

Student thesis series INES nr 365

Evaluation of Standards and Techniques for Retrieval of Geospatial Raster Data

– A study for the ICOS Carbon Portal



Shirin Danehpash

2015
Department of
Physical Geography and Ecosystem Science
Lund University
Sölvegatan 12
S-223 62 Lund
Sweden



Shirin Danehpash (2015)

Evaluation of Standards and Techniques for Retrieval of Geospatial Raster Data

Master degree thesis, 30 credits in Geomatics

Department of Physical Geography and Ecosystems Science, Lund University

Level: Master of Science (MSc)

Course duration: January 2015 until June 2015

Disclaimer

This document describes work undertaken as part of a program of study at the University of Lund. All views and opinions expressed herein remain the sole responsibility of the author, and do not necessarily represent those of the institute.

Evaluation of Standards and Techniques for Retrieval of Geospatial Raster Data

– A study for ICOS Carbon Portal

Shirin Danehpash

Master thesis in Geomatics, 30 credits

August 2015

Supervisor Lars Harrie, PhD Associate professor
Lund University GIS Centre, Department of Physical Geography and Ecosystem
Science

Exam committee:

Harry Lankreijer, PhD Lecturer
Department of Physical Geography and Ecosystem Science

Roger Groth, Programmer
Lund University GIS Centre, Department of Physical Geography and Ecosystem
Science

Abstract

Evaluation of Standards and Techniques for Retrieval of Geospatial Raster Data – A study for ICOS Carbon Portal

Geospatial raster data represent the world as a surface with its geographic information which varies continuously. These data can be grid-based data like Digital Terrain Elevation Data (DTED) and geographic image data like multispectral images.

The Integrated Carbon Observation System (ICOS) European project is launched to measure greenhouse gases emission. The outputs of these measurements are the data in both geospatial vector (raw data) and raster formats (elaborated data). By using these measurements, scientists create flux maps over Europe. The flux maps are important for many groups such as researchers, stakeholders and public users. In this regard, ICOS Carbon Portal (ICOS CP) looks for a sufficient way to make the ICOS elaborated data available for all of these groups in an online environment. Among others, ICOS CP desires to design a geoportal to let users download the modelled geospatial raster data in different formats and geographic extents.

Open GeoSpatial Consortium (OGC) Web Coverage Service (WCS) defines a geospatial web service to render geospatial raster data such as flux maps in any desired subset in space and time. This study presents two techniques to design a geoportal compatible with WCS. This geoportal should be able to retrieve the ICOS data in both NetCDF and GeoTIFF formats as well as allow retrieval of subsets in time and space.

In the first technique, a geospatial raster database (Rasdaman) is used to store the data. Rasdaman OGC component (Petascope) as the server tool connects the database to the client side through WCS protocol. In the Second technique, an advanced file-based system (NetCDF) is applied to maintain the data. THREDDS as the WCS server ships the data to the client side through WCS protocol. These two techniques returned good result to download the data in desired formats and subsets.

Keywords: geospatial raster data, ICOS Carbon Portal, Web Coverage Service (WCS), Rasdaman, Petascope, NetCDF, THREDDS, OpenLayers

Acknowledgment

I would like to express my deepest gratitude to my supervisor, Lars Harrie, for his excellent guidance, patience, enthusiasm and providing me with an excellent environment for my master study.

Very special thanks go out to all members of ICOS Carbon Portal group for their technical supports and useful opinions about each parts of my thesis.

I would like also to thank my parents for all the moral support and encouragement they gave me over this project. Finally, I dedicate this thesis to my love, Mohammad Reza, for his kindness, patience and motivation he provided me through my entire project. Without him, I would never have been able to be successful in my work.

Table of Contents

Abstract.....	iv
Acknowledgment.....	v
Abbreviation and Dictionary.....	ix
1. Introduction	1
1.1 Background	1
1.2 Problem Context	2
1.3 Research Objective	4
1.4 Research Method.....	4
1.5 Disposition	5
2. GeoPortals for Geospatial Raster Data (state of arts)	7
2.1 Introduction	7
2.2 Geoportals.....	8
3. ICOS Carbon Portal and its user requirements	14
3.1 ICOS Carbon Portal.....	14
3.2 Constraints on this study from the ICOS Carbon Portal.....	16
3.3 User requirements	16
3.4 Evaluation requirements.....	16
4. Storage of Geospatial Raster Data	18
4.1 File Systems.....	18
4.1.1. <i>TIFF and GeoTIFF</i>	18
4.1.2. <i>NetCDF</i>	19
4.1.3. <i>NetCDF Application Program Interface(API)</i>	22
4.1.4. <i>Applications of NetCDF</i>	22
4.2 Databases.....	22
4.2.1. <i>Relational databases</i>	23
4.2.2. <i>Object-oriented databases</i>	23
4.2.3. <i>Object-relational databases</i>	24
4.2.4. <i>Array-databases</i>	24
4.2.5. <i>Rasdaman</i>	24
4.2.6. <i>Own presentation of Rasdaman</i>	27
4.2.7. <i>Applications of Rasdaman</i>	28

4.3	Selection of storage technique for geospatial raster data	29
5.	Distribution of Geospatial Raster Data	30
5.1	OGC Web Services.....	30
5.1.1.	<i>Web Coverage Service (WCS)</i>	31
5.1.2.	<i>Web Coverage Processing Service (WCPS)</i>	32
5.2	OPeNDAP	33
5.3	Applications of WCS and OPeNDAP	34
5.4	Selection of distribution standard	36
6.	Tools for WCS Server	38
6.1	Candidates of WCS server implementations	38
6.2	Petascop.....	38
6.3	THREDDS	39
6.4	Applications of Petascop and THREDDS.....	40
6.5	Selection of WCS server implementations	40
7.	Tools for WCS Client	41
7.1	Candidates of WCS client implementations	41
7.2	OpenLayers	41
7.2.1.	<i>Background</i>	41
7.2.2.	<i>Evaluation of OpenLayers</i>	42
7.3	QGIS	43
7.3.1.	<i>Background</i>	43
7.3.2.	<i>Evaluation of QGIS</i>	43
7.4	Selection of WCS client implementations.....	44
8.	Case Study.....	45
8.1	Introduction	45
8.2	Datasets	46
8.3	Methods.....	46
8.4	Case study 1-Rasdaman and Petascop.....	48
8.4.1.	<i>Implementation of Server</i>	48
8.4.2.	<i>First Evaluation of Petascop WCS Server</i>	50
8.4.3.	<i>Implementation of Client</i>	54
8.4.4.	<i>Result</i>	55

8.5	Case study 2- NetCDF and THREDDS.....	57
8.5.1.	<i>Implementation of Server</i>	57
8.5.2.	<i>First evaluation of THREDDS WCS Server</i>	59
8.5.3.	<i>Implementation of Client</i>	61
8.5.4.	<i>Result</i>	62
9.	Evaluation	64
9.1	Evaluation of case study 1- Rasdaman and Petascope	64
9.1.1.	Evaluation of Retrieval of Subsets in Space	64
9.1.2.	Evaluation of Retrieval of Subsets in time	67
9.2	Evaluation of case study 2- NetCDF and THREDDS.....	69
9.2.1.	Evaluation of Retrieval of Subsets in space	69
9.2.2.	Evaluation of Retrieval of Subsets in time	73
10.	Discussion	75
10.1	Case study 1- Rasdaman and Petascope.....	75
10.2	Case study 2- NetCDF and THREDDS.....	76
10.3	Future work.....	77
11.	Conclusions	78
	References	81
	Appendix A-GetCapabilities Response	85
	Appendix B-DescribeCoverage Response	87
	Appendix C- OpenLayers JavaScript and HTML Code	89
	Appendix D- THREDDS Configuration Catalog file in importing data	94
	Seminar Series	95

Abbreviation and Dictionary

API	Application Programming Interface
CRS	Coordinate Reference System
CS-W	Catalog Service Web
GeoTIFF	Geo Tagged Image File Format
GML	Geography Markup Language
ICOS	Integrated Carbon Observation System
ICOS CP	Integrated Carbon Observation System Carbon Portal
KVP	Key Value Pair
NetCDF	Network Common Data Form
OGC	Open Geospatial Consortium
OPeNDAP	Open-Source Project for a Network Data Access Protocol
Petascop	Rasdaman OGC component
RASBASE	Rasdaman Database
Rasdaman	Raster Data Manager
Rasdl	Rasdaman Definition Language
Rasml	Rasdaman Manipulation Language
Rasql	Rasdaman Query Language
TDS	THREDDS Data Server
THREDDS	Thematic Realtime Environmental Distributed Data Services
TIFF	Tagged Image File Format
WCPS	Web Coverage Processing Service
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
WPS	Web Processing Service

1. Introduction

1.1 Background

Geospatial data refer to an object or phenomena located on the specific scene in space, in relation with the other objects (Kraak & Ormeling, 2011). In other words, geospatial data refer to data objects which are linked to geometry and topology (Rigaux *et al.*, 2001). Geometry defines the size, location and shape of the spatial objects whereas topology presents the spatial relationships between spatial objects including adjacency and overlapping. Unlike to non-geospatial data which only involve with numbers and characters, geospatial data are more complicated and require complex operations (Shekhar & Xiong, 2008). Geospatial data can characterize real world features with continuous boundaries as well as real world phenomena with discrete boundaries. This dichotomy is resulted to generate two main model classes of geospatial data storage: the field-based model to represent the continuously geographic phenomena such as raster and the object-based model to represent the discrete entities like vector data (Worboys & Duckham, 2004).

Geospatial raster data are categorized in two types: grid-based data and geographic image data. Grid based data is a rectangular grid consists of many cells spread over an area. These cells have the same size to define the resolution of the grid. Digital terrain elevation, land use information and many others are some example of grid-based data. Image data is generated from the sensors by e.g. satellite remote sensors. Image data can be consisted by many bands called multispectral or hyperspectral images. Increasing of different remote sensors is resulted to mass production of geospatial image data (Shekhar & Xiong, 2008).

Appropriate selection of storage techniques helps users to work more flexibly, easily and quickly. There are various approaches used to store geospatial raster data. These include standard file structures as well as database management systems which allow the indexed information to be stored in a structured or organized repository. Each approach has its advantages and disadvantages. On one hand, standard file-based structures provide good compression and require less storage space, but it is difficult to index the data resulted in slower retrieval operation. While on the other hand, properly indexed database storages allow quicker retrieval of geospatial raster data but at the same time can require storage of millions of significantly sized records (Carter *et al.*, 1995).

Besides the rapid growth of geospatial raster data, there is an increasing demand for flexible geospatial raster data retrieval (Baumann *et al.*, 2003). Consequently, a few standards are introduced to facilitate the complicated processing of multi-dimensional geospatial raster data. These standardizations are executed by e.g. Open GeoSpatial Consortium (OGC) to make geoservices interoperable. OGC Web Coverage Service (WCS) is designed to provide geospatial raster data accessible for further processing (Whiteside & Evans, 2008). A new OGC service with processing functionality is called Web Coverage Processing Service (WCPS). The main aim of WCPS generation is to navigate, extract and evaluate the large and multidimensional coverage repositories (Baumann *et al.*, 2003). Another open-source protocol is introduced to facilitate and promote access to the data through the network: Open-source project for a Network Data Access Protocol (OPeNDAP). This protocol is mainly designed to provide the easy access to the Network Common Data Form (NetCDF) of data (Baart *et al.*, 2012). At the end, users apply each of these protocols according to their requirement to retrieve the data.

1.2 Problem Context

The challenges present in working with geospatial raster data are related to three important components: I) *storage and management systems*, II) *standardized services* and III) *software interface of geospatial raster data*. Each component has its own importance in the aim of improving the interaction with geospatial raster data. A proper geospatial raster data *storage and management system* makes it easy to classify, search and retrieve the desired data. A *standardized service* is needed to unify, access, process and share these data among other users since most of the data are different in data format, coordinate reference systems and geometry models. The last challenge is choosing suitable *software interface* to support the standardized services.

The efficient storage and management systems of geospatial raster data are becoming an important factor to interact with these data. Commonly, geospatial raster data are mainly stored in simple file systems without index structures. Accordingly data retrieval, query processing and complex manipulation takes substantial time. Furthermore, a simple file structure does not support enough security, availability, reliability and manageability for the geospatial raster datasets. Due to a rapid increase in the amount of geospatial raster data being acquired by the high-resolution remote sensors, simple file-based methods have limitations to store and retrieve

geospatial raster data. In this order, the usage of **database systems** and **advanced file systems** (e.g. NetCDF) for storage and distribution of geospatial raster data is emerging gradually (Baumann, 2001). Use of database lessens the complexity of storage and retrieval of data because it provides the ability for users to view the data in their own way. Consequently, the views of the data deliver a level of security and a mechanism to customize the appearance of the database. Moreover, the usage of advanced-file based systems like NetCDF has many advantages. The main advantage of NetCDF is its self-describing. It means that software using the NetCDF can easily read the data and determine the structure, the variables and the information about the data. Also, the information useful to reduce the incidence of the errors is available in this type of storage format (Nativi *et al.*, 2005). In this study the data are stored in either an array database (Rasdaman) or in an advanced-file based system (NetCDF).

Although it would be possible to only provide the user with files to download, the file format and data structure are not always compatible with the user's requirement. OGC standards offer some solutions to develop an open geospatial web service to meet many user requirements (OGC, 2015). In this study we evaluate the use of OGC services and other protocols (e.g. OPeNDAP) for downloading geospatial raster data. One portal that requires downloading geospatial raster data is the ICOS carbon portal. This portal is used for distribution of ICOS geospatial raster data (described in chapter 3) to the general public as well as for the business, science, and educational communities. ICOS carbon portal should support downloading of both standard file and tailored data products.

One of the greatest challenges in this study is an interaction with the standardized service and user. There are a few products of software to support standardized services and provide the high qualified web mapping functionalities to interact with users. OpenLayers and QGIS are two Open-Source software which support OGC services.

In summary, there are limitations for retrieval of geospatial raster data which are stored in simple file based systems. Also, even if the geospatial raster data are stored in databases or advanced file-based systems, without using standardized services only a few users have access to these data. These shortcomings negatively impact the usage and distribution of geospatial raster data and may cause them to be inefficient. To improve the interaction with geospatial raster data, a service oriented technique has several advantages.

1.3 Research Objective

The general aim of this study is to evaluate standards and tools for retrieval of geospatial raster data.

The specific aims are:

- Study of the standards WCS and OPeNDAP for distribution of geospatial raster data.
- Comparison of WCS servers created by Petascope (Raster database server tool) and THREDDS as a server tool.
- Evaluation of OpenLayers and QGIS as a library to create WCS clients.

1.4 Research Method

First of all, as shown in figure 1.1, the requirements to retrieve the geospatial raster data in ICOS Carbon Portal project are defined. The techniques and information for storage of geospatial raster data and distribution standards are gathered through the literature research. The proper techniques for geospatial raster data storage and distribution standards are selected based on the requirements. After that, the appropriate tools compatible with storage and standards techniques are implemented and evaluated on both server and client sides. In this regard, two case studies are considered. In these two cases, two different techniques are used to fulfil the user requirements. At the end, an evaluation is done to show which of these techniques best supports the ICOS CP user requirements as well as to identify some advantages and disadvantages with each technique.

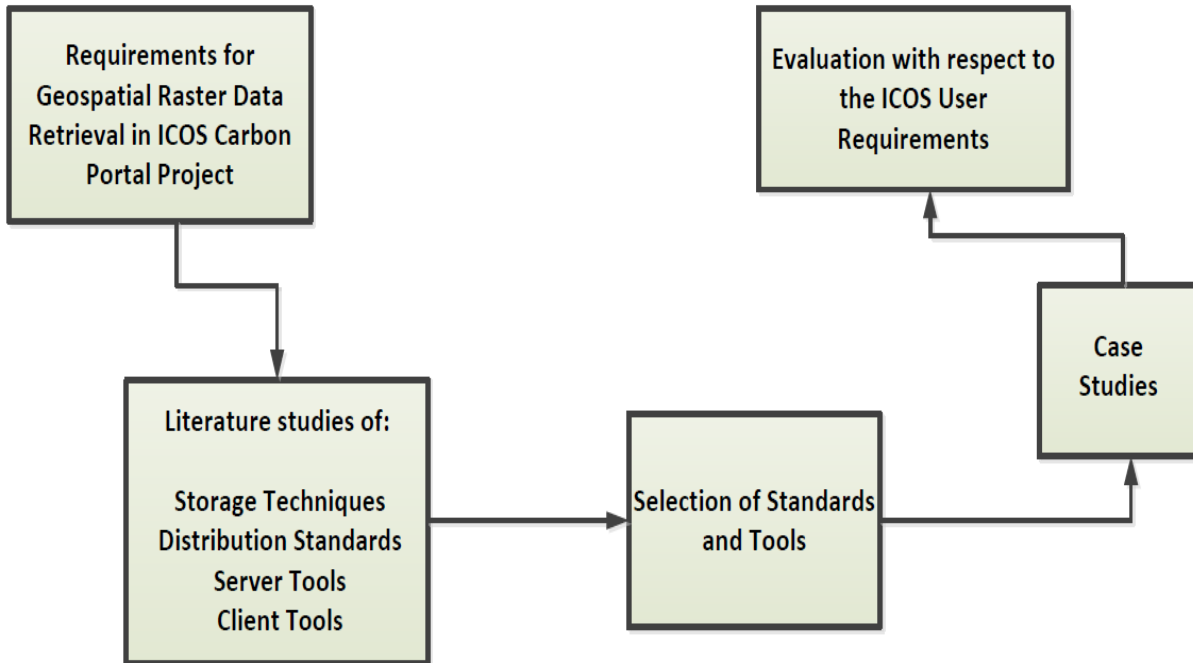


Figure 1.1. An overview of the Research Method's Structure.

1.5 Disposition

The remainder of this study is organized as follow (of figure 1.2): In section two, the state of arts in application of geospatial raster data are reviewed. Section 3 describes ICOS Carbon Portal and its user requirements. In section 4, we review the file-based approach which is the most common approaches in geospatial raster data storage. Additionally, in this section databases are described with focus on the ones used for geospatial raster data. In section 5, distribution of geospatial raster data are surveyed briefly. Section 6 and 7 describe tools for WCS server and client sides. Section 8 is the implementation part with the presentation of two case studies and the result of each one. The results are then evaluated in the evaluation section and weak points of each case study are described in discussion section. Finally, the conclusions summarize the result of this study. A list of explanations of the most used abbreviations is given by the beginning of the thesis.

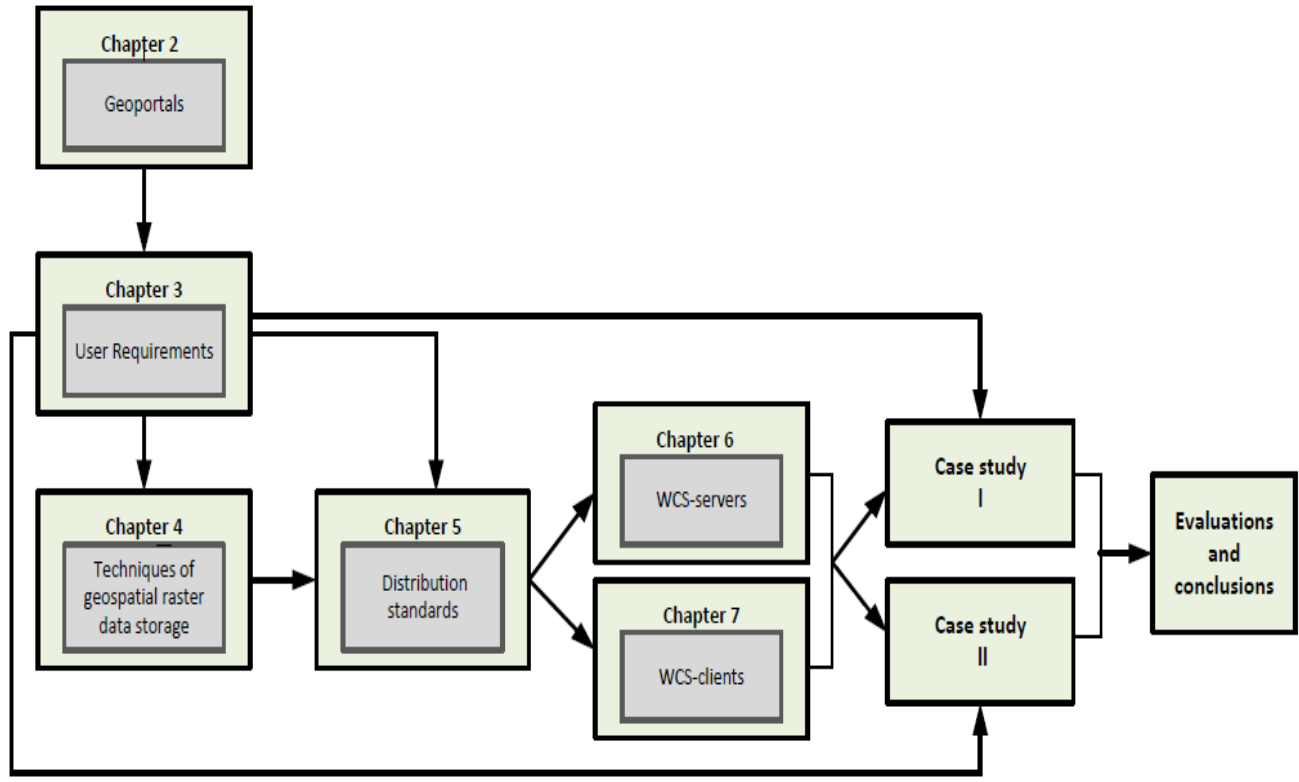


Figure 1.2. The disposition of this master thesis.

2. GeoPortals for Geospatial Raster Data (state of arts)

2.1 Introduction

There is an increasing demand of organizations to present and analyse their own data for public users on the internet (Karabegovic & Ponjavic, 2012). To satisfy this increased demand for the usage, distribution and exchanging of the data, techniques to support interoperability are required. Web portals are an entry point in the web environment to bring the information and data available from different resources to numerous users (Tatnall, 2005). Web portals do not store the data but they offer a centralized interface to access to the distributed resources of the data (Zhang *et al.*, 2006). A geoportal is a kind of web portal applied to find and access to geospatial information, data and the services for displaying, editing, analysing, etc. (Goodchild *et al.*, 2007). Geospatial data providers like government agencies and commercial sources use geoportals to publish the metadata of their own geospatial data. Geospatial data consumers use geoportals to find and access their required geospatial data (Karabegovic & Ponjavic, 2012). Therefore geoportals has an important role in the sharing of geospatial data. They prevent the users from duplicated efforts, inconsistencies, delays, confusion, and wasted resources (Manso & Bernabé, 2005).

Geoportals are generally created on three levels: *spatial web service platform*, *catalogue service*, and *spatial data warehouse* (Karabegovic & Ponjavic, 2012). A *spatial web service platform* contains Web GIS services often proposed by OGC. OGC designed the web-based geographic information and services accessible through the web environment (Lu, 2005). The OGC Web Services includes viewing services like Web Map Service (WMS), Accessing services like Web Feature Service (WFS) and Web Coverage Service (WCS), as well as Processing services like Web Processing Service (WPS) and Web Coverage Processing Service (WCPS) (described in section 5). *Catalog services* give information about web services and the data available from these services. Finally, the third level is the *data warehouse* i.e., the place for storage of the data (Karabegovic & Ponjavic, 2012). These three components of geoportals design a general workflow for all server-client architecture. As shown in figure 2.1, the mentioned services process the geospatial data in the spatial data warehouse based on the client request then the processed results are sent back to the client.

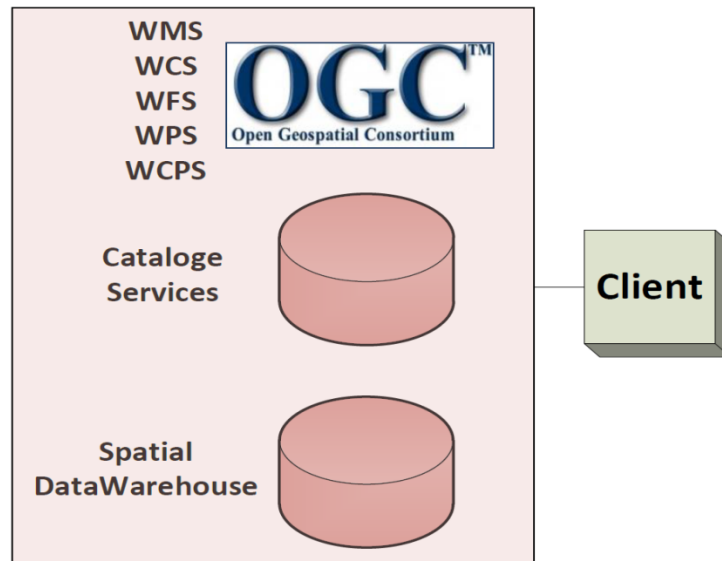


Figure 2.1. A general architecture of the Geoportals designed by OGC services.

Next section addresses three geoportals designed for geospatial raster data. The first portal addresses the research on the portal used to present and render the US cropland data. The second portal discusses a research related to the greenhouse gases fluxes from different surface of the Europe. The third portal focuses on the research study about presenting and prediction global climate change until 2100 by NASA.

2.2 Geoportals

CropScape Geoportal

The Cropland Data Layer (CDL) consists of the crop and land cover classifications data for United States. This product has been used in many applications like disaster assessments, land use researches, agricultural sustainability studies and etc. As using traditional methods including thematic map, CD/DVD media, etc. did not let users access to the US cropland data in convenient way, so Han *et al.* (2012) created an interactive web system (CropScape, 2015). The purpose of this system was to provide the ability for user to query, visualize and analyse the data through geospatial services in a publicly accessible environment. This system used the yearly CDL product which was distributed in GeoTiff format. The CDL product was obtained from US Department of Agriculture (USDA)/ National Agricultural Statistical Service (NASS) from 1997 to 2010 using satellite imagery from Advanced Wide Field Sensor (AWiFS) and Landsat TM 5 and ETM+ 7. The mid resolution, huge volume of the data and large spatial coverage of CDL

raster data, presents a big challenge to how to provide the web application of CDL data online across the internet. In order to achieve the answer of this question, general three layers were adopted to share CDL geospatial data. The first layer, *Application layer*, contained of Open-source JavaScript libraries, Goggle Earth and OpenLayers to develop rich internet application. These elements designed the CropScape client interface to offer an interactive and responsive way to view, query, download and analyse the CDL data. The second layer, *Service layer*, included open Web geospatial data services and some limitation tools to improve the web service abilities. WMS, WFS, WCS and WPS were used in this application. And the last layer, *data layer*, contained raster data included the CDL layers, a background map layer and the vector data included different boundary layers like states, counties and etc. The process of managing these three layers together was same as the process of all geoportal as described previously. Figure 2.2 illustrates the framework of the CropScape architecture.

