, C. and D. P. Ames. (2007). 'Evaluation of the OGC Web Processing Service for Use in a Client-Side GIS', OSGeo Journal.

[56] Django. (2013). 'Overview' [online], https://www.djangoproject.com/ (accessed 1 August 2013).

[57] Celery. (2013). 'Distributed Task Queue' [online], http://www.celeryproject.org/ (accessed 11 November 2013).

[58] Lalit Chandnani. (2013). 'Django Celery: Redis Vs RabbitMQ message broker' [online], http://blog.langoor.mobi/django-celery-redis-vs-rabbitmq-message-broker/ (accessed 5 December 2013).

[59] 'U.S. Geological Survey' [online], https://lpdaac.usgs.gov/data_access/data_pool (accessed 5 December 2013).

[60] 'NASA Distributed Active Archive Center (DAAC) at NSIDC' [online], http://nsidc.org/daac/about/ (accessed 18 August 2013).

[61] 'NASA Ocean Data' [online], http://oceandata.sci.gsfc.nasa.gov/products/product_level_desc.html (accessed 15 August 2013).

[62] Ed Masuoka. (2014). 'Level 1 and Atmosphere Archive and Distribution Sysytem' [online], http://ladsweb.nascom.nasa.gov (accessed 12 January 2013).

[63] Monique Jones. (2013). 'Advantages of using the web framework Django' [online], https://exploreb2b.com/articles/advantages-of-using-the-web-framework-django (accessed 3 August 2013).

[64] Piotrowski, Przemyslaw (July 2006). "Build a Rapid Web Development Environment for Python Server Pages and Oracle". Oracle Technology Network. Oracle. Retrieved 12 March 2012.

[65] Giglio, L. (2007). 'Characterization of the tropical diurnal fire cycle using VIRS and MODIS observations', Remote Sensing of Environment, 108, pp. 407– 421.