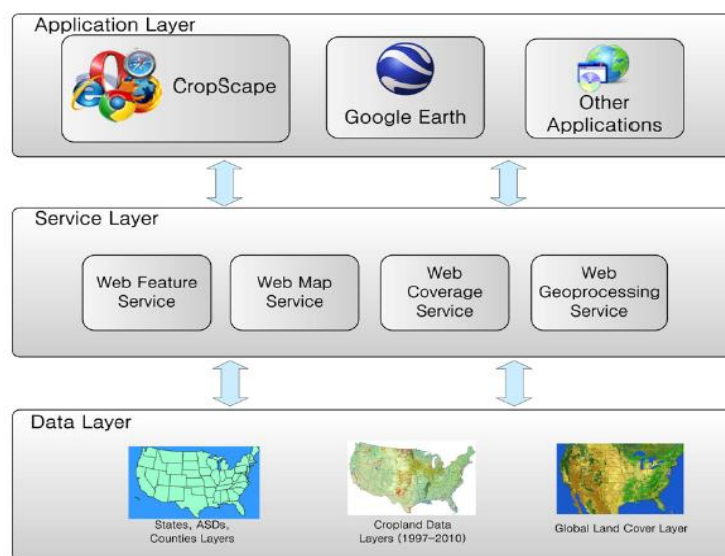


Figure 2.2. System Architecture of CropScape geoportal. This geoportal lets user retrieve the United States cropland data (Han et al., 2012).

The implemented CropScape was officially released on January, 2011. In the first five months of designing this system, around 17,000 users had visited this application to access to the crop land data. Now, CropScape is a user friendly web application with effective online geospatial capabilities of CDL data. It allows users to apply useful functionalities to view and query data without installing any geospatial software or information system. So, CropScape has proved that

it is possible to serve a high performance web application over the internet in spite of having the huge volume of the raster format. Figure 2.3, shows an overview of CropScope user interface.

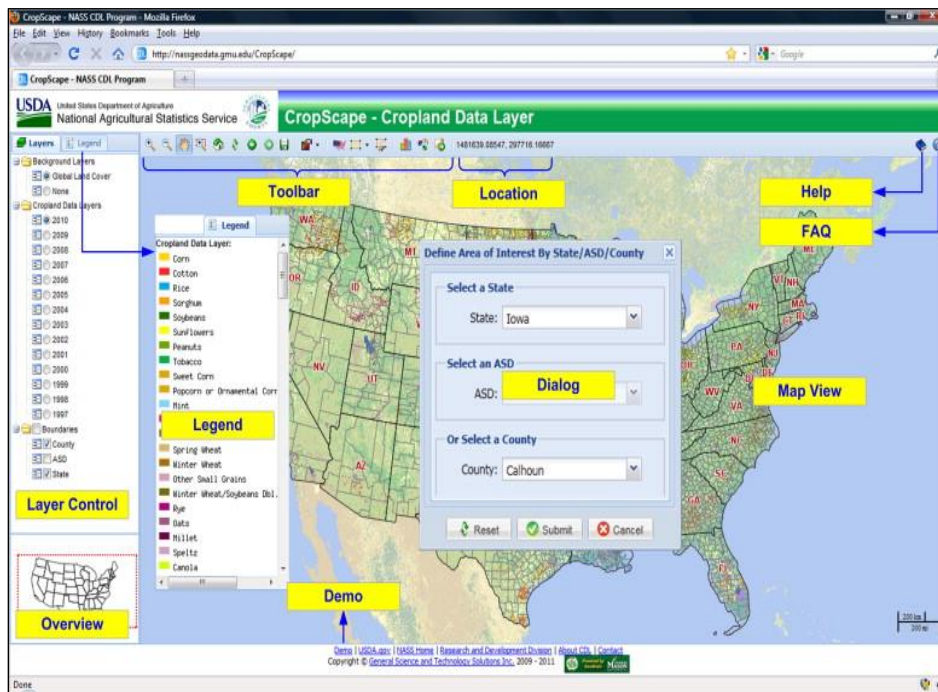


Figure 2.3. CropScope user interface (CropScope, 2015).

CarboScope Geoport

The atmospheric concentration of greenhouse gases carbon dioxide (CO₂), methane (CH₄) and carbon monoxide (CO) have globally increased during the last 200 years. These three gases have the most important effects on the global warming. Researchers are looking for a good solution to estimate and reduce the increase of gases. CarboScope (CarboScope, 2015) is an exploring tool focused on greenhouse gases (CO₂, CH₄ and CO). CarboScope provides a user-friendly web service to make a comparison between these three gases emitted from different surfaces in Europe. It also renders the information about these gases and the methods used to estimate their surface fluxes. The data applied in CarboScope were obtained as the output from the calculation of the modelling tools. The OGC standards were used to provide this web service. This service is not still completed but as shown in figure 2.4, it gives the ability to users to view the data and their information obtained from different models and inversions. These data can be shown for any desired time and space.

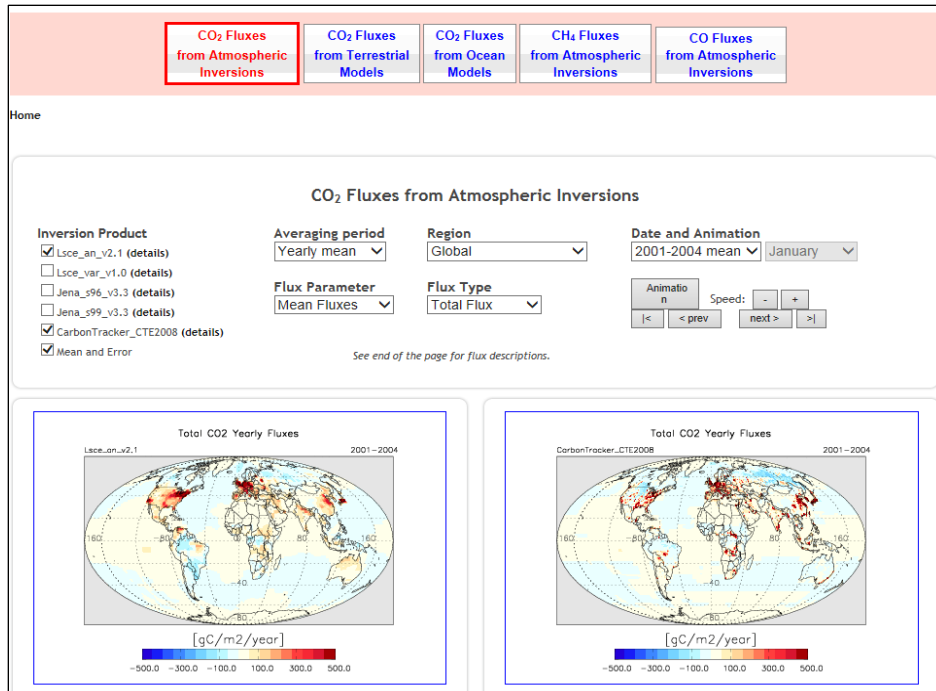


Figure 2.4. CarboScope user interface (CarboScope, 2015).

NASA Global Climate Change Geoportal

As the growing concentrations of greenhouse gases in Earth's atmosphere have an important effect on the temperature and rainfall worldwide, NASA has released a global climate change application. This application shows the behaviour of temperature and rainfall that are changing under the effect of the growing greenhouse gases by year 2100.

The data used in this system include downscaled projections from mathematical climate models and measurement tools. Each of the climate models consists of maximum temperature, minimum temperature, and precipitation as a month average from 1950 through 2005 and from 2006 to 2099 (prediction data). In order to make the data available for researchers and public users, NASA used THREDDS Data Service (TDS) to store the data. Then by using different protocols, these data get accessible for the users as an application (NASA, 2015). Figure 2.5 shows an overview of this NASA web-based map application. In this application, user has the ability to define the specific region (name of city, country or continent) to retrieve its maximum temperature, minimum temperature, and precipitation in a specific month from 1950 to 2100. Figure 2.6 shows an example to clarify how this application delivers the data to the user. In this

example, the predicted maximum temperature data for in North Africa, Middle East and Northern India for 2100 are retrieved.

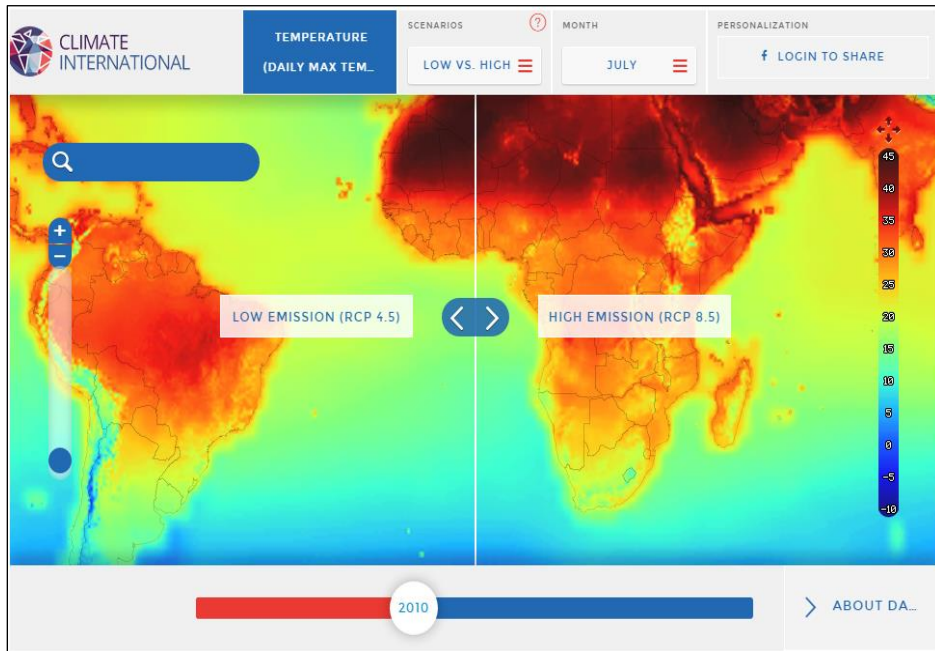


Figure 2.5. An overview of the NASA application to predict the temperature behavior under effect of greenhouse gases. (NASA, 2015)

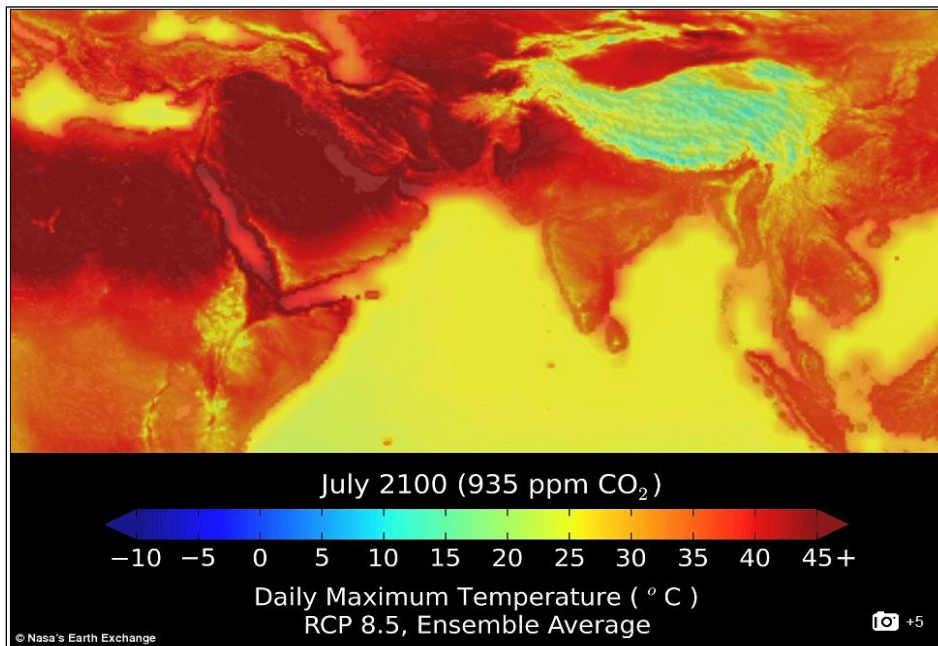


Figure 2.6. An example of NASA application output. This figure shows maximum temperatures in North Africa, the Middle East and Northern India more than 45°C by 2100. (NASA, 2015)

This web-based map application provides the data at a resolution of around 25 km, between 1950 and 2100 for the whole world.

This map service is the latest product of NASA's Earth Exchange. Unlike the other prediction map applications which want to show the difference of the temperature to its current level, this application gives the predicted value of temperature. By this application, scientists and planners can predict the risk of drought, floods, heatwaves, agricultural productivity reduction, etc. (NASA, 2015).

The geoportals reviewed in this chapter indicate the importance of using geoportals. They provide a representative set of tools for viewing and downloading geospatial raster data.

3. ICOS Carbon Portal and its user requirements

3.1 ICOS Carbon Portal

Climate change is one of the main concerns of humans to cope with in the coming years. As explained, most of the changing of the climate is caused by the emission of greenhouse gases coming from fossil fuels uses, industrial processes, deforestation and etc. (Buchwitz *et al.*, 2013). To measure the greenhouse gas concentrations, the Integrated Carbon Observation System (ICOS) European project is launched (Stakeholders-Handbook, 2013). This project is managed by eight European countries and includes around 100 measuring stations all over Europe.

Figure 3.1 shows that ICOS consists of three main components: the measurement stations, the thematic centres and the ICOS Carbon Portal (ICOS CP). The measurement stations consist of station networks and many other elements to measure greenhouse gases emitted from oceans, ecosystems and atmosphere (icos-cp, 2015). The second component of ICOS, the thematic centre, consists of three components: Atmospheric, Ecosystem and Oceanic thematic centres. The Atmospheric thematic centre is designed for the management of atmospheric measurements and online data processing. Ecosystem thematic centre is aimed to manage the ecosystem flux measurements, component fluxes, data processing and instrument development. Ocean Thematic Centre is planned for the management of continuous marine observations, initial data processing from marine network (Stakeholders-Handbook, 2013). The third ICOS component, the ICOS Carbon Portal, is the focus of this study.

The ICOS CP is a virtual place where all the ICOS users from stakeholders to general public users can have an access to the ICOS data. The ICOS CP is intended to combine the advanced web-based techniques to distribute the ICOS data. The ICOS CP has the following tasks:

- Maintenance of the ICOS data and information about them.
- Guarantee the safe back-up storage and data achieving for long time.
- Provide the viewing, accessing and processing tools to facilitate the retrieval of ICOS data.
- Make available data for all the users all over the world specifically researchers, authorities and decision makers.
- Provide the ability to have an interface with other portals around the world.

The ICOS data are divided in two main groups. The first group is the ICOS own data products (raw data). These data are the observation data in time series formats obtained from the measurement tools. These data are in the vector format. The second group is the ICOS elaborated data product. These data are the output of the models partly based on ICOS observations and typically created by researchers outside of ICOS. These data are often in raster formats. This study is aimed to work on the elaborated data and design a small part of ICOS CP project to manage these data.

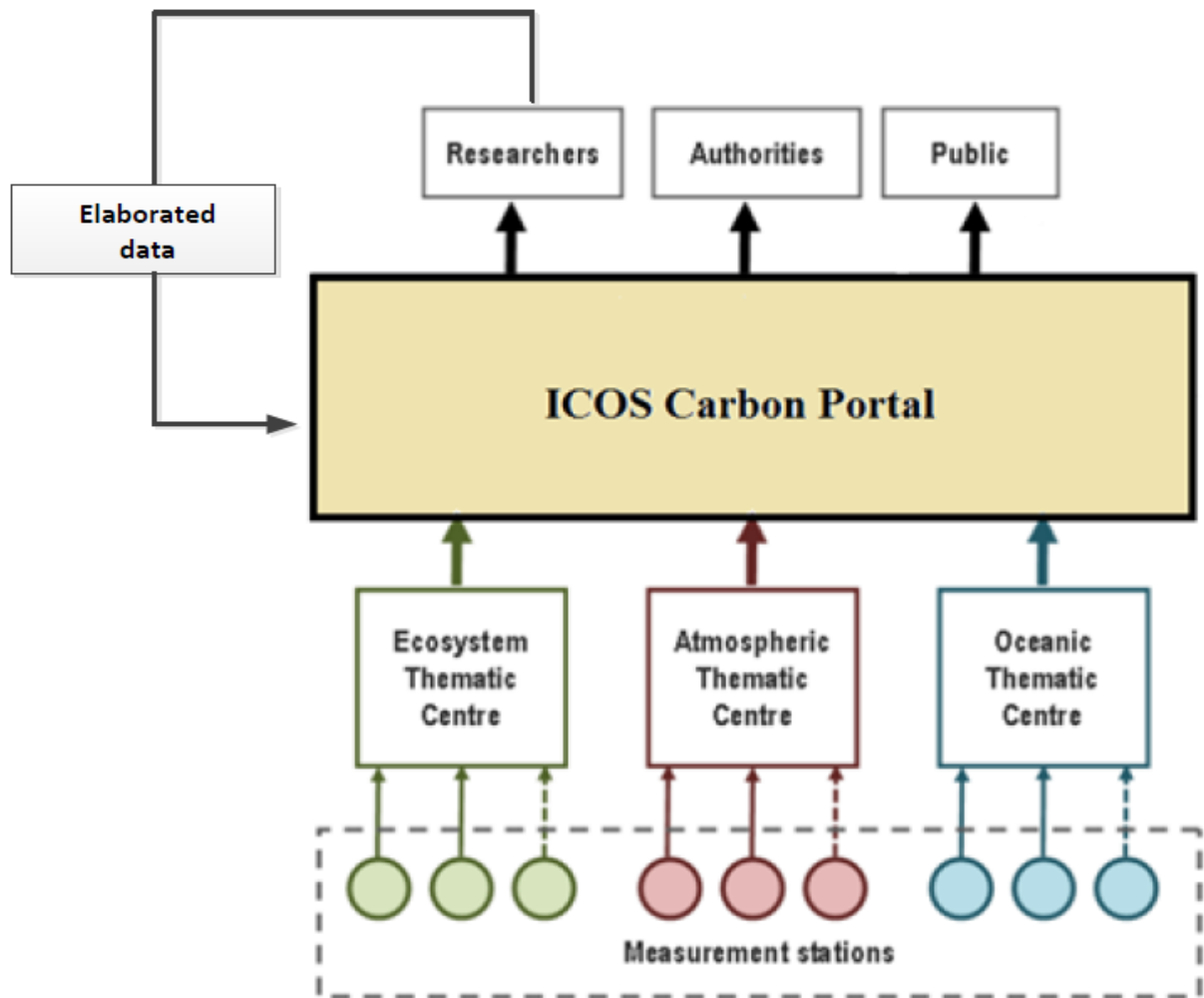


Figure 3.1. ICOS Project Components. This figure also shows the relationship between ICOS CP and elaborated data which are the focus of this study (inspired from icos-cp, 2015).

3.2 Constraints on this study from the ICOS Carbon Portal

The technical solution of ICOS CP is not yet finalised. This study is part of the process of learning about possible technical solutions for the portal. That there is a link to the development of ICOS CP also set a framework to this study on e.g. the storage environment of the geospatial raster data. In ICOS CP there are (at this stage in spring 2015) two possible candidates for storing the geospatial raster data:

- an array database (possibly rasdaman)
- an advanced file-based system (NetCDF is the best candidate).

Therefore the technical solutions in this study should be based on either of these two storage environments.

3.3 User requirements

The final user requirements of ICOS CP are not settled. In this study we use some preliminary user requirements to evaluate technical solutions of the distribution of geospatial raster data in ICOS CP. Parts of the requirements are based on a user survey of ICOS CP performed in 2014. It should be noted that the user requirements below are not covering the whole extent of ICOS CP; only user requirements that affect this study are of interest here. The requirements are based on a discussion with Lars Harrie in the ICOS CP project.

The requirements are that the user should be able to:

- Visualize the ICOS geospatial raster data.
- Download the measurement data (ICOS L2 data) as well as the ICOS geospatial raster data; this download could be a subset of a whole dataset both in terms of time and space.
- The download format for the geospatial raster data should be NetCDF and GeoTIFF.

3.4 Evaluation requirements

In this study, an evaluation is made of distribution standards as well as tools for the server and the clients. Therefore we have to translate the user requirements above into requirements of the different parts. Then the selection of standard and tools are based on these requirements.

Requirement of the standard for downloading data

- Retrieval of the data in different formats.

- Retrieval of subsets of the data.

When the standard is selected, we need tools that support this standard both on client and server side. Below we list the requirements for the server and the clients.

Requirements of the server side

- Support the download standard (including the requirement of the standard above).
- Support the NetCDF and GeoTIFF formats.

Requirement of the client

- Support the download standard (including the requirement of the standard above).
- Display the data in different formats (GeoTIFF and NetCDF) based on the server ability.

4. Storage of Geospatial Raster Data

4.1 File Systems

“[A] File is a collection of record which contains the related data” (Connolly & Begg, 2005, p-4). When data are stored in the file, this storage system is called file-based system. Simple file-based system is an early attempt to organize and store geospatial raster data. The first and simple traditional design of geospatial raster file-based systems support only limited image size. These simple systems are not efficient in making sub queries; they are used to send the whole file to the client and let the user manipulate the file (Xie, 2008). Today, some advanced file-based systems are designed to store geospatial raster data. These systems do not have the common limitations as simple file-based systems.

Two advanced file-based systems appropriate in storing geospatial raster data are described in the next sections.

4.1.1. TIFF and GeoTIFF

The tagged image file format (TIFF) has appeared as one of the most popular formats to store and display raster data. The most common application of TIFF format is in the logotypes and scanned documents and is popular among graphic artists and photographers. This format is compatible with many applications such as manipulation applications, 3-D imaging applications, photography and publishing applications (Mahammad & Ramakrishnan, 2003).

TIFF files are very useful and popular image format due to their high quality in comparison with other formats such as JPEG. In contrast, the problem of TIFF formats is their large size. Consequently, there are some options to make them smaller by using compression. Two main compression options which are used in editing and saving the TIFF data format are ZIP and LZW. Both of them are lossless compression options (Ritter & Ruth, 1997).

Lossless means no quality of image is lost due to the compression. Lossless guarantees that the data can always be saved and readable without any data corruption. So this feature of TIFF format makes it popular among the users who want to archive master copies of images. Therefore, as TIFF format doesn't suffer a compression loss in editing and saving the data.

Although TIFF has a lot of ability to work very flexible with scanner devices, software, etc., it has many limitations in many applications such as cartographic applications (Ritter & Ruth,

1997). The main problem is that it is almost impossible to store the geographic information of data with the data together in the TIFF format. Therefore, in order to transfer satellite images with TIFF format (from one system to another one), the user intervention is required to locate the image in the suitable geographic orientation beside their metadata (Ritter & Ruth, 1997).

The geo tagged image file format (GeoTIFF) is designed to provide a set of tagged image file format (TIFF) associated with TIFF imagery. GeoTIFF format has the ability of storing georeferencing information beside the geospatial raster data. It means that GeoTIFF is a kind of metadata format which makes a connection between the data and the information about the data. This information can be about the reference system, map projection and etc. (Ritter *et al.*, 2000). Generally, GeoTIFF is used for description, transfer and storage of geographic raster imagery (Mahammad & Ramakrishnan, 2003). This kind of format is applied in some GIS software which handle the geospatial raster data.

Although GeoTIFF is still one of the popular formats in storing geospatial raster data, the coordinate system definition between the GeoTIFF format and the most remote sensing geospatial raster data is different (Remotesensing, 2011). Consequently, this problem has induced researchers to think about a better dataset format for storage of geospatial raster data.

4.1.2. NetCDF

The network common data form (NetCDF) is a network common data access technique used to store many geospatial raster and structured data format (Rew & Davis, 1990; Rew *et al.*, 1997). At the beginning, NetCDF was designed to be used in the meteorological science but now it is one of the most common formats in the raster storage (Cong-cong & Li-ying, 2010). The main advantage of using NetCDF is its flexibility. Flexibility means the NetCDF could be read and written by some software (e.g. ArcGIS) for different user requirements (Gao *et al.*, 2009). NetCDF consists of three main parts: 1) dimensions 2) variables and 3) attributes (Jianwei *et al.*, 2003; Cong-cong & Li-ying, 2010).

Dimensions are used to denote the real physical dimension. It means that dimensions are applied to define the frame of the variables (Jianwei *et al.*, 2003). Time, longitude, latitude, etc. can be defined as a dimension in a NetCDF file. Dimension has two features: 'name' and 'length'. 'Name' refers to the title of dimension. 'Length' refers to the size of dimension. It should be a positive integer number. Also, unlimited length can be assigned as a length of a dimension. A

variable with an unlimited dimension can increase its length without any limitation. Below is an example of NetCDF dimension:

Example 4.1. The NetCDF dimensions with defined name and length. Lon, lat, tstep and day are the dimensions of this NetCDF file.

```
netcdf Tair_daily_WFDEI_201211 {  
  dimensions:  
    lon = 720 ;  
    lat = 360 ;  
    tstep = UNLIMITED ;  
    day = 30 ;
```

Variables are used to store huge amount of the data. A variable has predefined characteristics (data type, name and a shape) which are defined by a list of dimensions. These characteristics should be specified when a variable is created. Variable types can have different formats: character, float, short, double, byte and int. The shape of variables is defined by its dimensions. Some attributes can be assigned to a variable where it is possible to delete or change it after the creation of the variable (Rew & Davis, 1990). Some variables have the same name as a dimension. If a variable has the same name as a dimension, is called a coordinate variable. Example 4.2 shows a float NetCDF variable consisted of three dimensions.

Example 4.2. A NetCDF variable. Tair is a variable with float type and created by tstep, lat and lon dimensions.

```
variables:  
float Tair(tstep, lat, lon) ;  
  Tair:title = "Tair average" ;           //Tair's attribute  
  Tair:units = "K" ;                     //Tair's attribute  
  Tair:long_name = "Average Near surface air temperature \n", //Tair's attribute  
    " at 2 m at time stamp" ;           //Tair's attribute  
  Tair:actual_max = 311.1187f ;         //Tair's attribute  
  Tair:actual_min = 223.294f ;         //Tair's attribute
```

Attributes refer to the comments or explanation for variables and dimensions (Cong-cong & Liying, 2010). In the NetCDF file, most attributes provide the information about a particular

variable. In example 4.2, five titles about Tair’s attribute are shown. In addition, the attributes which provide metadata of the whole dataset are called global attributes (of example 4.3)

Example 4.3. *Global NetCDF attributes. It provides metadata about the whole dataset.*

Global attributes:

```
:Title = "WATCH Forcing Data methodology applied \n",
      "to ERA-Interim data" ;
:Note1 = " Daily averages of 3-hourly WFDEI file" ;
:Note2 = "Tair elevation- & bias-corrected using \n",
      "CRU TS3.21 mean monthly temperature \n",
      "and mean diurnal temperature range" ;
:Info = "www.ecmwf.int/products/data/archive \n",
      " /descriptions/ei/index.html" ;
:Institution = "Met Office, JCHMR, Wallingford, UK" ;
:History = "created July 2013" ;
```

As a summary, figure 4.1 illustrates a general overview of all elements of a NetCDF file.

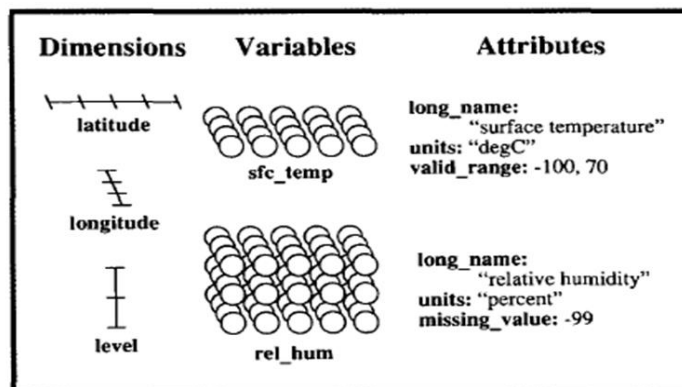


Figure 4.1. *An example of a NetCDF file. This file has three dimensions and two variables. Each variable has three attributes. (Rew & Davis, 1990)*

The information (metadata) about the variables, dimensions and attributes are stored in the file header of the dataset. Each variable is described by its name, shape, array size etc. in the file header while the array values are stored in the array data part of the dataset (Jianwei *et al.*, 2003).

4.1.3. NetCDF Application Program Interface(API)

The NetCDF APIs are supported in C, C++, Fortran 77, Fortran 90 and Java (Unidata-NetCDF, 2015). The NetCDF Java API is a complete implementation of NetCDF in Java. As some of the open-source GIS web-based software are based on the Java language, this API is commonly used in the software compatible with NetCDF advanced file-based system (Unidata-NetCDF, 2015). This advantage of NetCDF induces this study to store the ICOS data in the NetCDF system. By storing the ICOS data in the NetCDF system, it would be possible to link these data to many GIS servers and applications by NetCDF Java API.

4.1.4. Applications of NetCDF

NetCDF is mostly applied in climatology, meteorology and oceanography applications. It is an input/output format for some GIS applications, and for general scientific data exchange. Many well-known organizations such as NASA, NOAA, EUMETSAT Data Centre, US Navy, etc. use NetCDF as their input/output format (Unidata-NetCDF, 2015).

One of the well-known projects using NetCDF is NOAA's Climate Analysis Branch (CAB). This project is concentrated on studies of climate changes on time scales of months to centuries. CAB climatological data is stored and maintained in NetCDF format (NOAA, 2015).

4.2 Databases

Simple file-based systems cannot keep up with all user expectations in many aspects in relation to raster data storage (Connolly & Begg, 2005):

- Data are isolated in different files. Accordingly, for retrieving data with a specific feature, the application developer should search in many files to find the desired data. Hence, when the number of files increases, this difficulty will increase as well.
- Controlling the duplication of data in the file-based system is really difficult since a property of an object may appear in many files. Duplication has a result in wasting time and costs. In addition, it takes substantial space of the machine.

In summary, simple file-based systems do not allow flexible and scalable storage, fast querying and complicated manipulation of geospatial raster data (Xie, 2008).

Database management system (DBMS) was initiated in the 1960s, in the Apollo moon-landing project. On that time there was no system to handle and manage the huge amount of information generated by this project. Consequently, a database system was designed in the aim of providing a general view of data and processing of data storage and manipulation. Now, databases have the ability to control the data redundancy and security. Also, they can develop the accessibility and integrity of data. (Connolly & Begg, 2005). Conversely, the cost and the time-consuming in designing the database are much more in comparison with designing the file-based systems (Davidson & Aundhe, 2001).

Since the 1960s, many database models have been developed of which several can store geospatial raster data.

4.2.1. Relational databases

In the relational database management system (RDBMS), each data is embedded within tables. Each table has its own name and fields that store attributes. Additionally, each row has one value for each attribute (Weikum *et al.*, 2009). The positive point in using the relational databases is their simple logic structure (Connolly & Begg, 2005).

Unfortunately, the RDBMS which is designed to store arrays such as unstructured BLOB (binary large object: like images, animations, etc.) cannot support the huge amount of arrays in a database (Paredaens *et al.*, 2012). In addition, relational tables in RDBMs are in one dimension and they are not suitable in supporting the multidimensional data like geospatial raster data (Weikum *et al.*, 2009).

4.2.2. Object-oriented databases

The object-oriented database management system (OODBMS) was developed in 1980s due to its high ability in storing the large amount of data (Rieu & Nguyen, 1986; Nguyen & Rieu, 1987). OODBMS can capture and store complex objects (array) directly. Also, they have this ability to call the objects directly or through using a query interface (Bagui, 2003).

In spite of many achievements which are gained by OODBMSs, they have not been able to be used in raster data storage because of the weaknesses still present in OODBMSs. These databases have a lack of query facilities such as sub queries, set queries and etc. in managing and manipulating the data. They do not provide the security in using the data as well (Bagui, 2003).

4.2.3. Object-relational databases

Object-relational database management system (ORDBMS) has been designed to improve the object-oriented features within RDBMSs (Wang, 2010; Elmasri & Navathe, 2011). In other words, ORDBMS fills the gap which exists between RDBMSs and OORDBMSs (Frank, 1995). Since objects are stored in tables, ORDBMS can be defined as a relational model extended by the object-oriented model (Wang, 2010).

According to Stonebraker and Hellerstein (1998), these systems are not enough flexible in using query language for unlimited array data in size and dimension. It means that some of the arrays as data type constructor so-called (“template”) are not supported by ORDBMS (Stonebraker & Hellerstein, 1998; Ritsch, 1999). Furthermore, some of the fundamental optimization techniques such as pipelining and parallelization cannot be supported by ORDBMS (Baumann, 2001).

4.2.4. Array-databases

Array database management systems (ArrayDBMS) are designed to store Multi-dimensional Discrete Data (MDD). Geospatial raster data of arbitrary in their size and dimensions are called MDD (Ritsch, 1999). As MDD involves a massive number of operations, ArrayDBMS develops the database with query support and a MDD modeling (Akerkar, 2013).

These databases facilitate the access to the MDD by providing query languages. By this way, users can create, manipulate, search and delete the MDD. In these conditions, the implementation time with traditional database would be intolerable (Akerkar, 2013).

There are a large number of ArrayDBMS used in different fields. One example of this technique is Rasdaman.

4.2.5. Rasdaman

By considering all the issues of the other raster data storage techniques, Rasdaman (Raster Data Manager) was designed. Rasdaman is the result of the European ESPRIT long term research project in 1996 (Baumann *et al.*, 1998). The aim of this project was to establish a database to support MDD arisen from Centro Nacional de Información Geográfica and Hospital General de Manresa (Baumann *et al.*, 1998; Ritsch, 1999). Rasdaman is developed by the Bavarian Research Center for knowledge-based systems at Munich and Jacobs University (Ritsch, 1999).

Rasdaman is an open-source database management system which supports multidimensional geospatial raster data with arbitrary size and dimension. Generating raster data management in Rasdaman with the aim of storing MDD is carried out by extending query language (Data Definition Language and Data Manipulation Language), Application Program Interface (API), communication modules, query processor (optimizer and executor) and storage manager.

Rasdaman architecture is based on two-tier client-server that all query executions are implemented on the server internally (Baumann *et al.*, 1998; Ritsch, 1999). The client and server in the Rasdaman are on two different sides that can be placed on two different machines (Ritsch, 1999). The Rasdaman server side is designed to act as a middleware to make a connection between the client side and DBMS (Baumann *et al.*, 1998; Ritsch, 1999).

The language of Rasdaman is Rasql relies on standard SQL. Rasql consists of two conceptual languages: Rasdaman Definition Language (Rasdl) and Rasdaman Manipulation Language (Rasml).

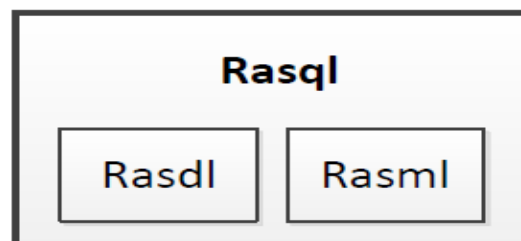


Figure 4.2. An overview of Rasql structure (simplified from Baumann (2013)).

Rasdl is used to create and manipulate the schema information as attributes (Ritsch, 1999). It means that Rasdl is used to create a database into the Rasdman. It gives the ability to a user to add any new type of data to the system. These types of data can be identified by their name, so, they must have the unique name in the database. Rasdl contains of three categories of data: Cell type, MDD type and Collection type (Baumann, 2013).

Cell type should be created as the base type of MDD type definitions. *Cell type* can be base type like float or be composite type like red, green and blue.

MDD type defines arrays or collection element types with spatial domain. Spatial domain can be defined with different degree of freedom. It can be created by a bounding box which its lower and higher bounds should be defined in it (example 4.4).

Collection Type describes a set of some MDD types.

In this regard, many kind of type definition can be defined in a Rasdl description. Example 4.4 shows an example to depict the Rasdl structure in a database creation for a RGB (Red,Green, Blue) raster data:

Example 4.4. *The Rasdl structure in database creation.*

```
struct RGBCell { char red, char green, char blue };           //this line shows the cell type
typedef marray< RGBCell, [1:800,1:600] > RGBImage;          //this line shows the MDD type
typedef set< RGBImage > RGBSet;                             //this line shows the collection type
```

In this example, cell type, MDD type and collection type are defined for a set of RGB image with 800* 600 dimensions.

Note: it is not necessary for users to use Rasdl language if they do not want to create a new database. Most of the time, they can use the default created database in Rasdaman as a name of RASBASE (Rasdaman database). They just need to create their own collection based on their data type.

Rasml provides the ability to manipulate, retrieve and update the queries by following the syntax of SQL language. The basic Rasml grammar consists of three main parts (Baumann, 2013):

Select: chooses the specified attributes

From: chooses the collection type which the specified attributes should be taken from them

Where: defines a condition to be executed on the query

Example 4.5. *The Rasml structure.*

```
Select <mdd_exp1>... <mdd_expn>
From <coll1>...<colln>
Where <mdd_boolean_exp>
```

At the end, in order to view Rasdaman query results visually, Rview as the desktop visualization component of Rasdaman provides this ability. It lets users see their query results without connecting to any specific client application.

In the next section, a presentation of Rasql language is shown to make it clear how Rasdaman works.

4.2.6. Own presentation of Rasdaman

Some example of Rasql query language is shown to make it clear how Rasdaman works. In this regard, a NetCDF file is inserted into the RASBASE. This NetCDF file is the one which was used in section 4.1.2 that shows the average temperature for November 2012. Then some queries are defined for this dataset:

- Create a collection ‘test’ of type ‘FloatSet3’ since the file has Float type with three dimensions.

```
Rasql -q ‘create collection test FloatSet3’ --user rasadmin --passwd xxx
```

- Print the name of all existing collection to be sure the collection is created.

```
Rasql -q ‘select r from RAS_COLLECTIONNAMES as r’ --out string
```

- Import a NetCDF file into the ‘test’ collection (‘Tair’ is a dimension name of the file which shows the temperature values).

```
Rasql -q ‘insert into test values inv_netCDF($1, “vars=Tair” )’ - file  
Tair_daily_WFDEI_201211.nc --user rasadmin --passwd xxx
```

- Extract the information of the imported data.

```
Rasql -q ‘select dbinfo(c) from test as c’ --out string | grep Result
```

- Show the imported data dimensions.

```
Rasql -q ‘select sdom(m) from test as m’ --out string | grep Result
```

- Extract the temperature value for the grid cells with (55°N, 13°E) coordinate on 3th of the November in 2012 (these coordinates refer to the corners of the cells).

```
Rasql -q ‘select m[3 , 55, 13] from test as m’ --out hex | grep Result
```

- Extract the temperature value for a subset of time and dimension. For example extract the data for the first to fifth day of the month and for the locations which their latitude is between (50° to 65°) and their longitude is between (10° to 16°).

```
Rasql -q ‘select m[1:5 , 50:65, 10:16] from test as m’ --out hex | grep  
Result
```

- Extract the temperature value for the previous subsets while their values are more than 30.

```
Rasql -q ‘select test[1:5,50:65,10:16] from test where some_cells (new[1:5,  
55:65, 10:16] > 30)’ --out hex | grep Result
```

- encode a 2D slice for every MDD in 3D collection ‘new’ and encode it to GeoTIFF, setting a bounding box and crs appropriately:

```
Rasql -q 'select encode( c[0, **], "GTiff", "xmin=25; ymin=40;
xmax=75; ymax=75; crs=EPSG:4326; INTERLEAVE=BAND; metadata=\" some
metadata\" \" ) from test as c' --out file
```

- Delete all ‘very dark’ values of collection ‘test’ with all pixel values lower than 30:

```
Rasql -q 'delete from test as a where all_cells (a<30)' --out string
```

4.2.7. Applications of Rasdaman

Due to the high abilities of Rasdaman, it is used in many sciences such as earth sciences (EarthLook, 2015), space sciences, life sciences, multimedia, etc and it is used by many famous organizations like NASA Ames Research Center, European Centre for Medium-Range Weather Forecast, Meteorological Environmental Earth Observation, etc. (Rasdaman, 2015c).

One of the interesting Rasdaman applications is its usage in life science. Few years ago, the hospitals and medical centres did not have the ability to save and retrieve the patient records such as the medical images, MRI images, x-ray, ultrasound, etc. They recognize if they have stored the patient records, they would have much cost, time and energy savings. Unfortunately, maintenance of these huge amounts of data was not possible for user friendly systems. Therefore, U.S National Library of Medicine (NIH) decided to use Rasdaman to maintain their data. Rasdaman helps to archive all kind of the digital data independent of its origin, dimensionality, and size. In addition, as Rasdaman has the analyzing ability of the data, it is so useful for the hospitals to analyze the patient records. Figure 4.3 shows the ‘Visible Human’ service designed foR NIH by Rasdaman. This service is a 3D data set composed from horizontal slices through a human body. Each slice of this cube reveals the human anatomy (VisibleHuman, 2015).



Figure 4.3. The 'Visible Human' service used Rasdaman. The data are stored in Rasdaman and show human anatomy. By using the left slider, this figure shows different locations of the body by retrieving data from Rasdaman. (EarthLook, 2015)

4.3 Selection of storage technique for geospatial raster data

There are currently two options for storing ICOS data in ICOS CP: Rasdaman and NetCDF. The reasons are:

Rasdaman supports ICOS data with arbitrary size, dimension and base type. This allows storing the huge amount of ICOS data together in the database. Also, Rasdaman has a very rich query language that makes retrieving ICOS data very flexible, including exporting as a file. Additionally, it has an OGC component to provide the ability to access to the OGC services in the most compatible way.

NetCDF is commonly used in the carbon modelling community. The output of the NetCDF format is very different from the other formats used in GIS community because it represents several parameters that can be varied in three spatial dimensions (time, latitude and longitude) (Nativi *et al.*, 2005). Second, as explained, NetCDF Java API allows the user to work with the most of the GIS applications. Finally, OGC services have the high performance in transferring the NetCDF datasets to the client applications.

5. Distribution of Geospatial Raster Data

5.1 OGC Web Services

Open Geospatial Consortium (OGC) is the main actor in standardizing the geo services to be interoperable (OGC, 2015). Around 500 companies, government agencies and universities are members in the OGC organization (OGC, 2015).

OGC Web Services (OWS) are the services which are defined by OGC to provide all kinds of geospatial functionalities. These services are self-describing and can be presented on the web or within any local network (Rautenbach *et al.*, 2013). OWS requests are examined by Hyper Text Transfer Protocol (HTTP) protocol and their information is encoded by XML (Extensive Markup Language) or KVP (Key Value Pair). These services are classified in four groups: visualization services, access services, processing services and catalog services (Kiehle *et al.*, 2007).

Visualization services allow the geospatial data to be displayed. Web Map Service (WMS) is an example of visualization services used to display the maps. Web Terrain Service is another example used to display three-dimensional data like terrain models.

Access services provide the ability for access to geospatial data through downloading. Web Feature Service (WFS) is an access service used to make an access to geospatial features which are encoded in GML. An OGC service which provides an access to spatio-temporal coverages is called Web Coverage Service (WCS) (Whiteside & Evans, 2006). WCS is used to make an easy access to geospatial raster data coverages with different dimensions and render them over the World Wide Web.

Processing services provide the functionality to process geospatial data. Web Processing Service (WPS) and Web Coverage Processing Service (WCPS) are laid in this class of services. These services execute some processing including coordinate system transformation, geometric calculation, filtering, analysis, etc.

Catalog services are used for describing, registering, searching, maintaining and accessing to the information about geospatial data available in the network. Catalog service-web (CS-W) is the most widely known catalog service.

As explained in previous chapters, the aim of this study is to provide an access to geospatial raster data through OGC Web Services. In this regard, WCS is playing an important role to facilitate access to ICOS carbon portal data in both space and time dimensions.

5.1.1. Web Coverage Service (WCS)

WCS provides access to geospatial raster data in forms that are useful for client-side rendering. WCS permits clients to select portions of a server's information holdings based on spatial constraints and other query criteria (OGC, 2015).

The WCS coverage structure defines geospatial raster data as bearing three components: a unique name for coverage, multi-dimensional geospatial raster data (coverage), and metadata to describe coverage. A unique name should be defined to address the specific coverage easily by its name. Coverage can have two dimensions (x and y), three dimensions (x, y and z) or four dimensions (x, y, z and t). Each coverage has an information list (metadata). This list not only shows all required information about the coverage, but also shows the coordinate reference systems associated in which the coverage can be queried (Baumann & Chulkov, 2007).

The common request structure for WCS standard is described in figure 5.1. First, the client requests the capabilities of the service by the *GetCapabilities* request. In response to a *GetCapabilities* request, an XML document is returned that explains available operations and Coverages (Appendix A). There is another request as a name of *DescribeCoverage* used to list the information about coverage (Appendix B). The returned request can be phrased in two ways: HTTP GET request which uses KVP and HTTP POST request using XML/GML language (Baumann & Chulkov, 2008). Finally, according to the obtained information from *DescribeCoverage* request, the client requests for coverage retrieval. This request is called *GetCoverage*. The response to a valid *GetCoverage* request consists of two files: an XML file containing coverage information and the coverage file in the requested format. The coverage file can be included of full coverage or a subset of the coverage in **space** through the bounding box provided in the *GetCoverage* request. Furthermore, in the *GetCoverage* request, the user has the ability to retrieve a subset of the coverage in **time**. Also, it is possible to define some desired parameters including supported formats, scale, reprojection, interpolation methods, etc. A conceptual overview of relationship between WCS server and client is depicted by figure 5.1.

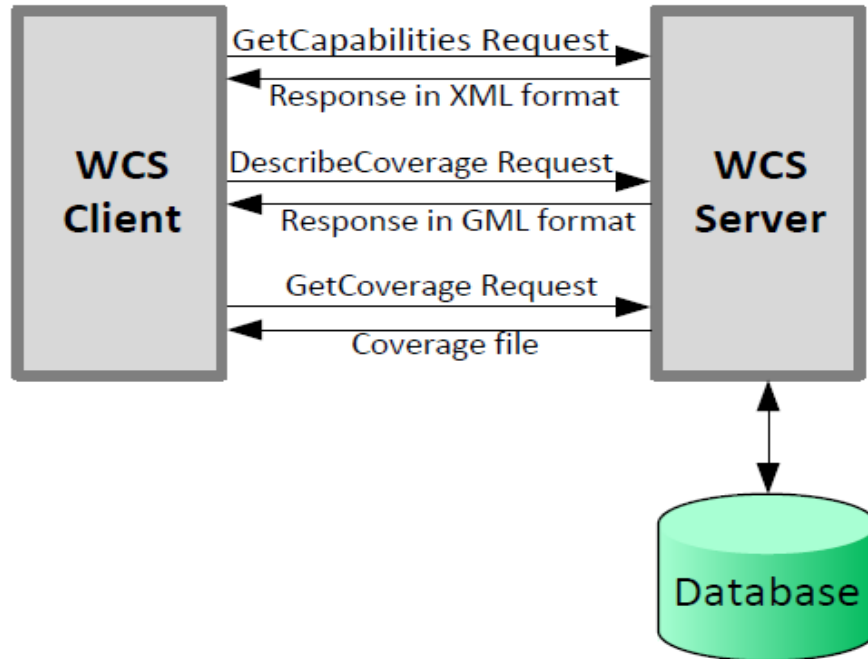


Figure 5.1. A WCS Client-Server Architecture.

This figure depicts a typical communication with a WCS service. WCS client sends a request (*GetCapabilities*, *DescribeCoverage* or *GetCoverage*) to the WCS server. WCS server accepts the request from the WCS client. Then the server generates a desired response by accessing to the database and transmits back the response to the WCS client.

5.1.2. Web Coverage Processing Service (WCPS)

Web Coverage Processing Service (WCPS) provides a language for retrieval and processing of the geospatial raster data. It allows the client to ask for the processing of a coverage on the server. WCPS offers three operations: *GetCapabilities*, *DescribeCoverage*, and *ProcessCoverage* (figure 5.2). The *GetCapabilities* operation, like in WCS, delivers an XML document. This XML document not only consists of the information about operations and coverages but also delivers specific processing service capabilities. Like in WCS, the *DescribeCoverage* operation provides the information about the coverages served by a particular WCPS server. The response of this operation is delivered in an XML/GML document. The *ProcessCoverage* operation provides the ability to process and analyze the coverages stored on the server. It also extracts the information from the coverages. Finally, the requests are formed to

be phrased in a defined processing language supporting coverage expressions. Then, result coverages can be sent back to the client (Baumann & Chulkov, 2008).

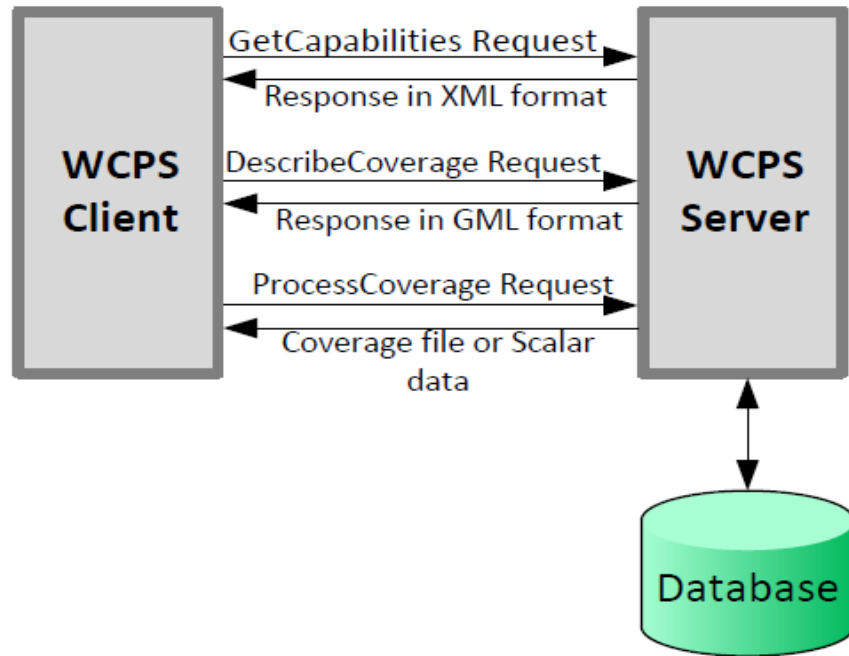


Figure 5.2. A typical communication with a WCPS-service.

5.2 OPeNDAP

Open-source Project for a Network Data Access Protocol (OPeNDAP) is a free protocol or data transporter designed to be used by earth scientists. The objective of developing OPeNDAP was to make an easy access to data through the network (Cornillon *et al.*, 2003). OPeNDAP standard is designed specifically for geospatial raster data based on Hypertext Transfer Protocol (HTTP). (Hankin *et al.*, 2010). The data served by OPeNDAP is often in NetCDF or HDF format. A key aspect of OPeNDAP is its ability to extract the subset of the files and unify the data from numerous files in one transfer operation (Baart *et al.*, 2012).

OPeNDAP makes the data accessible to the remote locations without considering the primary storage format. In the OPeNDAP architecture a client-server model is used. This is exactly the model used by the World Wide Web (WWW) where client programs (browsers) submit requests to web servers. So, in the OPeNDAP client-server model, a client sends requests to retrieve the data from the server side. The OPeNDAP server renders the data to the client program in the

desired format requested by the client. Figure 5.3 shows the general architecture of using OPeNDAP in the client-server model.

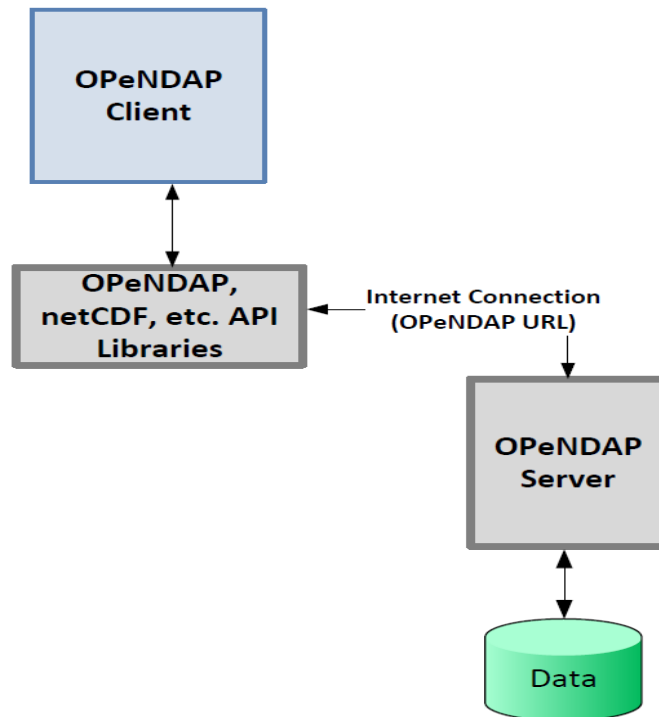


Figure 5.3. General architecture of OPeNDAP used in a client-server model.

5.3 Applications of WCS and OPeNDAP

Baltic Sea Hydrographic Commission (BSHC) has provided the ability for users to have the official bathymetry data for all Baltic Sea countries in one place and download the data freely. This commission used the WCS technology in order to render the bathymetry data to the users (Baltic-Sea, 2015). Figure 5.4 shows an overview of this geoportal using the WCS to deliver the data in ASCII and GeoTIFF formats. By using a WMS interface, the service also provides the data in image formats such as PNG.

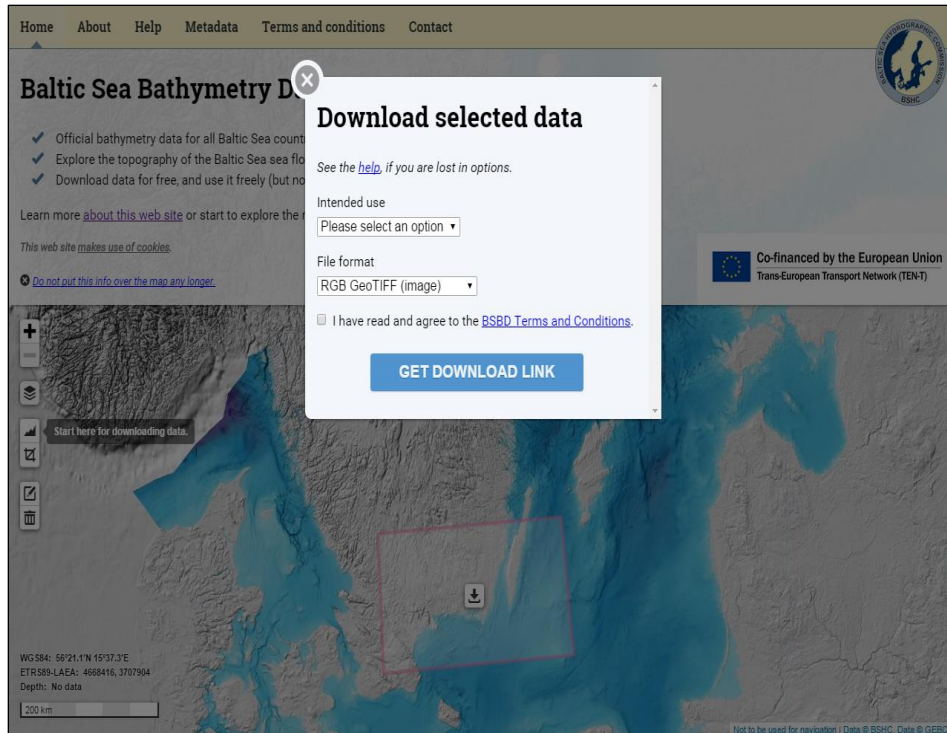


Figure 5.4. Baltic Sea Bathymetry Geoportal used WCS and WMS Standards in order to download the Baltic sea bathymetry data (Baltic-Sea, 2015).

OPeNDAP is considered by governmental agencies including National Oceanic and Atmospheric Administration (NOAA), National Science Foundation (NSF), Australian Bureau of Meteorology and NASA in order to present and render satellite images, observed earth science data and many others (Baart *et al.*, 2012). Ozone Mapping Instrument (OMI) project designed by NASA is one of the specific projects which OPeNDAP is used in it.. In this project an instrument is located on the Aurora spacecraft for continuously monitoring of the global and regional position of Ozone layer. Then the obtained data get available through OPeNDAP protocol to the users. Figure 5.5 shows an overview of the data available by OPeNDAP (NASA-OMI, 2015).

Name	Last Modified	Size	DAP Response Links	Webstart
Parent Directory/				
OMIAC0e_003/	2015-01-03T06:49:32	-	- - - - -	
OMIACUVd_003/	2015-01-03T06:19:30	-	- - - - -	
OMIQA03e_003/	2015-01-03T06:19:36	-	- - - - -	
OMIACR_003/	2010-12-09T14:21:15	-	- - - - -	
OMIQA2d_003/	2015-01-02T20:09:22	-	- - - - -	
OMIQA2e_003/	2015-01-03T06:19:38	-	- - - - -	
OMIQA3d_003/	2015-01-03T06:19:38	-	- - - - -	
OMIQA3e_003/	2015-01-03T06:19:38	-	- - - - -	
OMIACUVd_003/	2015-01-05T20:02:08	-	- - - - -	

THREDDS Catalog XML Hyrax development sponsored by NSF, NASA, and NOAA

[OPeNDAP Hyrax \(1.8.4\) Documentation](#)

Figure 5.5. An Overview of retrieving the Ozone layer data through OPeNDAP tool (NASA-OMI, 2015).

5.4 Selection of distribution standard

WCS is a useful protocol when a user wants to download a large part of the dataset. It means that it is suitable for the users who do not want to get into the details of the dataset. In contrast, when a user has the good knowledge about the datasets, WCS is not more efficient than OPeNDAP. When a user is familiar with a dataset, she can cache the indices required to access a specific part of the datasets or she can automate the calculation of the indices. In this condition, OPeNDAP is the most useful protocol (Baart *et al.*, 2012). Table 5.1 shows a comparison between WCS and OPeNDAP protocols.

Table 5.1. A comparison between WCS and OPeNDAP (Baart *et al.*, 2012, p-260).

	OPeNDAP	WCS
Querying	index	coordinate
Reprojection	no	yes
Dimensionality	n	4 (x,y,z,t)
Metadata	CF Convention	OWS Common
Unstructured grids	possible, not standardized	standardized, not possible
Response type	arrays + attributes	xml + file

According to table 5.1, it is really difficult to state which protocol is better than the other according to the ICOS user requirements. Each of them has its own advantages. Based on the user requirements section, it is essential to retrieve the data in different subsets. As OPeNDAP is essentially index-based, querying and retrieving subsets are too difficult for users with little programming experience. In contrast, it is easy with WCS to bin rectangular grids in different coordinate systems and retrieving the space subsets based on the created rectangular (Baart et al., 2012). Moreover, there are more free servers and clients designed to work with WCS in comparing with OPeNDAP (FOSS4G, 2011). In addition, WCS provides this ability to reproject the data to the other projections. In this regard, in this study WCS is chosen to be the middleware between server and client sides.

Additionally, WCS can support the data stored in the NetCDF format. So, it is useful for ICOS data which are stored in the NetCDF format.

6. Tools for WCS Server

6.1 Candidates of WCS server implementations

Previously it was decided that the two possible storage environments are NetCDF and Rasdaman. We select one WCS server implementation linked to each of these storage environments. One of these WCS server candidates is Petascope linked to the Rasdaman database and the other candidate is THREDDS linked to the NetCDF advanced file system. In the next sections, the architecture of these servers and the reasons of choosing these servers are described.

6.2 Petascope

Petascope is a layer of Rasdaman which has been designed to implement the OGC interface standards such as WCS, WCPS, WCS-T, WPS and WMS. This Java servlet package is a kind of middleware (in the sense of OGC) that lets both client and server to understand their language (Aiordăchioaie & Baumann, 2010). It relies on the Tomcat as servlet container and PostgreSQL as relational database system. As shown in figure 6.1, when Petascope receives a request from the OGC client, it translates the request into the Rasql to make it readable for Rasdaman. After that, Rasdaman implements some processing on the request and forwards back the result to the Petascope. Finally, Petascope sends the result to the OGC client.

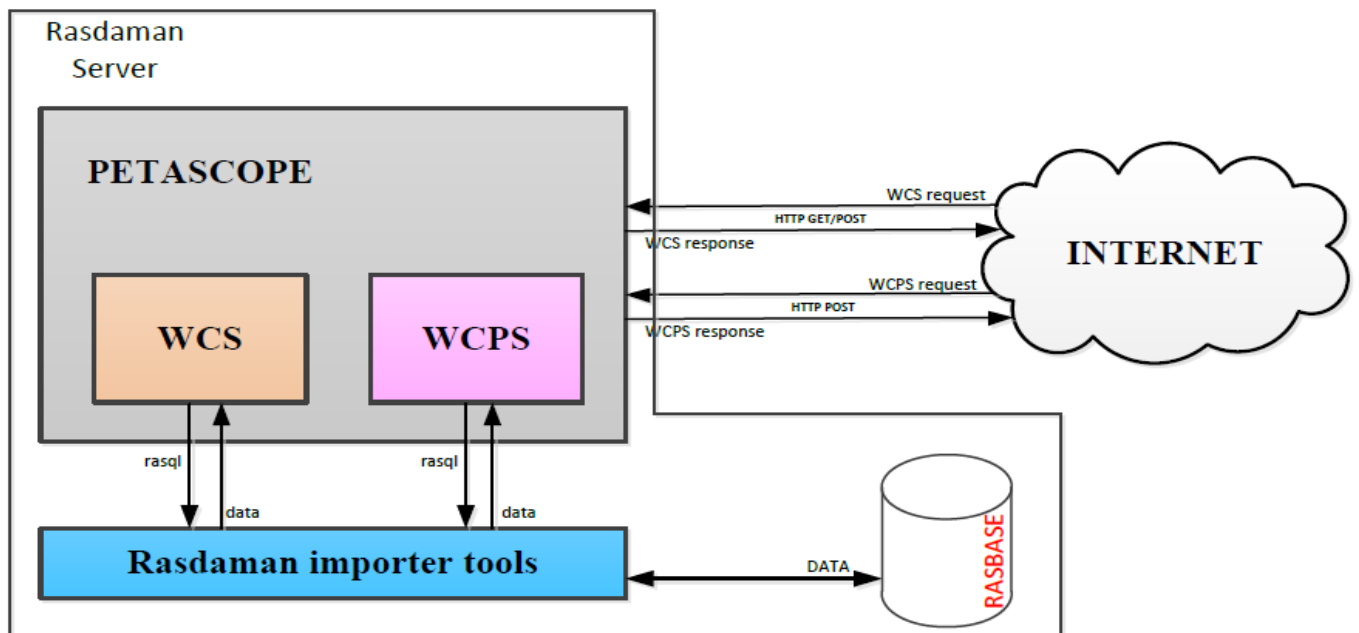


Figure 6.1. Petascope Overall Architecture (simplified from Campalani et al. (2014)).

Therefore, the OGC elements located into the Petascope lets Petascope act as an OGC server between Rasdaman and the OGC clients. In this study, Petascope used its WCS element to implement the WCS requests.

6.3 THREDDS

Thematic Realtime Environmental Distributed Data Services (THREDDS) is an open-source web server designed to make scientific data and metadata available and accessible for users. The aim of THREDDS is to create a dataset in many different formats. Then, it locates and presents this dataset in many different geographic locations accessible for users. One of the significant features of THREDDS is that it can use many protocols including WCS, WMS, HTTP and OPeNDAP (John Caron & Davis, 2006).

Java and Tomcat Web Server application are the system requirements for THREDDS installation. Java has to be installed because the THREDDS Server is executed in 100% free Java software and Tomcat should be installed since THREDDS has to be run as a Tomcat Web Server application (figure 6.2).

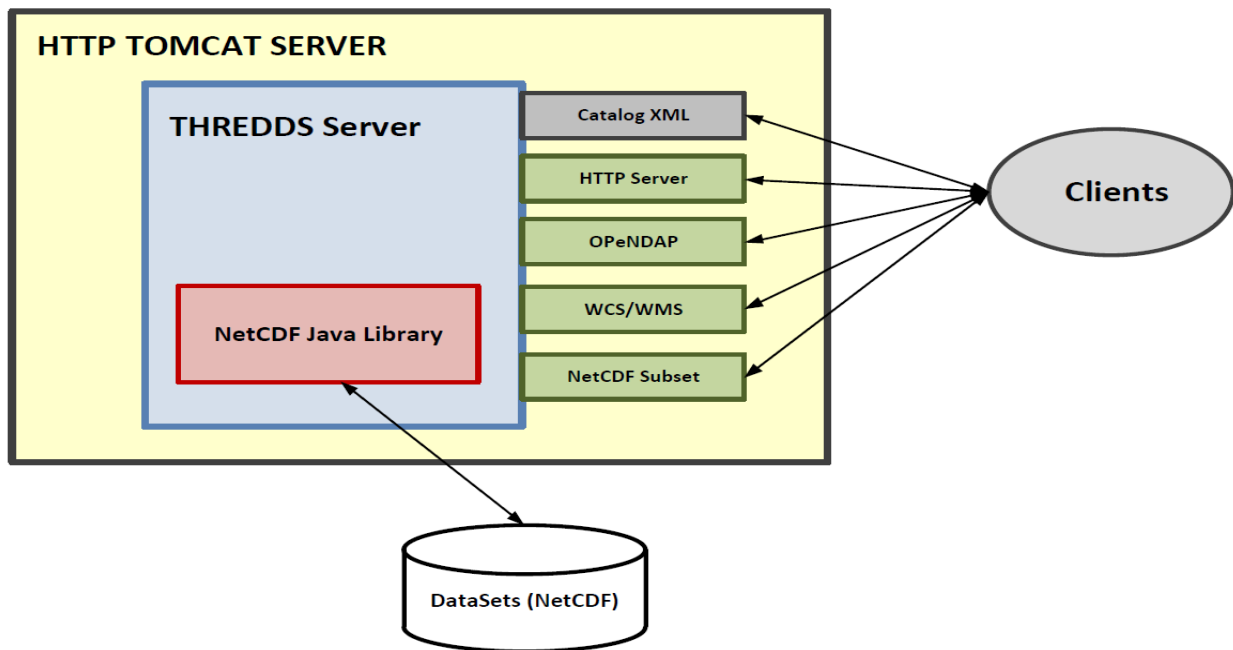


Figure 6.2. THREDDS Data Server Architecture (Inspired from Unidata (2015)).

As shown in above figure, THREDDS Server links to the dataset by configuring the dataset through its NetCDF-Java library. This library creates the metadata of the desired dataset in a

Catalog XML document. Clients connect and send a request to the THREDDS server by choosing one of the created protocols. These protocols ship the request to the NetCDF Java library connected to the datasets. Then the response is backed to the client by the chosen protocol.

6.4 Applications of Petascope and THREDDS

The main aim of generating Petascope was retrieving and processing the NASA's data located in the Rasdaman. In some NASA projects, Rasdaman is used as their geospatial raster database and Petascope as their middleware to their OGC protocols. In addition, Petascope is playing an important role in the EarthServer (European Scalable Earth Science Service Environment) project to make an open access to massive Earth Science data, based on the OGC geo service standards WCS and WCPS (Rasdaman, 2015a). So, Petascope's high ability in working with OGC services is resulted in gathering a lot of attentions from many remote sensing organizations.

In the new project of the NASA (the Global Climate Change application described in section 2.2), THREDDS is used as data service. It gives this ability to this project's users to download the daily-minimum temperature, daily-maximum temperature and precipitation data from 1950 to 2100 (NCCS-THREDDS, 2015).

6.5 Selection of WCS server implementations

Petascope is the natural choice of WCS server if the geospatial raster data are stored in Rasdaman. In addition, Petascope can support many different data formats like NetCDF and GeoTIFF formats.

On the other hand, THREDDS is another WCS server chosen to be used in this work. Today much of the environmental data obtained from remote sensing are stored in the NetCDF advanced file-based system. Therefore as THREDDS is based on the NetCDF Java library, it can support the NetCDF system and is a popular server for sharing environmental data. So, if data are stored in the NetCDF system, THREDDS is a suitable server to share the data. In this regard, since in the second approach of this study the NetCDF is chosen as the data storage technique, THREDDS is the best option to serve the data. In addition, THREDDS is chosen due to its high ability to serve the large number of data to the client side and support many services specifically WCS service.

7. Tools for WCS Client

7.1 Candidates of WCS client implementations

Two candidates for WCS client were selected: OpenLayers and QGIS. The reason of choosing these two clients was their good ability to support the WCS service. But, it was unknown before this study whether these products of software have the ability to retrieve the data in both NetCDF and GeoTIFF formats. In this regard, we have made an evaluation on these two clients before selecting any of them as the client in the case study.

7.2 OpenLayers

7.2.1. Background

In the past few years, there has been an increasing popularity of interactive web maps. In the past, creating interactive web maps required experts or large companies to provide it. But now, by the invention of free services and tools, anyone can easily design a web map with little knowledge about geography or cartography. OpenLayers is one such tool designed in 2005 by MetaCarta, a private company in USA, to make it easy to design a fast, accurate and interactive web map applications (Hazzard, 2011). OpenLayers is an open-source library which is aimed to create interactive web maps to be viewable in any web browser. It can be served by almost all browsers such as Google Chrome, Firefox, Safari or Opera.

OpenLayers is written in JavaScript to provide the varied range of user interface modules and supports the functions for some common tasks like animations, Ajax handling and etc. (Han et al., 2012). Also, it is applied to improve an Ajax (Asynchronous JavaScript and XML) internet application. Ajax is a technique that can create a better and more effective interactive web map applications (Han et al., 2012). In addition, OpenLayers library is used for dealing with different cartographic services like google maps, Open Street Maps.

Furthermore, OpenLayers offers an API to design powerful web-based geographical applications similar to Google Maps and Bing Maps. This API has two important features which are essential in order to design a web-based map: ‘*Map*’ and ‘*Layer*’. An OpenLayers ‘*Map*’ provides the information about the projection, units, extents and other additional information about the map. The data are shown as the ‘*Layer*’ inside the map. A ‘*Layer*’ renders the information about how OpenLayers should request the data and display them.

Additionally, with the aim of designing a rich web-based map application, OpenLayers supports Geography Markup Language (GML), Keyhole Markup Language (KML), GeoJSON and OGC service standards as WMS, WFS, WCS and WPS (Hazzard, 2011). Figure 7.1 shows how OpenLayers renders the Maps and Layers in working with OGC standards as WCS.

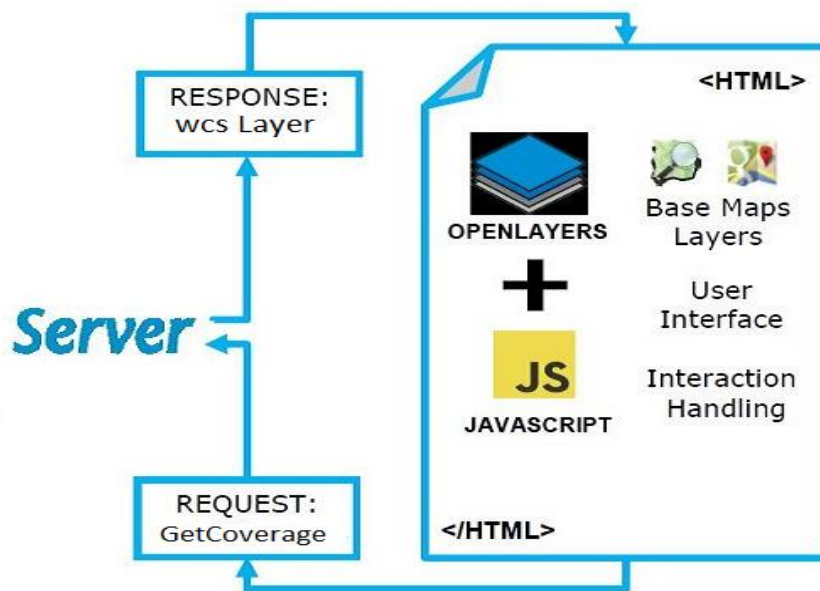


Figure 7.1. General Workflow of OpenLayers (simplified from geos.ed.ac.uk (2015)).

As shown in figure 7.1, OpenLayers improves its abilities with JavaScript language locate in an HTML file. The client generates the request through OpenLayers tools on the web and posts it to a web coverage server using HTTP. The web coverage server then executes the request and sends back to the OpenLayers.

7.2.2. Evaluation of OpenLayers

First, it was really important to know that JavaScript/OpenLayers is compatible to work with WCS standard or not. According to author experiments, JavaScript/OpenLayers is compatible with WCS service and they are suitable for creating a WCS client. Second, JavaScript/OpenLayers provides us to display the data in different formats (GeoTIFF and NetCDF) based on the server ability.

7.3 QGIS

7.3.1. Background

In 2002, Gary Sharman, designed a quick geographic raster and vector data viewer software. By that time, all of the geographic data viewer software were founded to be run on one operating system. His software, QGIS, has been designed to be run on many operating systems including Linux and windows. QGIS consists of around 94,000 code lines written in C++ under the conditions of the GPL license. It can support many geospatial raster and vector format including NetCDF, GeoTiff, PNG, JPEG, Shapefile and many others (Hugentobler, 2008).

QGIS has four main general layers which build the fundamental structure of it. The first and basic layer is called “datasource access”. For the geospatial raster data, this layer is responsible to convert images into the map layers. “Map layer” or “Map rendering” layer is the second fundamental layer of QGIS. This layer arranges and organizes the layer of the map, then renders the map to the next layer. The third layer is named geographical user interface (GUI). It plays its role as a glue to translate the user request for the map layer then pass the logic from the map layer to the user. Finally, the last layer consists of a set of plugins. This layer interacts with the user. It means plugins response to the mouse events on the map window and may add or remove buttons and menus from the main window (Hugentobler, 2008).

QGIS can be assigned as a WCS client. There is a WCS handler in QGIS to manage all of the requests. When a request is sent from the client, the WCS server located in QGIS acts upon that request. Finally, the request result is sent back to the client side in a geospatial raster format.

7.3.2. Evaluation of QGIS

An evaluation was performed on the QGIS client in connection to the THREDDDS server. This connection was done through WCS. The aim of this evaluation was estimating the ability of the QGIS to retrieve the data in both NetCDF and GeoTIFF formats (requirement of the client). First we tested QGIS to retrieve a GeoTIFF file from the dataset. In this order, a *GetCoverage* request was designed based on the located data on the THREDDDS. This request URL was inserted into the WCS URL of the QGIS. Figure 7.2 shows the data in the GeoTIFF format extracted from *GetCoverage* request in QGIS.

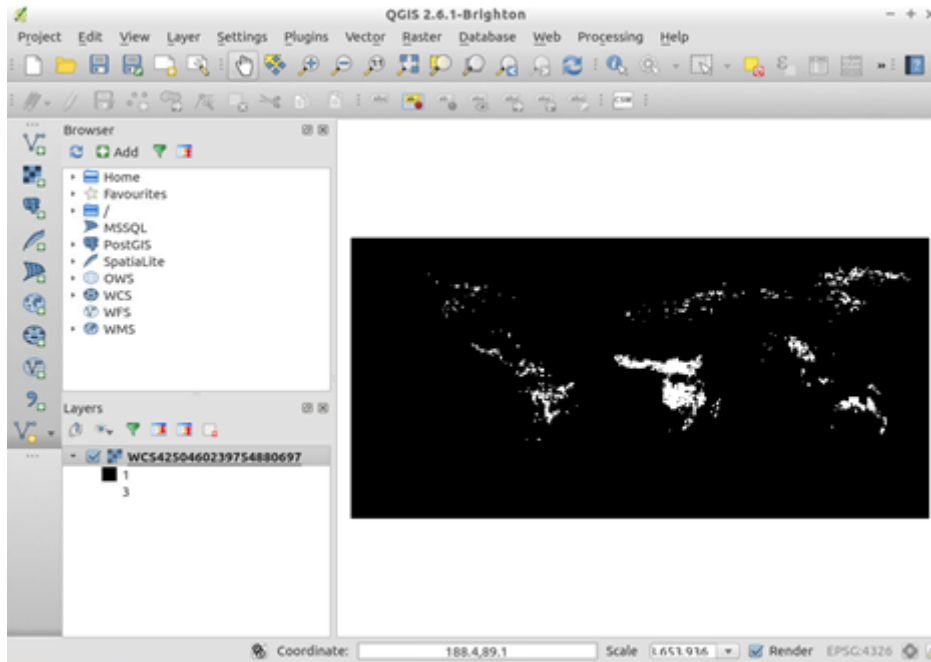


Figure 7.2. Extracted GeoTIFF file from GetCoverage request in QGIS client.

The same process was done to retrieve the data in the NetCDF format. In this step, QGIS was not successful to render the data in the NetCDF format. Even if QGIS is stated to support NetCDF, we faced compatibility problems.

7.4 Selection of WCS client implementations

Based on the previous evaluation, QGIS did not provide the client requirement of this project; it could not render the considered data in NetCDF format. So, it was not chosen as the client.

OpenLayers is chosen as the client library for this project. According to the previous evaluation of OpenLayers, this library is very powerful in creating web mapping applications. It is really user friendly and easy-to-use library. It provides great tools and functions to improve the web map services. OpenLayers gives the ability to the users to design a web-based map application and convert every aspect of the map-layers, controls, events, etc. Furthermore, this library is compatible and supportive with WCS service. JavaScript/OpenLayers are good for creating a WCS client. By using this client we can download a subset of the data using the WCS syntax. Then it is straight forward to evaluate the downloaded data using other tools.

8. Case Study

8.1 Introduction

This case study is an interoperability experiment in order to reveal the benefits of enhancing Petascope and THREDDS's middleware by implementing a WCS interface. Such aim was mainly implemented to answer the research objectives stated in section 1.3:

- Comparison of WCS servers created by Petascope and THREDDS.
- Evaluation of OpenLayers as a library to create WCS clients.

Figure 8.1 shows the system architecture implemented in the case study. The case study data were stored in Rasdaman and NetCDF advanced file system and then loaded on the Petascope and THREDDS. WCS requests related to the loaded data were evaluated by using OpenLayers. The connection between the servers and OpenLayers was JavaScript language over HTTP. By testing these approaches, the ability of WCS standard in downloading and retrieving the subset of the data in NetCDF and GeoTIFF formats were evaluated. Moreover, the performance of Petascope and THREDDS in supporting the WCS standard and NetCDF format were experimented in this study. In addition, the capabilities of OpenLayers in relation to WCS and usage of the data in both NetCDF and GeoTIFF formats were evaluated.

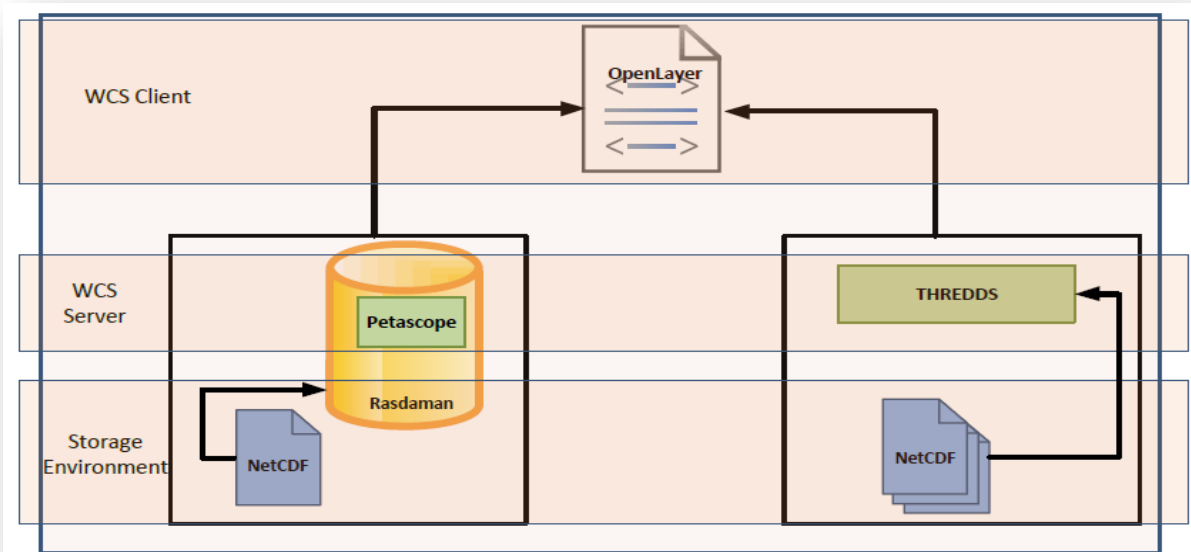


Figure 8.1. A general overview of the techniques used in this study.

8.2 Datasets

As the ICOS data are still not available, a similar dataset to the ICOS data taken from NASA Earth Exchange, NASA Ames Research Center was used in this work (NASA-EarthObservation-Download, 2015). These data are the result of NASA Global Climate Change application explained in section 2.2. The data file was named ‘NASA_tasmax_day’ in the NetCDF format as shown in the THREDDS website (<http://thredds.icos-cp.eu/thredds>). These data contained the variables and information about the daily maximum air temperature near the earth surface. They were predicted for 365 days of year 2100 in Kelvin unit.

‘NASA_tasmax’ data had three dimensions (latitude, longitude and date) and they were in Float32 format. As shown in the following the latitude and longitude of this variable has covered entire of the world:

```
netcdf tasmax_day_BCSD_rcp85_rlilpl_ACCESS1-0_2100{  
dimensions:  
  date = 365 ; // (365currently)  
  latitude = 720 ;  
  longitude = 1440 ;
```

8.3 Methods

The practical part of this study was formed in two case studies: Rasdaman/Petascop and NetCDF/THREDDS. Each of these two case studies was divided into three parts: Implementation of the server side, evaluation of WCS server and implementation of the client. In the first case study, as shown in figure 8.2, the data were imported to the Rasdaman through Petascop component. Then the WCS requests were extracted according to the imported data. Openlayers was connected to the created WCS requests by using JavaScript codes. At the end, the map created for the data were loaded on the web browser to let users download the data.

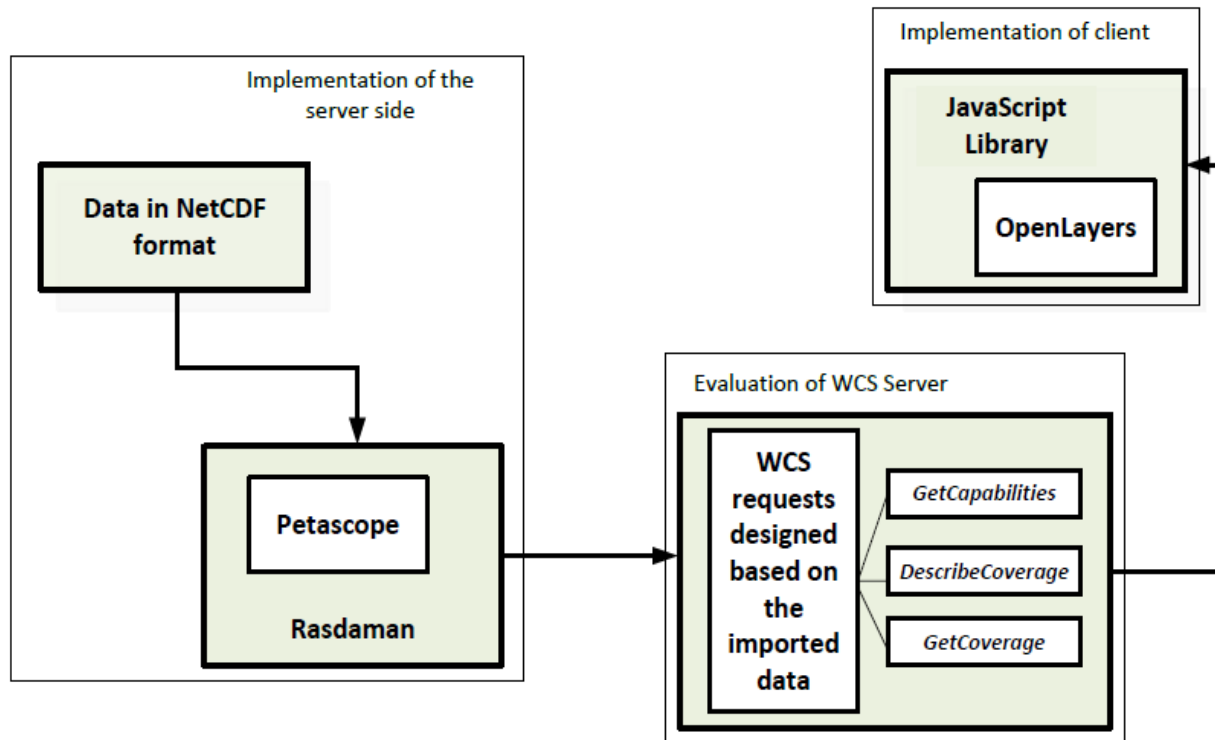


Figure 8.2. An overview of the first case study technique (Rasdaman/Petascope).

In the second case study, as shown in figure 8.3, the first step was the implementation on the server. In this step, the data were located in NetCDF advanced file-based system and configured in the THREDDS data server (Appendix D). Then the WCS requests were designed according to the configured data in THREDDS. Finally, the same processes were done as the first case study to connect the WCS request to the OpenLayers.

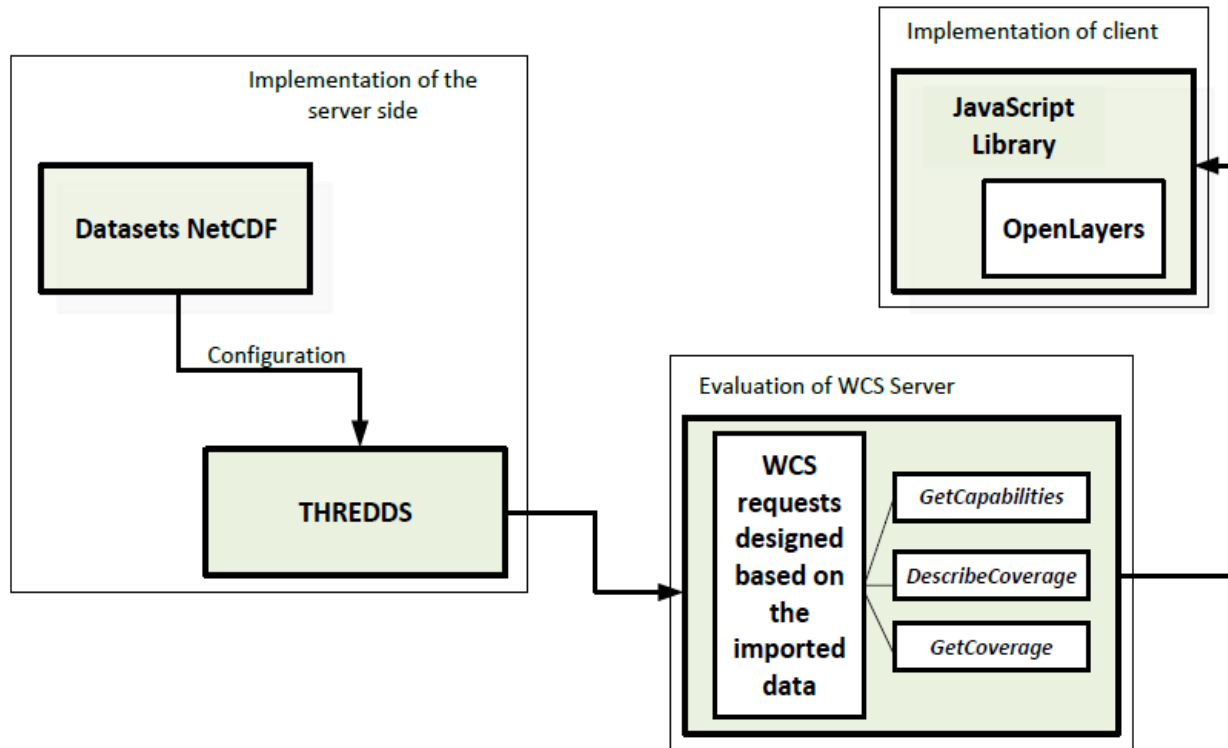


Figure 8.3. An overview of the second case study technique (NetCDF/THREDDS).

8.4 Case study 1-Rasdaman and Petascope

In figure 8.2, the three sections of this case study are shown. They are described in detail below.

8.4.1. Implementation of Server

There were two ways to install Rasdaman: Installing Rasdaman or using Virtual Machine like OSGeo-live. In this study, Rasdaman 9.0.0 was installed on the Linux environment. The installation procedure is described in Rasdaman website (Rasdaman, 2015d). After installation, Rasdaman is run from a text interface (figure 8.4).

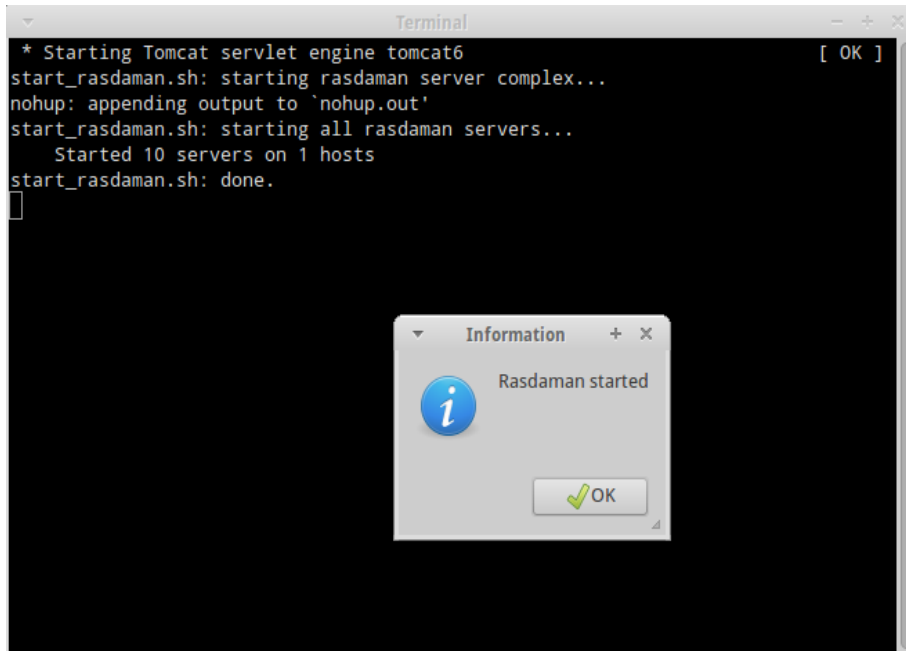


Figure 8.4. Rasdaman interface.

Two required software were installed as the prerequisite of Petascope installation: PostgreSQL 9.3 and Tomcat 6.0. Petascope as the server side of this technique was implemented as a war file to give an access to coverages stored in Rasdaman. For installation, this war file was deployed in Tomcat and then configured into the Rasdaman (Petascope.User.Guide, 2015).

In order to use Petascope, two additional Rasdaman components were necessary to be configured: **Rasgeo** and **SCORE**. Rasgeo component was configured in order to ease the insertion of geospatial raster data into the Rasdaman through Rasimport utility. By using Rasimport, the Rasdaman database was connected to the Petascope. SCORE was introduced in order to let Petascope know all the coordinate system spaces in which the coverages are defined. It was done by:

```
$ export SCORE= 'http://localhost:8080/def'
```

\$ wget 'http://localhost:8080/def/crs/OGC/0/Index3D' // A Cartesian coordinate system is defined based on the data dimensions. Users can choose any coordinate system they want. For example if a user define a coordinate system like 'http://localhost:8080/def/crs/EPSG/0/4326', then a WGS-84 CRS would be defined for the data.

(\$ sign means that the command was run in the Linux Terminal)

After all explained installations and configurations, the data insertion steps were started. First, a collection was created as a name of 'ICOSData' based on the data type. It was done in the way that explained in section 4.2.6:

```
$ Rasql -q "create collection ICOSData FloatSet3" -user rasadmin -passwd xxx
```

Then the data in the NetCDF format were imported into this collection by Rasimport tool. By this way, the data were inserted to Rasdaman database through Petascope middleware based on the data information. The below codes are showing how the data were exported and located in Rasdaman:

```
$ export tasmax_day_BCSD_rcp85_r1i1p1_ACCESS1-0_2100.nc  
$ rasimport -f NETCDF:"tasmax_day_BCSD_rcp85_r1i1p1_ACCESS1-0_2100.nc":tasmax --coll  
'ICOSData' --coverage-name 'tasmax' --crs-uri 'http://localhost:8080/def/crs/OGC/0/Index3D'
```

- -f →defines that a single image file was imported into the target collection
- --coll →defines the name of the target collection
- --coverage-name →defines the name for the imported file to be exposed as WCS coverage
- --crs-uri →defines the coordinate reference system identifier

- Note that Tomcat and PostgreSQL should be run before using Petascope, otherwise it does not work.
- Note that all the Rasdaman geospatial raster data are not available through the Petascope; the required geospatial raster data should be registered through the Petascope administration interface.

By this way, Petascope translated all the requests requiring the evaluation coverage into the Rasdaman.

8.4.2. First Evaluation of Petascope WCS Server

Before designing WCS operations, the WCS URL was created in Rasdaman. This URL was created according to this structure: *http://your.server/rasdaman/ows/wcs*.

The below command is shown how this URL was created:

```
$ export WCS2_ENDPOINT=' http://localhost:8080/rasdaman/ows/wcs '
```

Then the three WCS operations were designed based on the imported data in Rasdaman: *GetCapabilities*, *DescribeCoverage* and *GetCoverage*. They were aimed to evaluate how Petascope WCS server is providing user requirements.

– *GetCapabilities*

In the first step, a *GetCapabilities* request was designed from WCS server to determine capabilities of WCS in executing and providing the coverage.

Request:

```
http://localhost:8080/rasdaman/ows/wcs?service=WCS&version=2.0 &request=GetCapabilities
```

The main component of this request, “request=*GetCapabilities*”, shows the capabilities of WCS service and response in the form of XML data. The table 8.1 specifies the meaning of all parameters used in this request.

Table 8.1. URL parameters for *GetCapabilities* request.

Parameters	Parameter Values	Description
Service	WCS	The type of service that was requested.
Version	2.0	The WCS protocol version used in this request was 2.0
Request	GetCapabilities	Since the request was designed to show the capabilities of WCS, the request was specified as <i>GetCapabilities</i> .

In response to a *GetCapabilities* request, WCS server produced an XML document (Appendix A). This XML document served the information about WCS service, describing about the supported operations through WCS, and the information about the available coverage. Additionally, this document gave the information about WCS Service details including the name,

title and URLs. Also, it gave the information about WCS capabilities including *GetCapabilities*, *DescribeCoverage* and *GetCoverage* along with respective formats and URLs.

– *DescribeCoverage*

In this step, *DescribeCoverage* request was implemented based on WCS server’s capabilities information to provide the description about the supported coverage. Based on this step, the next request (*GetCoverage*) was used to access to the metadata associated with ‘tasmax’ data.

Request:

```
http://localhost:8080/rasdaman/ows/wcs?service=WCS&version=2.0.1&request=DescribeCoverage&Coverageid=tasmax
```

The main component of this request was “Coverageid=tasmax”. It defined that the response of this request was retrieved for the data with ‘tasmax’ name. The table 8.2 specifies the meaning of all of the parameters used in this request.

Table 8.2. URL parameters for *DescribeCoverage* request.

Parameters	Parameter Values	Description
Service	WCS	The type of service that was requested.
Version	2.0.1	The WCS protocol version used in this request was 2.0.1.
Request	DescribeCoverage	The name of the request; used to retrieve the information about the specific coverage
Coverageid	tasmax	The name of the coverage which was described

In response to this *DescribeCoverage* request, a GML document has been returned which contained the information about the coverage. Appendix B shows this coverage response.

– *GetCoverage*

In order to take the description of supported *GetCoverage*, the following requests were executed. According to Appendix B (response of the *DescribeCoverage* request), the coverage supported ‘image/TIFF’ and ‘application/NetCDF’ formats. As one of the main aims of this project was to retrieve the data in both GeoTIFF and NetCDF formats, then the *GetCoverage* request was implemented for both of these formats. In addition, retrieving the subset of data by changing the time dimension was another important aim of this project. So, ‘time’ was introduced in the *GetCoverage* request.

Request for NetCDF format:

```
http://localhost:8080/rasdaman/ows/wcs?service=WCS&version=2.0.1&request=GetCoverage&Coverageid=tasmax&format=image/TIFF&&Subset=Longitude(10,20)&Subset=Latitude(10,20)&Subset=Time(10)
```

Request for GeoTiff format:

```
http://localhost:8080/rasdaman/ows/wcs?service=WCS&version=2.0.1&request=GetCoverage&Coverageid=tasmax&format=application/NetCDF&Subset=Longitude(10,20)&Subset=Latitude(10,20)&Subset=Time(10)
```

Table 8.3 describes the definition of each parameters used in this request.

Table 8.3. URL parameters used in *GetCoverage* request.

Parameters	Parameter Values	Description
Service	WCS	The type of service that was requested.
Version	2.0.1	The WCS protocol version used in this request was 2.0.1.
Request	GetCoverage	The name of the request

Coverageid	tasmax	The name of the coverage to be described
Format	Application/NetCDF & image/TIFF	The format in which the coverage response was returned (only one value should be defined for each request)
Subset=Longitude()	(10,20)	It shows the minimum and maximum longitude of the corners. (subset of space)
Subset=Latitude()	(10,20)	It shows the minimum and maximum latitude of the corners. (subset of space)
Subset=Time()	(10)	It shows that only the data for these specific days were returned. (subset of time)

In response to above *GetCoverage* requests, a GeoTIFF and NetCDF file were obtained as well as the information about the subset of the data. It shows that this request worked correctly. Otherwise, an error would be shown in an XML document indicating that this request was not working.

8.4.3. Implementation of Client

In order to access and display the data, OpenLayers3 (OpenLayers, 2015) was chosen. The JavaScript codes in HTML were written in the IntelliJ IDEA editor (Jetbrains, 2015) and they were tested by Google Chrome web browser. The code is provided in Appendix C. Then the ‘Jsfiddle’ environment was applied to make the created map available on the internet for all users (jsfiddle, 2015). (Jsfiddle is an online environment that let code designers test their codes quickly and share them with others).

Figure 8.5 shows the interface to the map client. Two options were created with check boxes to allow users to define the format of the received data: ‘Download GeoTIFF’ and ‘Download NetCDF’. Then a ‘DragBox’ interaction used to select features by drawing boxes and retrieve the data in subset of space. Additionally, a combo box as a name of ‘Date’ was created to let

users select the data in their desired time. Overall, this client interface renders users a direct, interactive, and responsive way to download the data in both NetCDF and GeoTIFF formats in a specific time (Figure 8.5).

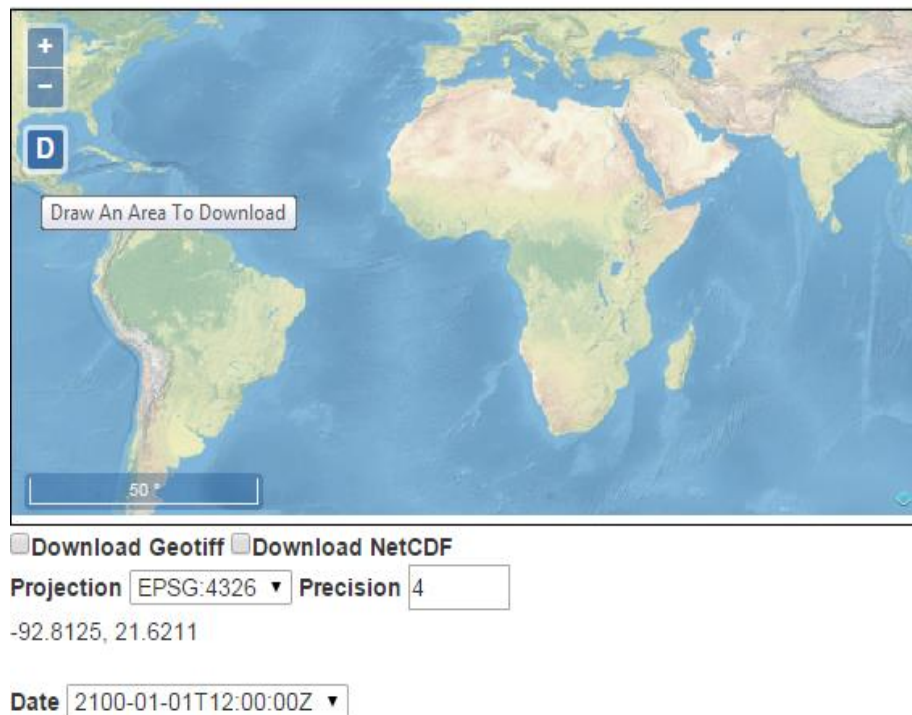


Figure 8.5. The designed map by OpenLayers client on the jsfiddle cloud.

8.4.4. Result

Figure 8.6 shows the retrieved data in the GeoTIFF format. In this figure, by dragging a box, the user received the data for the dragged area. Moreover, the data for the specific time could be defined through the 'Date' combo box as shown on the figure. All the information about the downloaded data was rendered in the green box under the map.

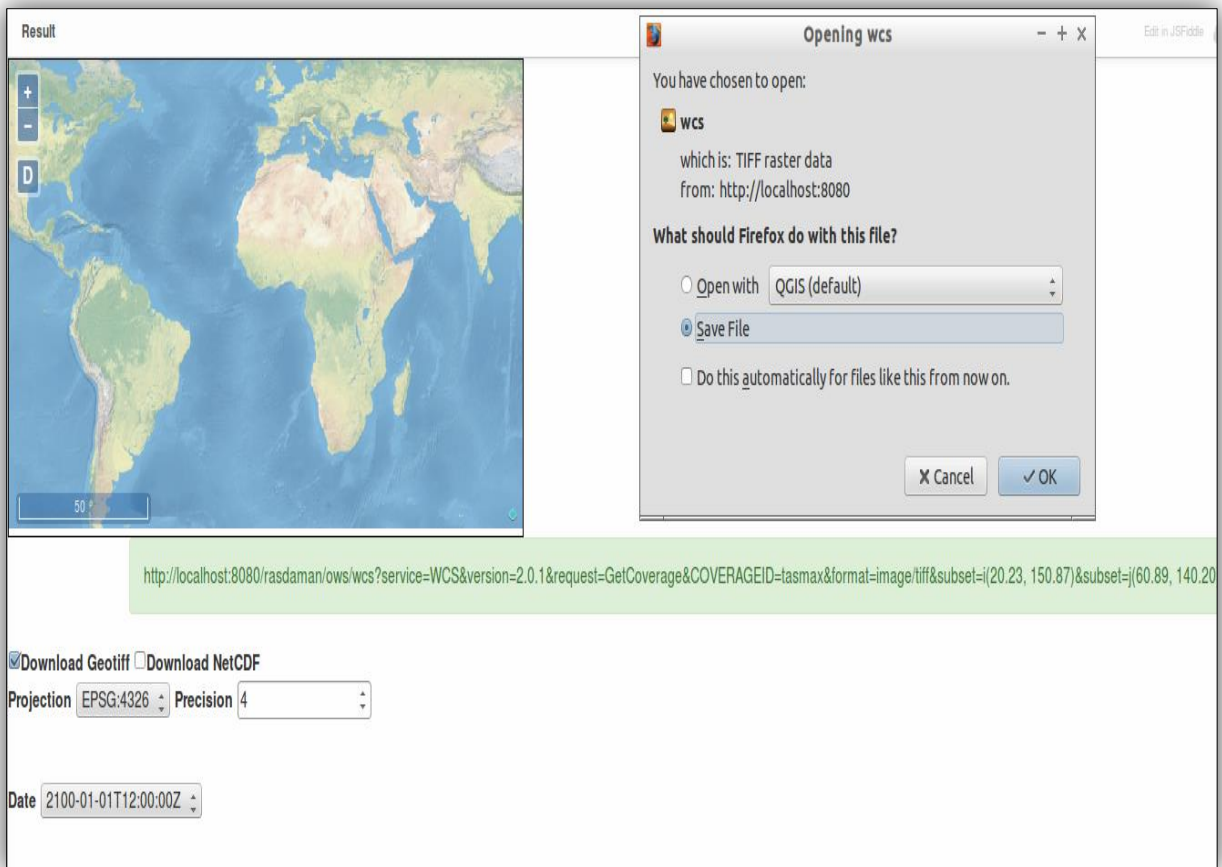


Figure 8.6. The map designed in OpenLayers in connection to Rasdaman/Petascop technique. By selecting a specific area and specific time, the data were downloaded and shown in GeoTIFF format.

Figure 8.7 shows the downloaded data in the NetCDF format. Same as figure 8.6, the data were retrieved for the dragged area and defined time. The information about the request was presented in a green box under the map.

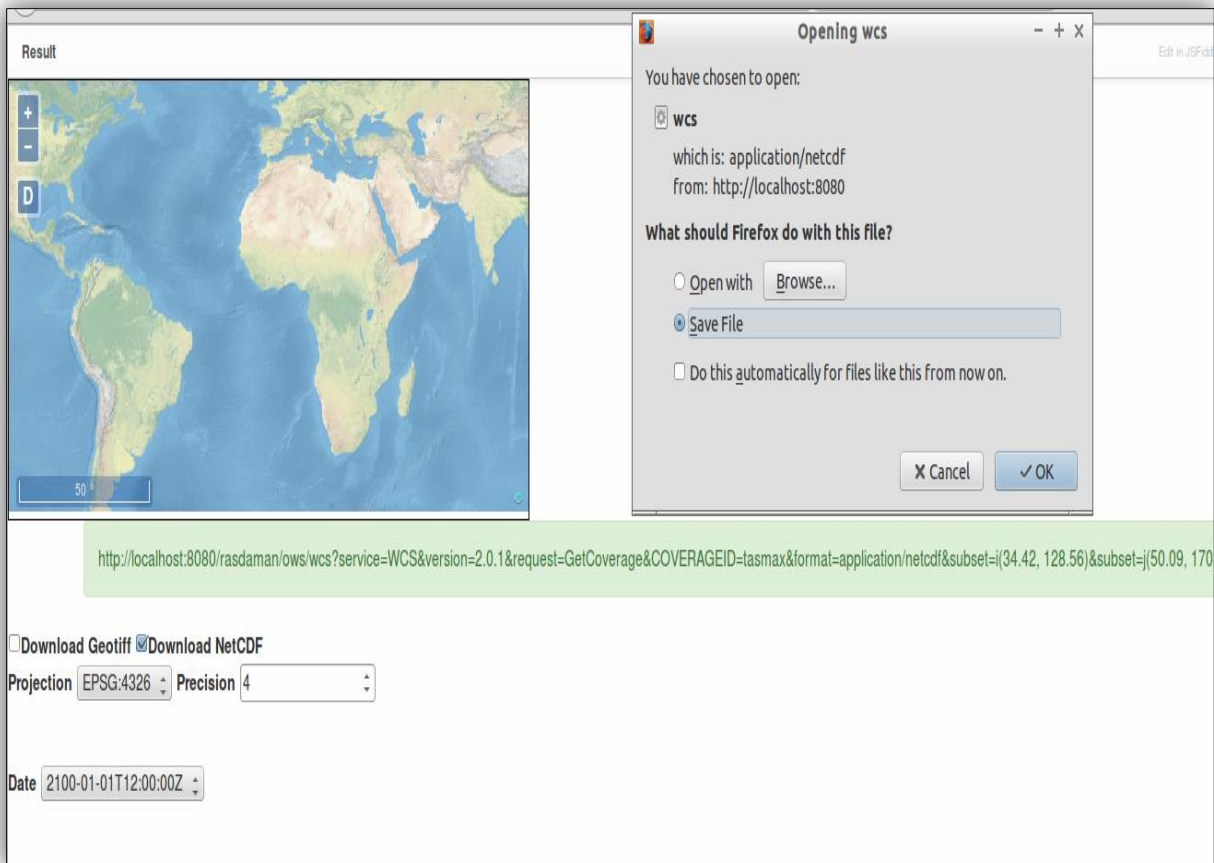


Figure 8.7. The map designed in OpenLayers in connection to Rasdaman/Petascop technique. By selecting a specific area and specific time, the data were downloaded and shown in NetCDF format.

8.5 Case study 2- NetCDF and THREDDS

Like the previous case study, this section also was divided into three main parts: Server, standard and client parts. These three sections are described respectively.

8.5.1. Implementation of Server

The first step of the server part was installation of Java and Tomcat on the Linux environment in order to install THREDDS. These installations were done by Oleg Mirzov (System Architecture at the ICOS CP, Lund University) and Roger Groth (Programmer at the ICOS CP, Lund University). After the successful installation of Tomcat 6.0 and Java 1.6 on Linux Environment (in this case Centos 7), THREDDS 4.6.0 war file was downloaded and deployed into Tomcat.

The reason that THREDDS was inserted into the WAR file was because it made it possible to have an easy installation into Tomcat. To be sure that THREDDS was installed successfully, it was run as a Tomcat Web server application by locating the own local host address in browser as shown in figure 8.8.

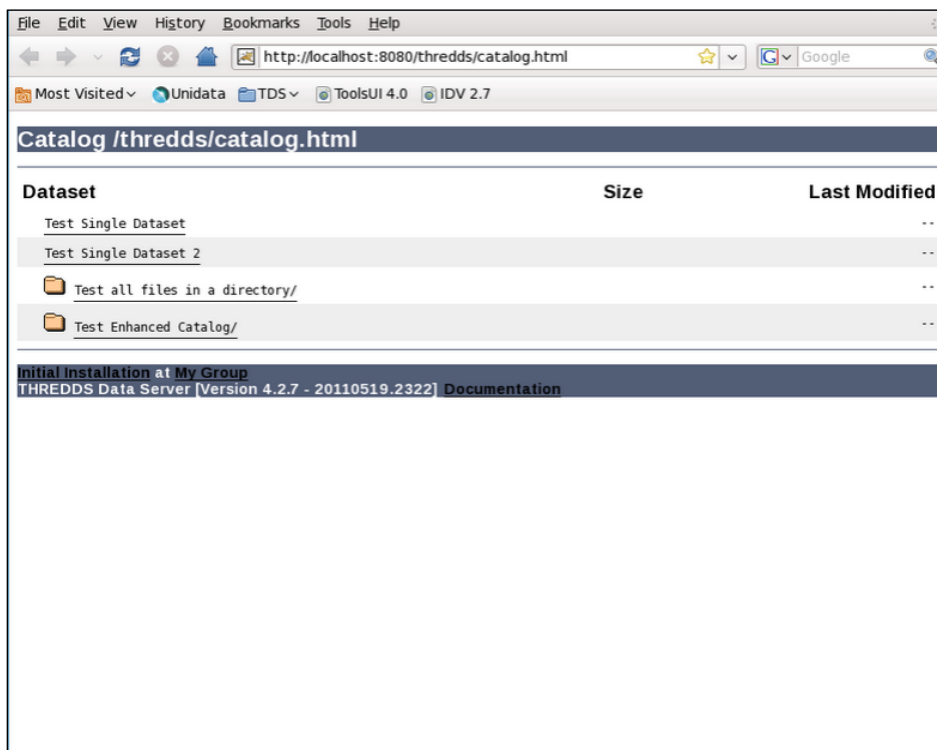


Figure 8.8. The THREDDS environment deployed on Tomcat application.

THREDDS threddsConfig.xml configuration file was used in order to collect, organize and describe desired datasets. The WCS URL was introduced to the THREDDS catalogue file as well. In THREDDS catalog, further information and parameters about the dataset were added. These parameters were included of the name of service (WCS), name of data and data URL path. Appendix D shows the environment of THREDDS configuration catalog file, how the URL for WCS protocol was designed and how the data were loaded on THREDDS configuration file.

As shown in figure 8.9, all the data located in the configuration file of THREDDS were loaded on it.

The screenshot shows a web browser window with the URL `http://thredds.icos-cp.eu/thredds/catalog.html`. The page title is "Catalog http://thredds.icos-cp.eu/thredds/catalog.html". Below the title is a table with three columns: "Dataset", "Size", and "Last Modified". The table contains the following entries:

Dataset	Size	Last Modified
Yearly fluxes		--
CO2 EUROPE LSCE		--
Tair_daily_WFDEI_201211		--
NASA_tasmax_day_BCSM_rcp85_r1i1p1_ACCESS1-0_2100		--

Below the table, there is a footer section with the text: "ICOS Carbon Portal THREDDS server at INES at Lund University see Info" and "THREDDS Data Server [Version 4.6.0 - 20150326.1318] Documentation".

Figure 8.9. The datasets loaded on THREDDS.

8.5.2. First evaluation of THREDDS WCS Server

Although WCS URL was introduced in the configuration file, it was not enabled. In order to implement the WCS specification in THREDDS server, WCS had to be enabled in the `threddsConfig.xml` configuration file. It was done by adding an 'allow' element into this file as shown in below:

```
<WCS>
<allow>true</allow>...
</WCS>
```

When WCS was enabled, datasets were configured to have a WCS access method in THREDDS catalog configuration files as described before. The WCS Dataset URLs was designed in a specific structure:

<http://servername/thredds/wcs/> → <http://thredds.icoscp.eu/thredds/wcs/>

Then in the following of this URL, the name of data which was configured in the catalog file was defined. This is typically the URL passed to WCS client:

```
http://thredds.icoscp.eu/thredds/wcs/common/netcdf/dataDemo/tasmax_day_BCSD_rcp85_r1i1p1_
ACCESS1-0_2100.nc
```

After designing the WCS URL for THREDDS based on the imported data, WCS requests were designed.

– *GetCapabilities*

In the first step, a *GetCapabilities* request was designed from WCS server (located in THREDDS) to determine the capabilities of WCS in executing and providing the coverage.

Request:

```
http://thredds.icos-
cp.eu/thredds/wcs/common/netcdf/dataDemo/tasmax_day_BCSD_rcp85_r1i1p1_ACCESS1-
0_2100.nc?service=WCS&version=1.0.0&request=GetCapabilities
```

As explained in the section 8.4.2, The XML document which was created in the response to *GetCapabilities* request contained the information about WCS Service details including the name, title and URLs (see Appendix A).

– *DescribeCoverage*

In this step, a *DescribeCoverage* request was formatted based on the WCS server's capabilities information to provide the description about the supported coverage.

Request:

```
http://thredds.icos-
cp.eu/thredds/wcs/common/netcdf/dataDemo/tasmax_day_BCSD_rcp85_r1i1p1_ACCESS1-
0_2100.nc?service=WCS&version=1.0.0&request=DescribeCoverage&COVERAGE=tasmax
```

In response to this *DescribeCoverage* request, a GML document was returned which contained the information about the coverage (see Appendix B). (The definitions of the parameters used in this request are described in section 8.4.2).

– *GetCoverage*

In order to take the description of supported *GetCoverage*, the following requests were executed. As what was done in the previous case study, this request was designed to retrieve the data in both GeoTIFF and NetCDF formats in a specific time subsets.

Request for GeoTIFF format:

```
http://thredds.icos-  
cp.eu/thredds/wcs/common/netcdf/dataDemo/tasmax_day_BCSD_rcp85_r1i1p1_ACCESS1-  
0_2100.nc?service=WCS&version=1.0.0&request=GetCoverage&format=GeoTIFF&coverage=tasm  
ax&bbox=10,10,12,12&time=2100-01-01T12:00:00Z/2100-12-31T12:00:00Z
```

Request for NetCDF3 format:

```
http://thredds.icos-  
cp.eu/thredds/wcs/common/netcdf/dataDemo/tasmax_day_BCSD_rcp85_r1i1p1_ACCESS1-  
0_2100.nc?service=WCS&version=1.0.0&request=GetCoverage&format=NetCDF3&coverage=tasm  
ax&bbox=10,10,12,12&time=2100-01-01T12:00:00Z/2100-12-31T12:00:00Z
```

(The definitions of the parameters used in these requests are described in section 8.4.2). The ‘bbox’ in this request shows ($Long_{min}$, Lat_{min} , $Long_{max}$, Lat_{min}) of corner. It means that it defines the portion of the coverage in space. ‘Time’ in this request shows the specific time that the data was retrieved.

In response to above *GetCoverage* requests, a GeoTIFF and NetCDF file as well as the information about the subset of data were obtained.

8.5.3. Implementation of Client

In the last side of this case study, client side, OpenLayers was used to display the data on the map in the web browser. The JavaScript codes in HTML were used to connect THREDDS to OpenLayers. All the procedures done in this step were as same as section 8.4.3 with some changes in JavaScript code.

8.5.4. Result

Figure 8.10 shows the retrieved data in the GeoTIFF format. As shown in this figure, ‘Download Geotiff’ button was checked and then by dragging a rectangle in a specific area through Dragbox button, the data was retrieved for that area. Furthermore, the data for a specific time was defined through the ‘Date’ combo box.

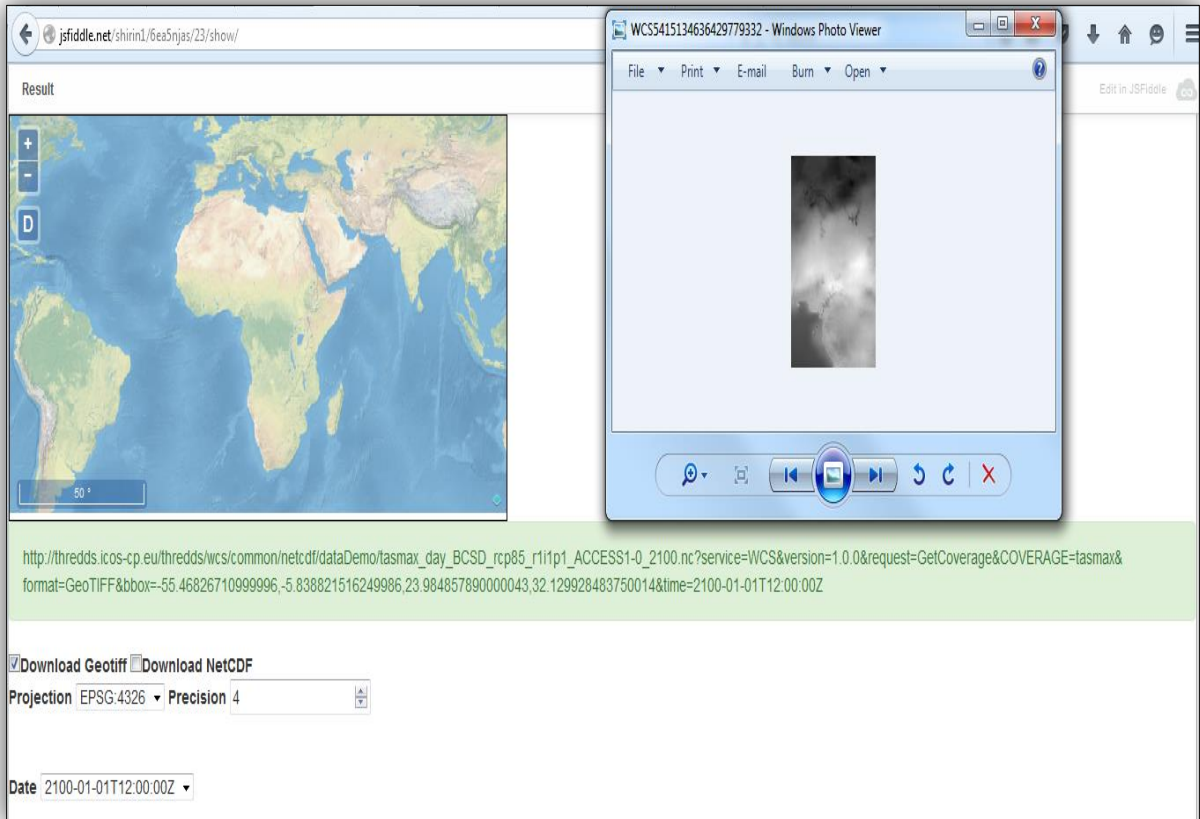


Figure 8.10. The map designed in OpenLayers in connection to NetCDF/THREDDS technique. By selecting a specific area and specific time, the data were downloaded and shown in GeoTIFF format.

Figure 8.11 shows the retrieved data in the NetCDF format. By dragging a box, the data in NetCDF format was retrieved for the dragged area. The time also could be specified through ‘Date’ combo box. The green box under the map gives all the information about the request.

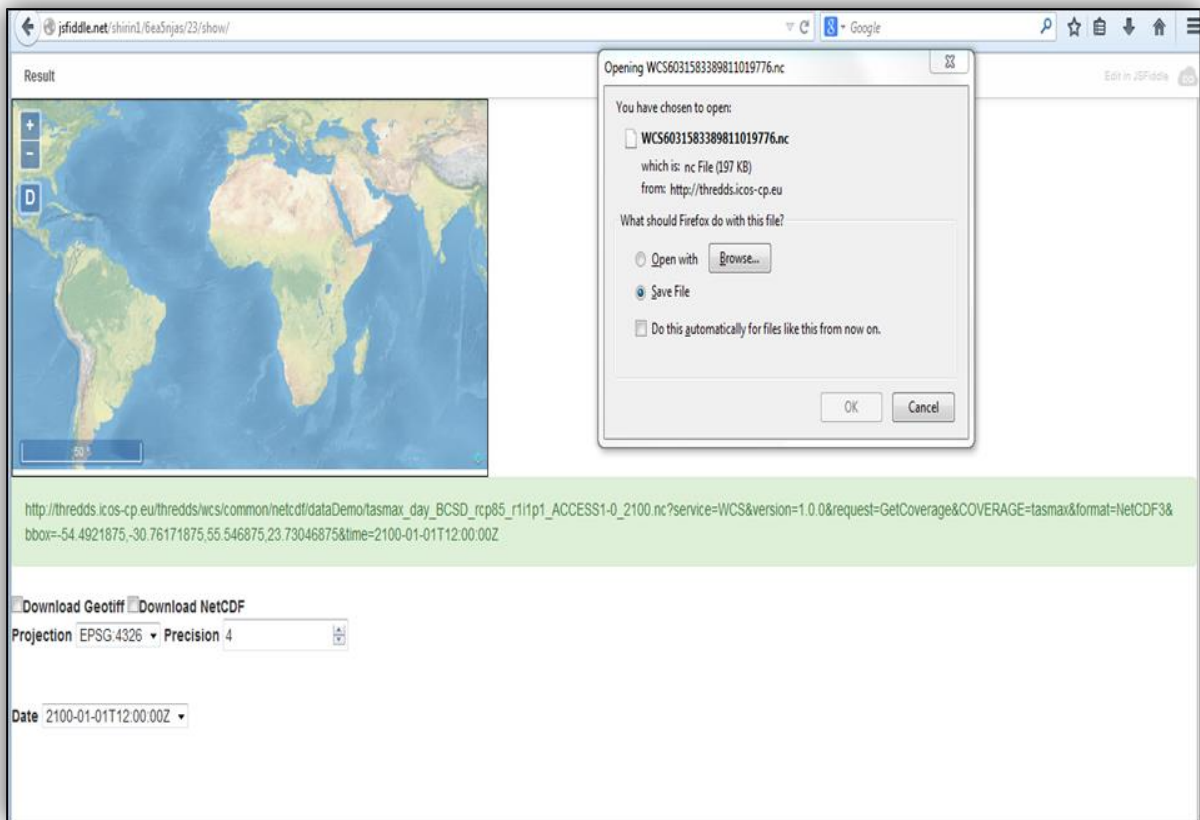


Figure 8.11. The map designed in OpenLayers in connection to NetCDF/THREDDS technique. By selecting a specific area and specific time, the data were downloaded and shown in NetCDF format.

9. Evaluation

As described in previous sections, it was possible to retrieve the data in both NetCDF and GeoTIFF formats by Rasdaman/Petascop and NetCDF/THREDDS techniques. It was done through OpenLayers client in connection with Petascop and THREDDS data server and WCS protocol. But the question is ‘Did the downloaded formats give the data for any desired space and time subsets correctly or not?’. To answer this question, two evaluations were done for both space and time subsets in GeoTIFF and NetCDF formats obtained from each technique.

9.1 Evaluation of case study 1- Rasdaman and Petascop

9.1.1. Evaluation of Retrieval of Subsets in Space

We retrieved the data for a specific subset. First, we evaluated the GeoTIFF data in space subset obtained from the Rasdaman/Petascop technique. These data were evaluated both visually and through their metadata. Then the NetCDF downloaded were evaluated in space subset both visually and through their metadata. At the end, it was checked that the numerical values of obtained data are the same with original data to prove that these obtained data are showing the original data in different subsets.

Figure 9.1 shows the GeoTIFF file visually which obtained with Longitude (112, 159) and Latitude (-54, -9) for Australia by Rasdaman and Petascop technique.

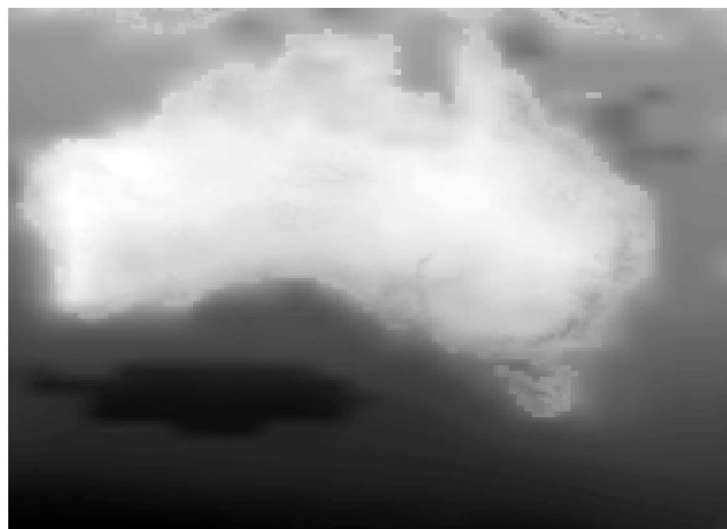
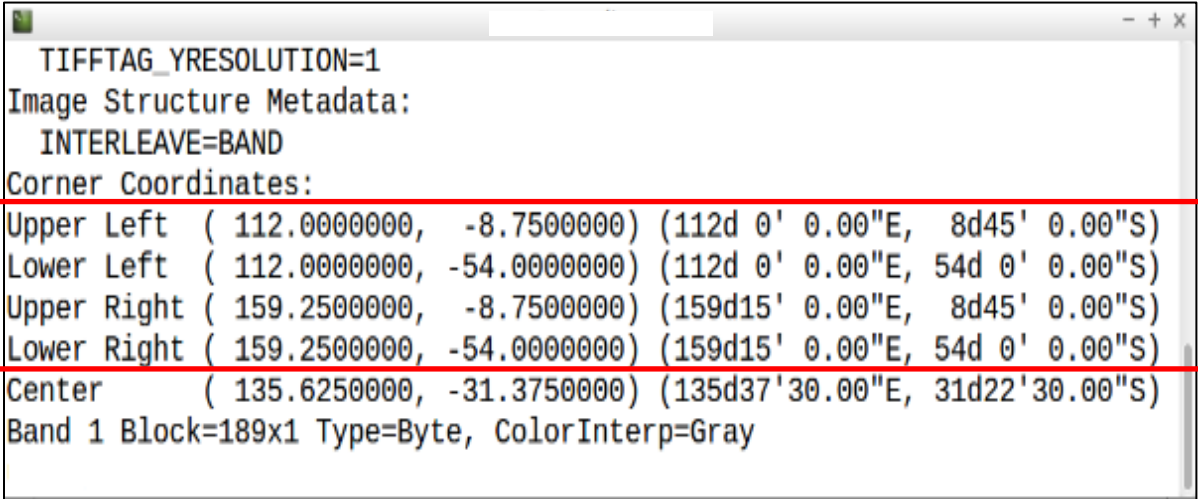


Figure 9.1. The subset of the data which are downloaded for specific space with Longitude ($x_{min}=112$, $x_{max}=159$) and Latitude ($y_{min}= -54$, $y_{max}= -9$) in GeoTIFF format through Rasdaman/Petascop technique.

The coordinate system used to retrieve these data was Geographic coordinate system (WGS-84/EPSSG-4326). Rasdaman server has this ability to let designers define any desired coordinate system before connecting to any client.

To prove that these data were retrieved correctly in their defined subsets, we have investigated their metadata. In this regard, the Linux Terminal Command was used to retrieve the information about the GeoTIFF downloaded data. Figure 9.2 shows the coordinates of the data which were retrieved for Longitude ($x_{min}=112$, $x_{max}=159$) and Latitude ($y_{min}= -54$, $y_{max}= -9$) subset for Australia.



```
TIFFTAG_YRESOLUTION=1
Image Structure Metadata:
  INTERLEAVE=BAND
Corner Coordinates:
Upper Left  ( 112.0000000, -8.7500000) (112d 0' 0.00"E,  8d45' 0.00"S)
Lower Left  ( 112.0000000, -54.0000000) (112d 0' 0.00"E,  54d 0' 0.00"S)
Upper Right ( 159.2500000, -8.7500000) (159d15' 0.00"E,  8d45' 0.00"S)
Lower Right ( 159.2500000, -54.0000000) (159d15' 0.00"E,  54d 0' 0.00"S)
Center      ( 135.6250000, -31.3750000) (135d37'30.00"E, 31d22'30.00"S)
Band 1 Block=189x1 Type=Byte, ColorInterp=Gray
```

Figure 9.2. The extracted information for GeoTIFF file obtained from Rasdaman/Petascopie technique to prove that these data belong to ($x_{min}=112$, $x_{max}=159$) and ($y_{min}= -54$, $y_{max}= -9$) subset.

After it was proved that the data in the GeoTIFF format show the correct space subsets, the data in the NetCDF format were investigated with the same space subsets as above for Australia. This NetCDF file format also was evaluated visually and through its metadata. For evaluating this NetCDF file visually, ‘ncview’ tool was used through Linux Terminal command as shown in figure 9.3 (ncview is a visual browser for viewing netCDF format file). In addition, to prove that these data were extracted in their defined subsets correctly, their metadata was extracted through Linux Terminal command. Figure 9.4 shows that the downloaded NetCDF data were retrieved correctly in their defined subsets.

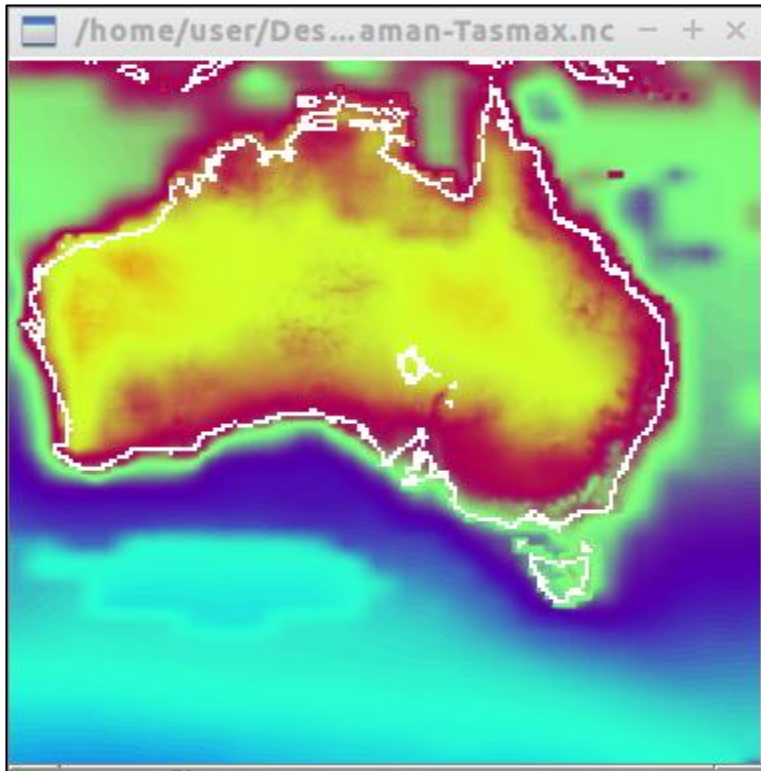


Figure 9.3. The NetCDF format obtained from Rasdaman/Petascopie technique. These data show the specific subset with Longitude (112, 159) and Latitude (-54, -9).

```

c; Translation Date = 2015-09-25T09:27:49.449Z" ;
      :geospatial_lat_min = -53.875 ;
      :geospatial_lat_max = -8.875 ;
      :geospatial_lon_min = 112.125 ;
      :geospatial_lon_max = 159.125 ;
}
user@osgeolive:~$

```

Figure 9.4. The extracted information for the NetCDF file obtained from Rasdaman/Petascopie technique with subsets in ($x_{min}=112$, $x_{max}=159$) and ($y_{min}=-54$, $y_{max}=-9$). It proves the extracted data belongs to this defined subsets.

Now, the question is how space subsets of data are selected from the original data. As shown in figure 9.5, black gridded lines show whole of the data. If a user select a subset of data by blue rectangle, a subset of data which are shown with red rectangle will be downloaded for the user. It

means that since the corners of the blue rectangle is selected on the middle of the cells, the red rectangle will be selected to send back whole of the corner cells.

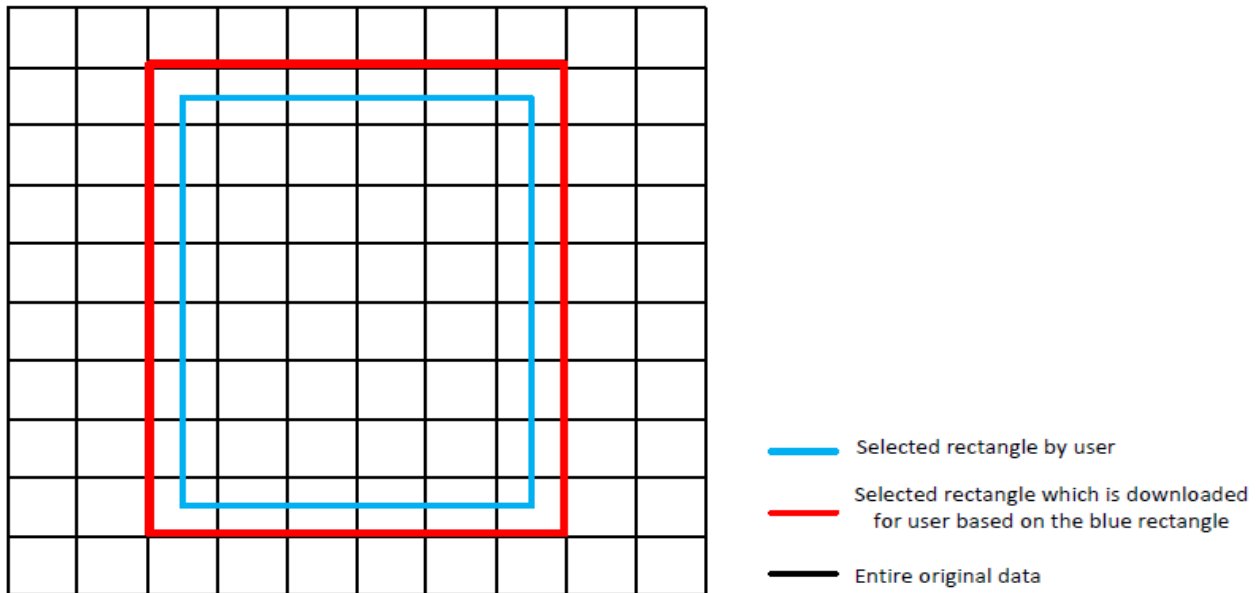


Figure 9.5. The structure of how a subset of the data is downloaded. Blue rectangle shows the drawn rectangle by user and the red rectangle is selected rectangle by server.

After all investigation to prove that these data showed the correct desire subsets, now it should be investigated that the downloaded data are coming from the original data. In this regard, the original data were imported to ArcGIS software. After that, the downloaded data (in both GeoTIFF and NetCDF format) were imported into the ArcGIS. Then the original data and the downloaded data were dropped on each other to be sure that they have the same coordinates and showing the same data. At the end, it was proved that the downloaded data was exactly extracted from the original data.

9.1.2. Evaluation of Retrieval of Subsets in time

Although retrieving the data in **space** subset was really important for this study, it was essential to investigate the WCS ability in retrieving the data in the **time** subset. In this regard, two downloaded data were compared. These data were examined to prove that they are showing the maximum daily temperature for two different times (figure 9.6). In order to know what times are available to be extracted from the original data, the metadata of the data should be examined (*DescribeCoverage* in Appendix B shows all the available time of the data that user can

download). The left figure shows the maximum daily temperature for first of January and the right figure shows the maximum daily temperature for first of February obtained from the Rasdaman/Petascopie technique.

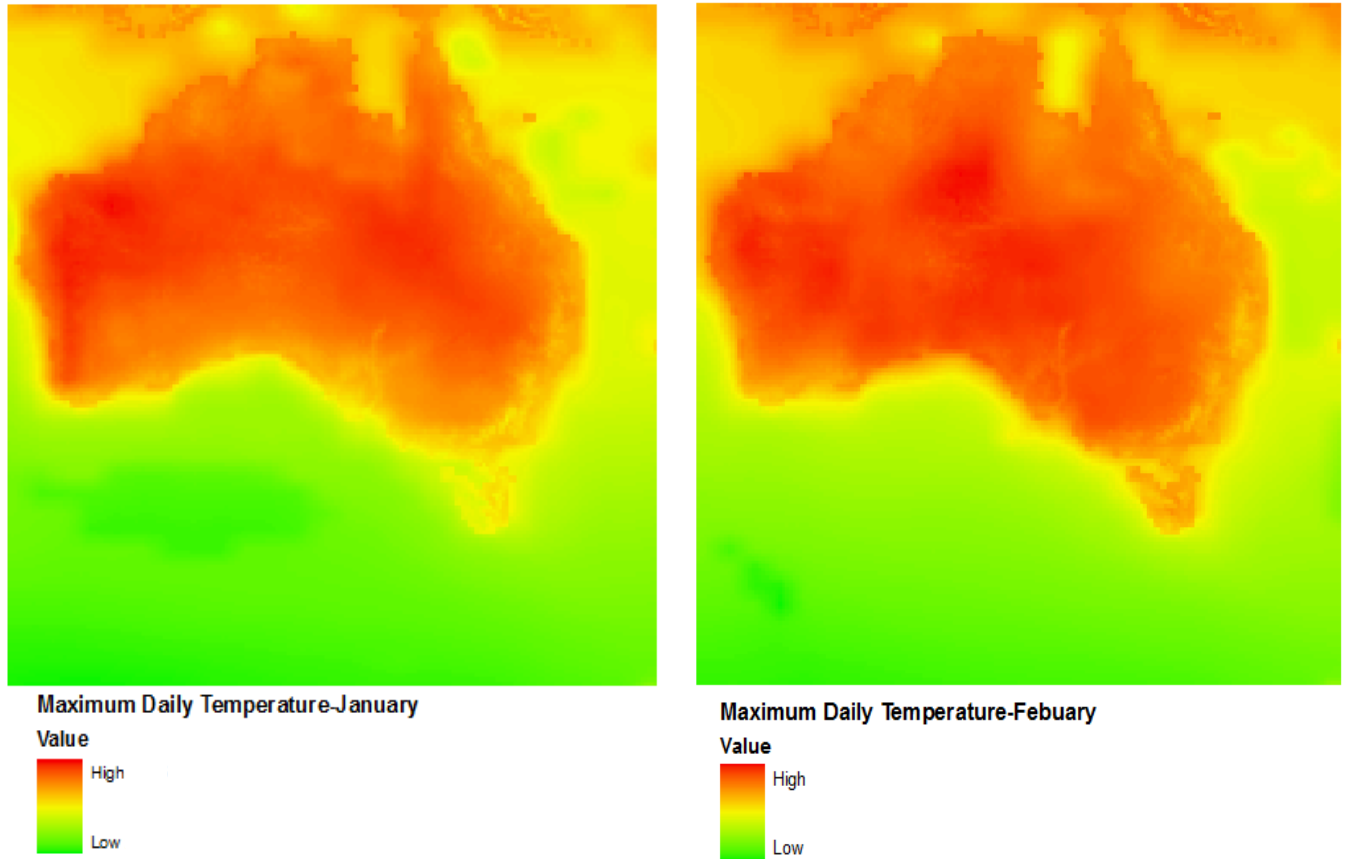


Figure 9.6. Two different downloaded data for two different times obtained from Rasdaman/Petascopie technique. The left figure shows maximum daily temperature for first of January and the right one shows first of February in 2100.

Based on the figure 9.6, it is clear that these two data show two different time subset. But in order to be sure that they retrieved exactly for the requested subset of time, their metadata also were retrieved through ArcGIS Layer properties (of figure 9.7).

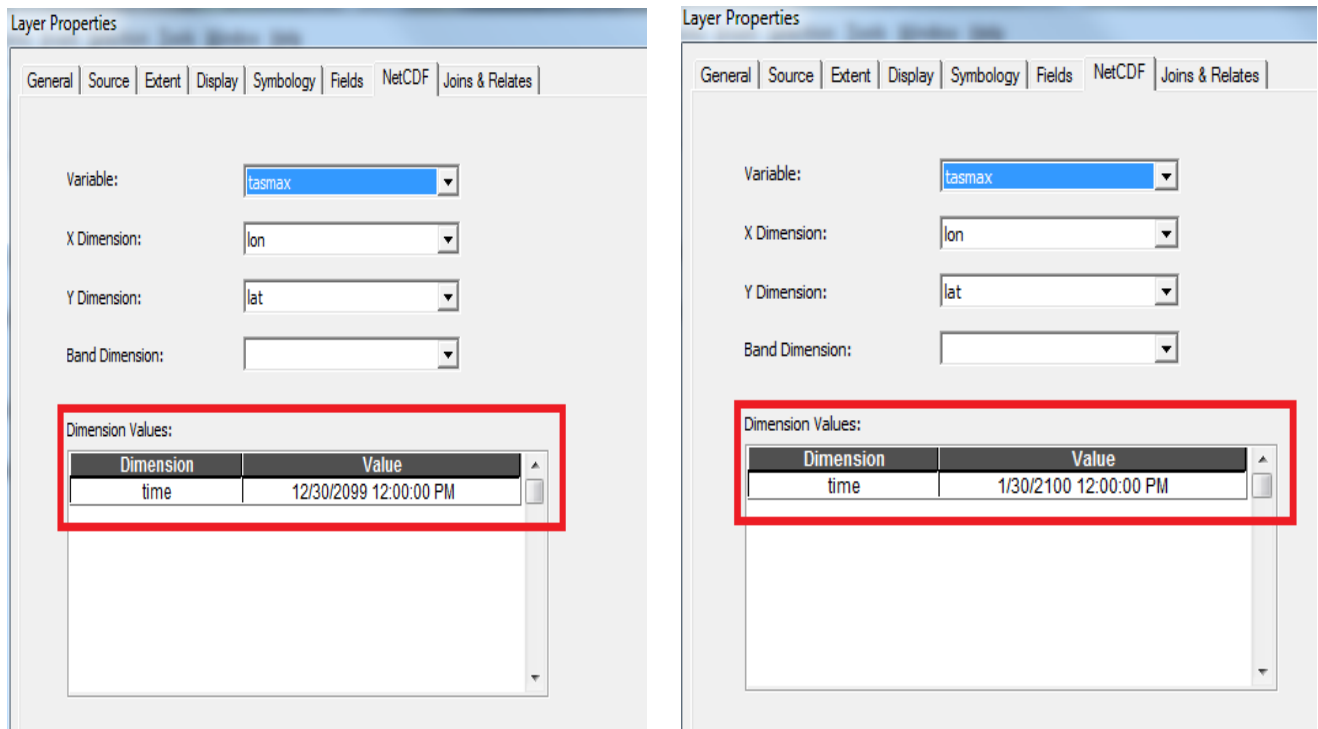


Figure 9.7. Two different downloaded data belong to their desired time obtained from Rasdaman/Petascopie technique. The left figure shows the first of January of year 2100. The right figure shows the first of February of year 2100.

As shown in figure 9.7, the left figure shows the time for the first of January in 00:00 AM and the right one shows first of February in 00:00 AM in year 2100.

So, the time subset functionality works correctly for the first case study as well.

9.2 Evaluation of case study 2- NetCDF and THREDDS

9.2.1. Evaluation of Retrieval of Subsets in space

In this section, an evaluation was done to prove the second technique works correctly to download the data in any desired space subsets. Like what was done for the first technique evaluation, two data in same space subsets were evaluated in both GeoTIFF and NetCDF formats visually and through their metadata. At the end, the numerical values of downloaded data were checked to be same with original data.

Figure 9.8 shows the GeoTIFF file visually which obtained for Longitude (112, 159) and Latitude (-54, -9) for Australia by NetCDF and THREDDS technique.

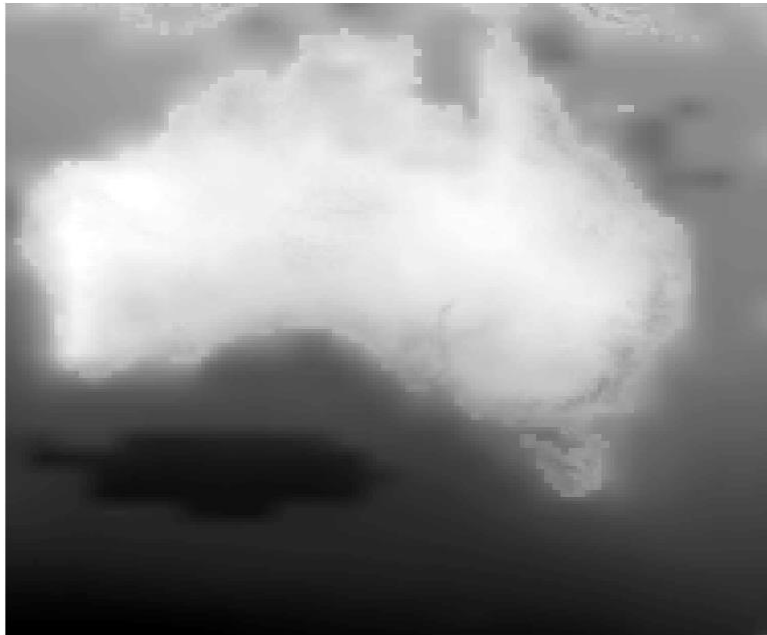


Figure9.8. *The subset of the data which are downloaded for specific space with Longitude ($x_{min}=112$, $x_{max}=159$) and Latitude ($y_{min}= -54$, $y_{max}= -9$) for Australia in GeoTIFF format obtained through NetCDF/THREDDS technique.*

The coordinate system of these data was Geographic coordinate system (WGS-84/EPSSG-4326). The locations were retrieved by Longitude and Latitude in degree.

So, to prove that these data are showing the correct space subsets, their metadata were retrieved. The Linux Terminal Command was used to show their metadata. Figure 9.9 illustrates the information of these data.


```
TIFFTAG_YRESOLUTION=1
Image Structure Metadata:
  INTERLEAVE=BAND
Corner Coordinates:
Upper Left  ( 112.0000000, -8.7500000) (112d 0' 0.00"E,  8d45' 0.00"S)
Lower Left  ( 112.0000000, -54.0000000) (112d 0' 0.00"E,  54d 0' 0.00"S)
Upper Right ( 159.2500000, -8.7500000) (159d15' 0.00"E,  8d45' 0.00"S)
Lower Right ( 159.2500000, -54.0000000) (159d15' 0.00"E,  54d 0' 0.00"S)
Center      ( 135.6250000, -31.3750000) (135d37'30.00"E, 31d22'30.00"S)
Band 1 Block=189x1 Type=Byte, ColorInterp=Gray
```

Figure 9.9. The extracted information for the GeoTIFF file obtained from NetCDF/THREDDS technique with bounding box ($x_{min}=112$, $y_{min}=-54$, $x_{max}=159$, $y_{max}=-9$). It proves the extracted data belongs to this defined bounding box.

Then, it was investigated if the NetCDF file extracted from this technique shows the desired subset of space or not. The data in the NetCDF format were investigated with the same space subset as above for Australia. This NetCDF file format was evaluated visually and through its metadata. For evaluating this NetCDF file visually, 'ncview' tool was used through Linux Terminal command as shown in figure 9.10. In addition, to prove that these data were extracted in their defined subsets correctly, their metadata was extracted through Linux Terminal command. Figure 9.11 shows that the downloaded NetCDF data were retrieved correctly in their defined subsets.

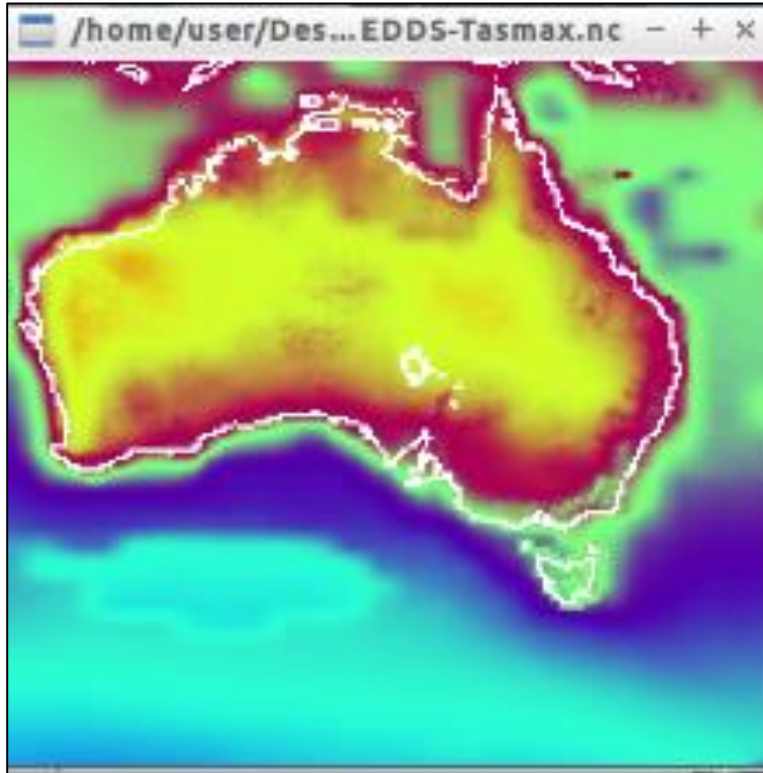


Figure 9.10. The NetCDF format obtained from NetCDF/THREDDS technique. These data show the specific subset with Longitude (112, 159) and Latitude (-54, -9).

```

"Original Dataset = /usr/local/tomcat/content/thredds/public/common/netcdf/dataDemo/tasmax_day_BCSd_rcp85_r1i1p1_ACCESS1-0_2100.nc; Translation Date = 2015-09-25T09:27:49.449Z" ;
:geospatial_lat_min = -53.875 ;
:geospatial_lat_max = -8.875 ;
:geospatial_lon_min = 112.125 ;
:geospatial_lon_max = 159.125 ;
}

```

Figure 9.11. The extracted information (metadata) for the NetCDF file obtained from NetCDF/THREDDS technique with bounding box ($x_{min}=112$, $y_{min}=-54$, $x_{max}=159$, $y_{max}=-9$). It proves the extracted data belongs to this defined bounding box.

Finally, As what was done for the first technique, the original data and the downloaded data were dropped on each other through ArcGIS software to prove that the downloaded data show exactly the original data. Based on this evaluation, the downloaded data which obtained through NetCDF/THREDDS technique showed the original data.

9.2.2. Evaluation of Retrieval of Subsets in time

To prove that the second technique works correctly to retrieve the data in subset of time, two downloaded data were compared for two different times. Figure 9.12 shows these two data. One of them was extracted for the first of January and the other one was extracted for the first of February.

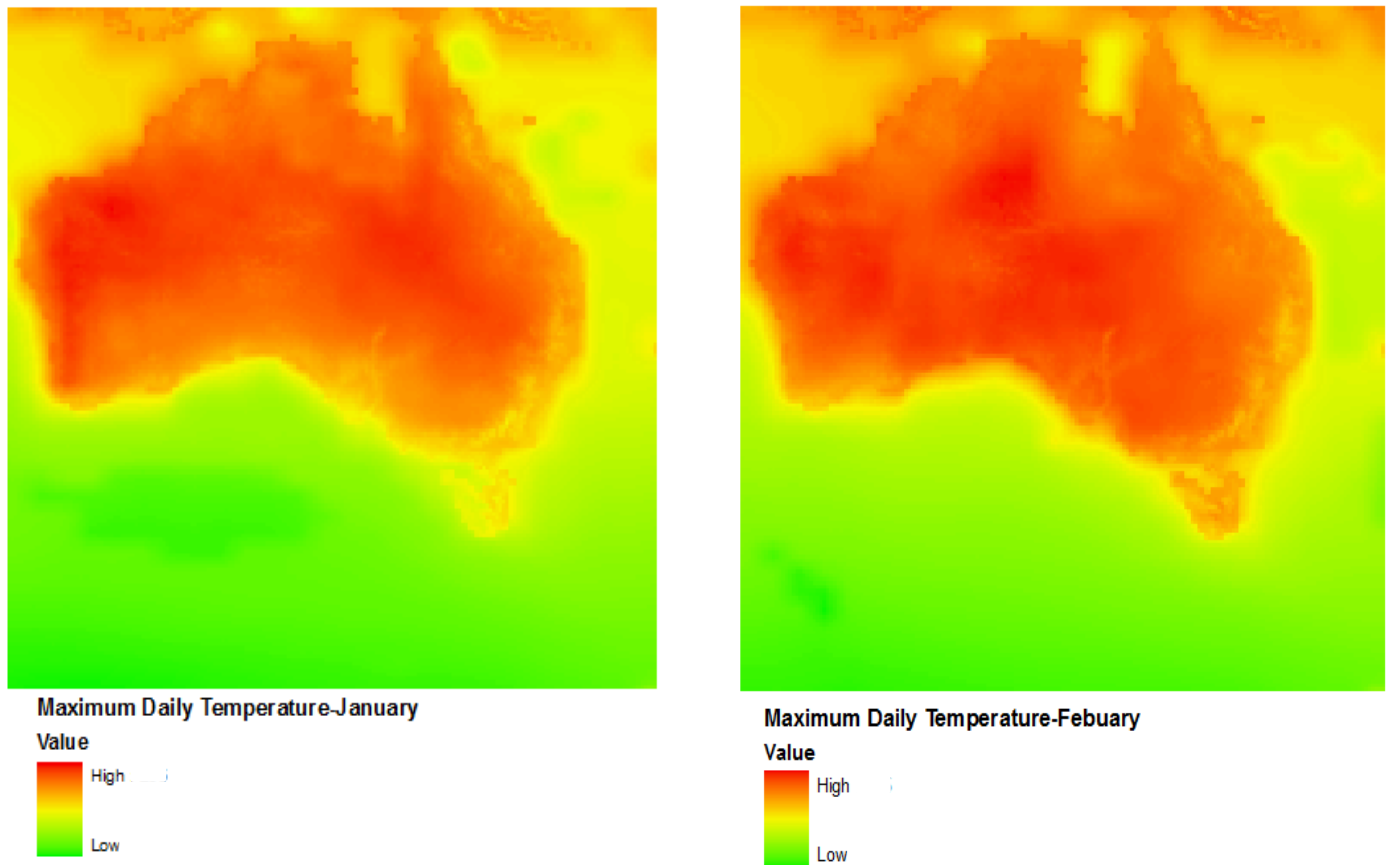


Figure 9.12. Two different downloaded data belong to their desired time obtained from the second case study. The left figure shows maximum daily temperature for first of January and the right one shows first of February in 2100.

It is clear that the extracted data are showing two different data for two different times. But in order to be sure that they were retrieved exactly for the desired time, their metadata is extracted by ArcGIS Layer properties. Figure 9.13 shows the information about the specific time of these downloaded data.

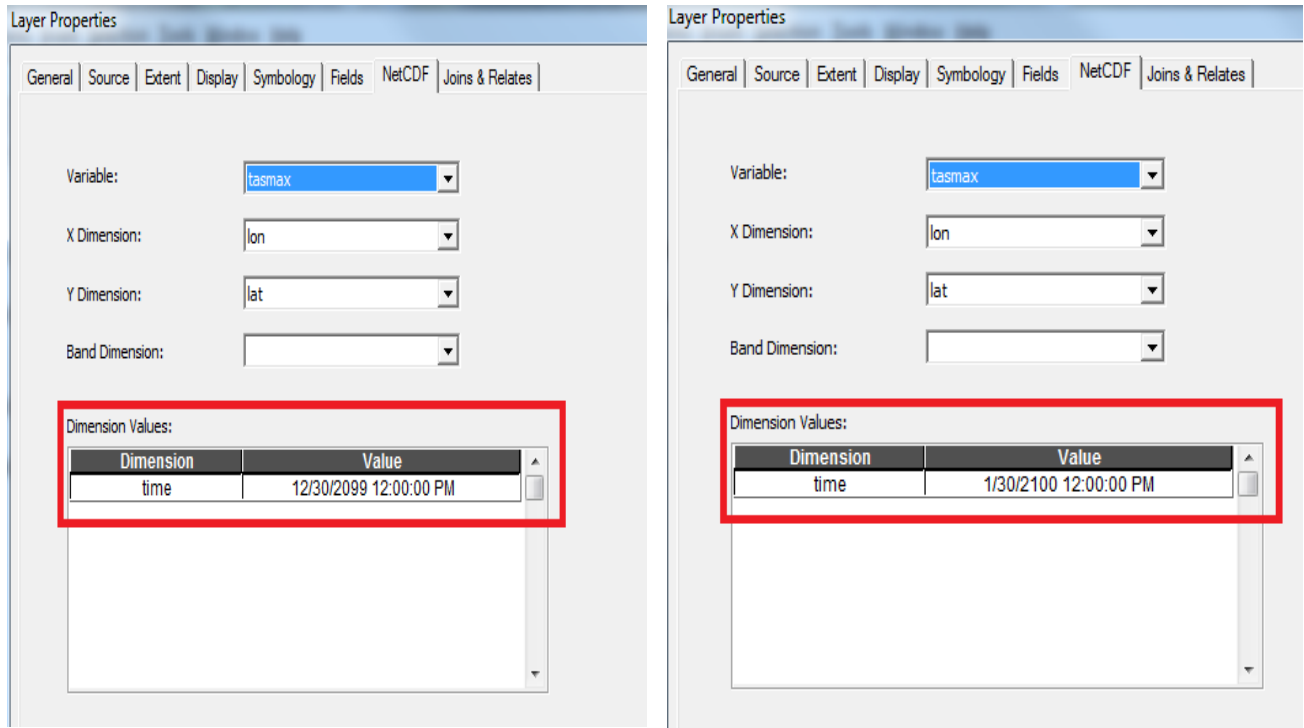


Figure 9.13. Two different downloaded data belong to their desired time obtained from the second case study. The left figure shows the first of January of year 2100. The right figure shows the first of February of year 2100.

As shown in figure 9.13, the left figure shows the time for the first of January in 00:00 AM and the right one shows first of February in 00:00 AM in year 2100.

Based on the above evaluations, the time subset functionality works correctly through NetCDF/THREDDS technique.

10. Discussion

The geospatial raster data typically have challenges in establishing a good technical solution for their storage, standard services and software interfaces. As geospatial raster data are gradually increased in different formats, it is important to develop good solutions. The solutions should be tested for each kind of geospatial raster data. Since ICOS data are geospatial raster data in NetCDF format, this study was in challenge to find the best technical solution for them. This solution was aimed to fulfill the user's requirements of ICOS CP.

The purpose of this study was to evaluate the WCS standard for distribution of ICOS data by experimenting with two different WCS web server techniques. We sought to determine which server is proper to be assigned in ICOS CP project. It was also important to evaluate that it is possible to create a WCS client service by using standard tools.

10.1 Case study 1- Rasdaman and Petascope

Rasdaman as a professional geospatial raster database was installed on the Linux environment. The most difficult part of working with Rasdaman was its installation. It took a very long time to conduct the installation since we always received many errors which were difficult to solve. Actually, as there are a few users who are working with Rasdaman, finding a solution for the errors was really difficult. After successful installation of Rasdaman, Petascope was configured in it. It also had its own problems. Receiving many individual errors in each step of working with Rasdaman and its Petascope was making it to be not user-friendly software. It requires a professional background in working with Linux environment and Rasql language.

After installation of Rasdaman and Petascope, the data were imported into the Rasdaman through Petascope tool. We had some problems with our data. Since the ICOS data are not still available, we had to change the data many times to find a proper one similar to ICOS data. When we had to change the imported data, the Petascope received unknown errors. In this situation, we had to do all the implementation from the beginning. In addition, when a data was large in its size, it was difficult for Rasdaman to store that data since it took a long time to import it.

In the evaluation of Petascope WCS server, it was really important for this study to see that WCS in relation with Petascope server can support the NetCDF and GeoTIFF formats. Although it took substantial time to set all the proper WCS parameters, WCS requests were working

perfectly in relation with Petascope. We could retrieve the data with *GetCoverage* request before shipping the data to the client.

In the last step, the OpenLayers was used as WCS client. It was very user-friendly library. It worked properly with WCS requests. But writing the WCS OpenLayers's JavaScript codes took time since there is not any good reference for WCS protocol working with OpenLayers available.

10.2 Case study 2- NetCDF and THREDDS

The THREDDS was installed on the Linux environment. There was not any challenge in installing the THREDDS. The data stored in NetCDF advanced file-based system were configured successfully in an easy way on the THREDDS. As NetCDF was on the file-based system, it did not allow taking a query or manipulating the data. It only retrieved the data with its whole file. THREDDS WCS server had some limitations. The data which were defined to be loaded on this server had to contain gridded data. In addition, the coordinate reference system had to be checked to be identifiable by the NetCDF-java Common Data Model. Otherwise the data were not readable through the server. As the THREDDS is software to keep the data in itself, it did not let us manipulate the data on the server side.

Then the WCS requests were designed based on the loaded data same as first case study. THREDDS was working perfectly with WCS requests without having any problem. It was possible to retrieve the data in both NetCDF and GeoTIFF formats before connecting to the client.

An important point was that WCS request structures were different in these two techniques. Petascope cannot support WCS before version 2. So, the version of WCS in first technique was different with the second technique. In this regard, the used parameters were different. For example, in the first technique, we called the coverage by 'coverageid' parameter. But in the second technique, we had to call the coverage by 'Coverage' parameter. Furthermore, the formats used in the first technique were stated as 'application/NetCDF' and 'image/tiff'. And the formats used in the second technique were defined as 'NetCDF3' and 'Geotiff'. Also, to retrieve the subset of space and time in the first technique, we had to use the 'subset' parameter to retrieve the data in space and time subsets but in the second technique we used 'bbox' for the

space subset and ‘time’ for the time subset. It was impossible to use the same structure for each technique since the request did not work.

At the end the same processes were done for the OpenLayers to connect to the THREDDS WCS server. But the JavaScript code was slightly different in these two techniques since the created WCS requests were different.

10.3 Future work

Based on the results of study, there are several recommendations for future research. Some of the limitations outlined in this study may be minimized or eliminated in a revised implementation of the techniques. First, to improve the capability of the first case study technique, the user instead of taking a long time to install Rasdaman, they can use a Virtual Machine like OSGeo-Live (OSGeo-Live, 2015) which contains a wide variety of open-source geospatial software. In addition, if someone wants to install the Rasdaman, the new version of the Rasdaman released in July 2015 is easier to be installed and configured (Rasdaman, 2015b). This new version has a new python import tool for petascope, ‘wcst_import’ (an alternative to Rasimport). It does not have the Petascope configuration problems and is easier to work with in comparing to the ‘Rasimport’ utility. Second, in working with THREDDS, it is suggested to check the data type and its attribute before starting any implementation. Since, all the data are not proper to work with THREDDS, they should be manipulated before their usage. For example, in this study, we could have checked the information and coordinate systems of the data before importing them into the THREDDS when we received unsuitable result instead of wasting time on trying some different data. Finally, this study only provided the users to view and download the data in NetCDF and GeoTIFF formats in subset of the space and time. Future studies could implement the WCPS standard to provide the ability for users to also process and analyze the ICOS data.

11. Conclusions

Several conclusions can be made based on the results of this study. The first conclusion is that WCS standard is an easy-to-use protocol in distributing the geospatial raster data in subsets of space and time with different formats. The second conclusion is Petascope as a WCS server tool is a very professional server specifically in relation with geospatial raster data but it requires expert knowledge. In contrast, THREDDS as a WCS server does not require professional skill to work with. The geospatial raster data in NetCDF advanced file-based system can be located in THREDDS data server very easily. The third conclusion is that OpenLayers is a compatible library in working with WCS standard and user can download a subset of the data using the WCS syntax.

First conclusion

WCS as a standard protocol for distribution of geospatial raster data was under evaluation in this study. Although fixing the proper parameters for WCS standard takes a long time, it is a sufficient protocol in order to retrieve the data. It supports NetCDF and GeoTIFF data formats and lets users define properly the subset of the data in space dimension, time dimension, etc. Additionally, it gives all the required information about the data before requesting for *GetCoverage*. It means that the user is aware about all the data's details. OPeNDAP was not chosen in this study since there are a few servers and clients to support it (for example Petascope and OpenLayers cannot support it). In addition, retrieving the different subsets by OPeNDAP is not as easy as WCS.

Second conclusion

The Rasdaman/Petascope technique enables a user to manipulate and store the data in a professional geospatial raster database. This technique provides the user with some options.

- The user can store the data in any desired formats in Rasdaman. It means that the user is not concern about the format of the data since Rasdaman supports most kind of Geospatial raster data formats and can convert the data format to the other formats.
- It is possible to make any desired query from the data. These queries can be shown through Rview interface Rasdaman tools without connecting to any client software.

- Furthermore, the user can extract all the information about the data directly from the Rasdaman database.
- The Petascope tool of Rasdaman supports OGC standards specifically WCS standard.
- Petascope gives this ability to user to make a subset of the data in space, time, etc. internally before connecting to any OGC standard. It means that the user can divide the data to many different subsets in the server side and then render any of these subsets to the standards.
- Petascope tool supports NetCDF and GeoTIFF formats of the data.
- In addition, it is possible to define any desired coordinate reference system of the data when they are imported to the Rasdaman by Petascope tool.

But, Rasdaman/Petascope technique is not an easy way to store the data and connect them to the WCS client. Since this technique is completely run on the Linux environment, it is not an easy technique for all kind of users. Actually, Rasdaman is not fully compatible with all NetCDF formats. In some situation, we have to change the NetCDF format to the other geospatial raster formats by Rasql then store it in the Rasdaman. In addition, it always receives some unknown errors which are really difficult to find the solution. But, this technique has some potential for users who are professional in working with Linux environment and the SQL language.

Conversely, the NetCDF/THREDDS technique lets user locate the data in an advanced file-based system. It provides the user with different kind of abilities.

- The NetCDF/THREDDS technique is an easy-to-use method and is faster in comparison with the first technique.
- It does not need any professional skill in working with it. The client makes a request and the request is sent to the THREDDS directly.
- It supports most of the downloading standards such as WCS and OPeNDAP.
- It supports different formats such as NetCDF and GeoTIFF formats.
- As THREDDS is basically designed in working with NetCDF advanced file-based systems, it is a suitable technique to be used for the ICOS data sever.

But since this technique is file-based, it does not let users make a query from the data. It sends whole of the data to the client side (Rasdaman as a raster database is more efficient on the server side to work with the data). Additionally, the users can not define their own coordinate reference

systems internally in the server side. It just defined the default coordinate reference system exists in the origin data. If users want to change the coordinate system, they should manipulate the data.

Third conclusion

OpenLayers was evaluated in this study to prove that it is an efficient client to work with WCS standard. This library works in a very compatible way with OGC standards specifically WCS standards. Its high ability in designing a web-based map with JavaScript language makes it prominent in comparing with other WCS clients. QGIS software is not an efficient client since it does not provide this study to support the NetCDF format.

References

- Aiordăchioaie, A. & Baumann, P. 2010. PetaScope: an open-source implementation of the OGC WCS geo service standards suite. *Scientific and Statistical Database Management*, pp. 160-168.
- Akerkar, R. 2013. *Big data computing*. CRC Press.
- Baart, F., de Boer, G., de Haas, W., Donchyts, G., Philippart, M., van Koningsveld, M. & Plieger, M. 2012. A comparison between WCS and OPeNDAP for making model results and data products available through the internet. *Transactions in GIS*, pp. 249-265.
- Bagui, S. 2003. Achievements and weaknesses of object-oriented databases. *Journal of Object Technology*, pp. 29-41.
- Baltic-Sea. 2015. Baltic Sea Bathymetry Database. Retrieved 30 May 2015, from <http://data.bshc.pro/>.
- Baumann, P. 2001. Web-enabled raster gis services for large image and map databases. *Database and Expert Systems Applications, 2001. Proceedings. 12th International Workshop*, pp. 870-874.
- Baumann, P. 2013. Query Language Guide. URL: http://rasdaman.eecs.jacobsuniversity.de/trac/rasdaman/browser/manuals_and_examples/manuals/pdf/ql-guide.pdf.
- Baumann, P. & Chulkov, G. 2007. Web Coverage Processing Service (WCPS). *OGC document*, pp. 08-068.
- Baumann, P. & Chulkov, G. 2008. Web coverage processing service (WCPS) implementation specification.
- Baumann, P., Dehm, A., Furtado, P., Ritsch, R. & Widmann, N. 1998. The multidimensional database system RasDaMan. *ACM SIGMOD Record*, pp. 575-577.
- Baumann, P., Diedrich, E., Glock, C., Lautenschlager, M. & Toussaint, F. 2003. Large-scale multidimensional coverage databases. *26th GITA Annual Conference*, pp. 76-93.
- Buchwitz, M., Reuter, M., Schneising, O., Boesch, H., Guerlet, S., Dils, B., Aben, I., Armante, R., Bergamaschi, P. & Blumenstock, T. 2013. The Greenhouse Gas Climate Change Initiative (GHG-CCI): Comparison and quality assessment of near-surface-sensitive satellite-derived CO₂ and CH₄ global data sets. *Remote Sensing of Environment*, pp. 344-362.
- Campalani, P., Guo, X. & Baumann, P. Spatio-Temporal Big Data.
- CarboScope. 2015. *CarboScope - Carbon Greenhouse gases at the Earth's Surface*. Retrieved 9 August 2015, from <http://www.carboscope.eu/>.
- Carter, R.N., Higgins, W.F. & Lee, R.O. 1995. File based and highly available hybrid database. In Google Patents.
- Cong-cong, X. & Li-ying, W. 2010. Notice of Retraction Study of Image Display with NetCDF Data on WebGIS. *Information Technology and Computer Science (ITCS), 2010 Second International Conference*, pp. 368-371.
- Connolly, T.M. & Begg, C.E. 2005. *Database systems: a practical approach to design, implementation, and management*. Pearson Education.
- Cornillon, P., Gallagher, J. & Sgouros, T. 2003. OPeNDAP: Accessing data in a distributed, heterogeneous environment. *Data Science Journal*, pp. 164-174.
- CropScope. 2015. *CropScope - Crops Land Data Layer*. Retrieved 5 June 2015, from <http://nassgeodata.gmu.edu/CropScope/>.
- Davidson, J. & Aundhe, S. 2001. Text-file based relational database. In Google Patents.

- EarthLook. 2015. Big Earth Data tandards. Retrieved 10 July 2015, from <http://www.earthlook.org/>.
- Elmasri, R. & Navathe, S. 2011. *Database systems*. Pearson Education.
- FOSS4G. 2011. OPeNDAP vs. WCS. Retrieved 8 July 2015, from <http://2011.foss4g.org/sessions/.opendap-vs-wcs.html>.
- Frank, M. 1995. Object-relational hybrids. *DBMS*, pp. 46-57.
- Gao, K., Liao, W.-k., Nisar, A., Choudhary, A., Ross, R. & Latham, R. 2009. Using subfiling to improve programming flexibility and performance of parallel shared-file I/O. *Parallel Processing, 2009. ICPP'09. International Conference*, pp. 470-477.
- geos.ed.ac.uk. 2015. OpenLayers. Retrieved 14 May 2015, from www.geos.ed.ac.uk.
- Goodchild, M.F., Fu, P. & Rich, P. 2007. Sharing geographic information: an assessment of the Geospatial One-Stop. *Annals of the Association of American Geographers*, pp. 250-266.
- Han, W., Yang, Z., Di, L. & Mueller, R. 2012. CropScope: A Web service based application for exploring and disseminating US conterminous geospatial cropland data products for decision support. *Computers and Electronics in Agriculture*, pp. 111-123.
- Hankin, S.C., Blower, J.D., Carval, T., Casey, K.S., Donlon, C., Lauret, O., Loubrieu, T., Srinivasan, A., Trinanes, J. & Godoy, O. 2010. NetCDF-CF-OPeNDAP: Standards for ocean data interoperability and object lessons for community data standards processes. *Oceanobs 2009, Venice Convention Centre, 21-25 septembre 2009, Venise*.
- Paredaens, Jan, Paul De Bra, Marc Gyssens, and Dirk Van Gucht. 2012. *The structure of the relational database model*. Vol. 17. Springer Science & Business Media.
- Hazzard, E. 2011. *Openlayers 2.10 beginner's guide*. Packt Publishing Ltd.
- Hugentobler, M. 2008. Quantum GIS. *Encyclopedia of GIS*, pp. 935-939. Springer.
- icos-cp. 2015. ICOS Carbon Portal Structure. Retrieved 13 June 2015, from www.icos-infrastructure.eu.
- Jetbrains .2015. JetBrains IntelliJ IDEA editor tool. Retrieved 31 May 2015, from <https://www.jetbrains.com/>.
- Jianwei, L., Liao, W., Choudhary, A., Ross, R., Thakur, R., Gropp, W., Latham, R., Siegel, A., Gallagher, B. & Zingale, M. 2003. Parallel netCDF: A high-performance scientific I/O interface, Supercomputing ACM. *IEEE Conference*, p. 39.
- John Caron, U. & Davis, E. 2006. UNIDATA's THREDDS data server. *22nd International Conference on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology*.
- jsfiddle. 2015. Jsfiddle Cloud. Retrieved 24 May 2015, from <https://jsfiddle.net/>.
- Karabegovic, A. & Ponjavic, M. 2012. Geoportal as decision support system with spatial data warehouse. *Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on*, pp. 915-918.
- Kiehle, C., Greve, K. & Heier, C. 2007. Requirements for Next Generation Spatial Data Infrastructures-Standardized Web Based Geoprocessing and Web Service Orchestration. *Transactions in GIS*, pp. 819-834.
- Kraak, M.-J. & Ormeling, F. 2011. *Cartography: visualization of spatial data*. Guilford Press.
- Lu, X. 2005. An investigation on service-oriented architecture for constructing distributed web gis application. *Services Computing, 2005 IEEE International Conference on*, pp. 191-197.
- Mahammad, S.S. & Ramakrishnan, R. 2003. GeoTIFF-A standard image file format for GIS applications. *Map India*, pp. 28-31.

- Manso, M. & Bernabé, M. 2005. Open Source components to build a GeoPortal. *11 th EC GI & GIS Workshop ESDI: Setting the Framework Sardinia, June 2005*, p. 90.
- NASA-EarthObservation-Download. 2015. Download Real-Time Data. Retrieved 12 July 2015, from <https://earthdata.nasa.gov/earth-observation-data/near-real-time/download-nrt-data>.
- NASA-OMI. 2015. Download NASA Global Climate Change Data. Retrieved 1 May 2015, from http://acdisc.gsfc.nasa.gov/opensap/HDF-EOS5/Aura_OMI_Level3/.
- NASA. 2015. NASA New Global Climate Change Projections. Retrieved 17 July 2015, from <http://www.dailymail.co.uk/sciencetech/article-3125113/Earth-2100-Nasa-maps-reveal-world-need-adapt-rising-temperatures-caused-climate-change.html>.
- Nativi, S., Caron, J., Davis, E. & Domenico, B. 2005. Design and implementation of netCDF markup language (NcML) and its GML-based extension (NcML-G ML). *Computers & Geosciences*, pp. 1104-1118.
- NCCS-THREDDS. 2015. NASA THREDDS DataServer. Retrieved 21 June 2015, from <http://dataserver3.nccs.nasa.gov/thredds/catalog/bypass/NEX-GDDP/catalog.html>.
- Nguyen, G.T. & Rieu, D. 1987. Expert database support for consistent dynamic objects. *Proceedings of the 13th International Conference on Very Large Data Bases*, pp. 493-500.
- NOAA. 2015. Earth System Research Laboratory. Retrieved 13 July 2015, from <http://www.esrl.noaa.gov/psd/>.
- OGC. 2015. Open Geospatial Consortium. Retrieved 13 March 2015, from <http://www.opengeospatial.org/>.
- OpenLayers. 2015. *OpenLayers*. Retrieved 11 May 2015, from <http://openlayers.org/>.
- OSGeo-Live. 2015. OSGeo-Live Virtual Machine. Retrieved 1 May 2015, from www.live.osgeo.org.
- Petascopes.User.Guide. 2015. Petascope Installation and Guidance. Retrieved 8 Feb 2015, from <http://www.rasdaman.org/wiki/PetascopeUserGuide>.
- Rasdaman. 2015a. Raster Data Manager. Retrieved 25 January 2015, from <http://www.rasdaman.com/>.
- Rasdaman. 2015b. Rasdaman.9.1.0. Retrieved 25 January 2015, from <http://rasdaman.org/wiki/Versions#Version9.1.x>.
- Rasdaman. 2015c. Rasdaman Applications. Retrieved 28 Feb 2015, from <http://rasdaman.org/wiki/ApplicationDomains>
- Rasdaman. 2015d. Rasdaman Installation. Retrieved 8 Feb 2015, from <http://www.rasdaman.org/wiki/Install>.
- Rautenbach, V., Coetzee, S. & Iwaniak, A. 2013. Orchestrating OGC web services to produce thematic maps in a spatial information infrastructure. *Computers, Environment and Urban Systems*, pp. 107-120.
- Remotesensing. 2011. GeoTiff usage in Remotesensing Science. Retrieved 15 Feb 2015, from <http://www.remotesensing.org/geotiff/spec/contents.html>.
- Rew, R. & Davis, G. 1990. NetCDF: an interface for scientific data access. *Computer Graphics and Applications, IEEE*, pp. 76-82.
- Rew, R., Davis, G., Emmerson, S. & Davies, H. 1997. NetCDF user's guide for C. *Unidata Program Center*, June, pp. 997.
- Rieu, D. & Nguyen, G.T. 1986. Semantics of CAD objects for generalized databases. *Proceedings of the 23rd ACM/IEEE Design Automation Conference*, pp. 34-40.

- Rigaux, P., Scholl, M. & Voisard, A. 2001. *Spatial databases: with application to GIS*. Morgan Kaufmann.
- Ritsch, R. 1999. *Optimization and evaluation of array queries in database management systems*. Technische Universität München.
- Ritter, N. & Ruth, M. 1997. The GeoTiff data interchange standard for raster geographic images. *International Journal of Remote Sensing*, pp. 1637-1647.
- Ritter, N., Ruth, M., Grissom, B.B., Galang, G., Haller, J., Stephenson, G., Covington, S., Nagy, T., Moyers, J. & Stickley, J. 2000. GeoTIFF format specification GeoTIFF revision 1.0. URL: <http://www.remotesensing.org/geotiff/spec/geotiffhome.html>.
- Shekhar, S. & Xiong, H. 2008. *Encyclopedia of GIS*. Springer Science & Business Media.
- StackExchange. 2015. GIS Stack Exchange question and answers. Retrieved 30 May 2015, from <http://gis.stackexchange.com>.
- Stakeholders-Handbook. 2013. Integrated Carbon Observation System.
- Stonebraker, M. & Hellerstein, J.M. 1998. *Readings in database systems*. Morgan Kaufmann, San Francisco.
- Tatnall, A. 2005. *Web portals: the new gateways to Internet information and services*. IGI Global.
- Unidata. 2015. THREDDS Data Server Structure. Retrieved 8 May 2015, from <http://motherlode.ucar.edu/>.
- Unidata-NetCDF. 2015. Network Common Data Form. Retrieved 18 Feb 2015, from <http://www.unidata.ucar.edu/software/netcdf/>.
- VisibleHuman. 2015. The Visible Human Project. Retrieved 28 July 2015, from http://www.nlm.nih.gov/research/visible/visible_human.html.
- Wang, M. 2010. Using object-relational database technology to solve problems in database development. pp. 10.
- Weikum, G., Kasneci, G., Ramanath, M. & Suchanek, F. 2009. Database and information-retrieval methods for knowledge discovery. *Communications of the ACM*, pp. 56-64.
- Whiteside, A. & Evans, J. D. 2006. Web coverage service (wcs) implementation specification. *Open Geospatial Consortium*.
- Whiteside, A. & Evans, J.D. 2008. Web Coverage Service (WCS) Implementation Standard. *Open Geospatial Consortium*.
- Worboys, M.F. & Duckham, M. 2004. *GIS: a computing perspective*. CRC press.
- Xie, T. 2008. Sea: A striping-based energy-aware strategy for data placement in raid-structured storage systems. *Computers, IEEE Transactions on*, pp. 748-761.
- Zhang, T., Clarke, K.C. & Guan, Q. 2006. The GIS Web portal: Beyond data services. *Proceedings, AutoCart*.

Appendix A-GetCapabilities Response

This is the response from the Petascope and THREDDS WCS GetCapabilities request in section 8.4.2 and 8.5.2. It describes clearly all the WCS capabilities and available coverages. It gives the information for all the available coverages stored through these techniques. As this response gives the same information in both techniques, only the response from THREDDS techniques is shown here. The document tree is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<WCS_Capabilities xmlns="http://www.opengis.net/wcs"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.0.0">

  <Service>
    <fees>NONE</fees>
    <accessConstraints>NONE</accessConstraints>
  </Service>
  <Capability>
    <Request>
      <GetCapabilities>
        <DCPType>
          <HTTP>
            <Get>
              <OnlineResource xlink:href="http://thredds.icos-
cp.eu/thredds/wcs/common/netcdf/dataDemo/tasmax_day_BCSd_rcp85_r1i1p1_ACCESS1
-0_2100.nc" />
            </Get>
          </HTTP>
        </DCPType>
      </GetCapabilities>
      <DescribeCoverage>
        <DCPType>
          <HTTP>
            <Get>
              <OnlineResource xlink:href="http://thredds.icos-
cp.eu/thredds/wcs/common/netcdf/dataDemo/tasmax_day_BCSd_rcp85_r1i1p1_ACCESS1
-0_2100.nc" />
            </Get>
          </HTTP>
        </DCPType>
      </DescribeCoverage>
      <GetCoverage>
        <DCPType>
          <HTTP>
            <Get>
              <OnlineResource xlink:href="http://thredds.icos-
cp.eu/thredds/wcs/common/netcdf/dataDemo/tasmax_day_BCSd_rcp85_r1i1p1_ACCESS1
-0_2100.nc" />
            </Get>
          </HTTP>
        </DCPType>
      </GetCoverage>
    </Request>
    <Exception>
```

```

    <Format>application/vnd.ogc.se_xml</Format>
  </Exception>
</Capability>
<ContentMetadata>
  <CoverageOfferingBrief>
    <description>tasmax K true Daily Maximum Near-Surface Air
Temperature</description>
    <name>tasmax</name>
    <label>Daily Maximum Near-Surface Air Temperature</label>
    <lonLatEnvelope srsName="urn:ogc:def:crs:OGC:1.3:CRS84">
      <gml:pos>0.125 -89.875</gml:pos>
      <gml:pos>359.875 89.875</gml:pos>
      <gml:timePosition>2100-01-01T12:00:00Z</gml:timePosition>
      <gml:timePosition>2100-12-31T12:00:00Z</gml:timePosition>
    </lonLatEnvelope>
  </CoverageOfferingBrief>
</ContentMetadata>
</WCS_Capabilities>

```


Appendix B-DescribeCoverage Response

This is the response of Petascope and THREDDS WCS DescribeCoverage request in section 8.4.2 and 8.5.2. This response describes the ‘tasmax’ layer stored in Rasdaman and NetCDF. In addition, it provides all the required information about this layer. As this response gives the same information in both techniques, only the response from THREDDS techniques is shown here. The document tree is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CoverageDescription xmlns="http://www.opengis.net/wcs"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.0.0">
  <CoverageOffering>
    <description>tasmax K true Daily Maximum Near-Surface Air
Temperature</description>
    <name>tasmax</name>
    <label>Daily Maximum Near-Surface Air Temperature</label>
    <lonLatEnvelope srsName="urn:ogc:def:crs:OGC:1.3:CRS84">
      <gml:pos>0.125 -89.875</gml:pos>
      <gml:pos>359.875 89.875</gml:pos>
      <gml:timePosition>2100-01-01T12:00:00Z</gml:timePosition>
      <gml:timePosition>2100-12-31T12:00:00Z</gml:timePosition>
    </lonLatEnvelope>
    <domainSet>
      <spatialDomain>
        <EnvelopeWithTimePeriod srsName="urn:ogc:def:crs:OGC:1.3:CRS84">
          <gml:pos dimension="2">0.125 -89.875</gml:pos>
          <gml:pos dimension="2">359.875 89.875</gml:pos>
          <gml:timePosition>2100-01-01T12:00:00Z</gml:timePosition>
          <gml:timePosition>2100-12-31T12:00:00Z</gml:timePosition>
        </EnvelopeWithTimePeriod>
        <gml:RectifiedGrid srsName="EPSG:0 [Latitude_Longitude]"
dimension="2">
          <gml:limits>
            <gml:GridEnvelope>
              <gml:low>0 0</gml:low>
              <gml:high>1439 719</gml:high>
            </gml:GridEnvelope>
          </gml:limits>
          <gml:axisName>x</gml:axisName>
          <gml:axisName>y</gml:axisName>
          <gml:origin>
            <gml:pos>0.125 -89.875</gml:pos>
          </gml:origin>
          <gml:offsetVector>0.25 0.0</gml:offsetVector>
          <gml:offsetVector>0.0 0.25</gml:offsetVector>
        </gml:RectifiedGrid>
      </spatialDomain>
      <temporalDomain>
        <gml:timePosition>2100-01-01T12:00:00Z</gml:timePosition>
        <gml:timePosition>2100-01-02T12:00:00Z</gml:timePosition>
        <gml:timePosition>2100-01-03T12:00:00Z</gml:timePosition>
        <gml:timePosition>2100-01-04T12:00:00Z</gml:timePosition>

```

.....

```

    <gml:timePosition>2100-12-29T12:00:00Z</gml:timePosition>
    <gml:timePosition>2100-12-30T12:00:00Z</gml:timePosition>
    <gml:timePosition>2100-12-31T12:00:00Z</gml:timePosition>
  </temporalDomain>
</domainSet>
<rangeSet>
  <RangeSet>
    <description xmlns="">tasmax K true Daily Maximum Near-Surface Air
Temperature</description>
    <name>tasmax</name>
    <label>Daily Maximum Near-Surface Air Temperature</label>
    <nullValues>
      <singleValue>NaN</singleValue>
    </nullValues>
  </RangeSet>
</rangeSet>
<supportedCRSs>
  <requestCRSs>OGC:CRS84</requestCRSs>
  <responseCRSs>EPSG:0 [Latitude_Longitude]</responseCRSs>
</supportedCRSs>
<supportedFormats>
  <formats>GeoTIFF</formats>
  <formats>GeoTIFF_Float</formats>
  <formats>NetCDF3</formats>
</supportedFormats>
<supportedInterpolations>
  <interpolationMethod>none</interpolationMethod>
</supportedInterpolations>
</CoverageOffering>
</CoverageDescription>

```

Appendix C- OpenLayers JavaScript and HTML Code

These codes are inspired from OpenLayers3 examples (OpenLayers, 2015) and Geographic Information System questions and answers Stack Exchange website (Stackexchange, 2015).

```
<HTML>
<div class="row-fluid">
  <div class="span12">
    <div id="map" class="map"></div>
  </div>
  <div class="span4 offset4 pull-right">
    <div id="info" class="alert alert-success">
      &nbsp;
    </div>
  </div>
</div>
<form class="form-inline">
  <label class="checkbox"><input type="checkbox" id="geodesic"/>Download
Geotiff</label>
  <label class="checkbox"><input type="checkbox"
id="geodesic1"/>Download NetCDF</label>
</form>

<div class="span6">
  <form>
    <label>Projection </label>
    <select id="projection">
      <option value="EPSG:4326">EPSG:4326</option>
      <option value="EPSG:3857">EPSG:3857</option>
    </select>
    <label>Precision </label>
    <input id="precision" type="number" min="0" max="12" value="4"/>
  </form>
</div>
<div class="span6" id="mouse-position">&nbsp;</div>
<label>Date</label>
<select id="date">
  <option value="2100-01-01T12:00:00Z">2100-01-01T12:00:00Z</option>
  <option value="2100-01-02T12:00:00Z">2100-01-02T12:00:00Z</option>
  <option value="2100-01-03T12:00:00Z">2100-01-03T12:00:00Z</option>
  .....
  <option value="2100-12-28T12:00:00Z">2100-01-23T12:00:00Z</option>
  <option value="2100-12-29T12:00:00Z">2100-01-24T12:00:00Z</option>
  <option value="2100-12-30T12:00:00Z">2100-01-25T12:00:00Z</option>
</select></HTML>

//for Rasdaman/Petascoppe technique the date values were fixed between 1 to
360 since it gets the values in these way. For example: <option
value="1">2100-01-01T12:00:00Z</option>

<script>
/**
```

```

* Define a namespace for the application.
*/
window.app = {};
var app = window.app;

//
// Define download control.
//
var mousePositionControl = new ol.control.MousePosition({
  coordinateFormat: ol.coordinate.createStringXY(4),
  projection: 'EPSG:4326',
  // comment the following two lines to have the mouse position
  // be placed within the map.
  className: 'custom-mouse-position',
  target: document.getElementById('mouse-position'),
  undefinedHTML: '&nbsp;';
});

/**
 * @constructor
 * @extends {ol.control.Control}
 * @param {Object=} opt_options Control options.
 */
app.RotateNorthControl = function(opt_options) {

  var options = opt_options || {};

  var button = document.createElement('button');
  button.innerHTML = 'D';

  // add the dragbox Interaction to the map to let users drag a rectangle on
  // the specific area.

  var this_ = this;
  var handleRotateNorth = function(e) {
    map.addInteraction(dragBox);
  };

  button.addEventListener('click', handleRotateNorth, false);
  button.addEventListener('touchstart', handleRotateNorth, false);

  var element = document.createElement('div');
  element.className = 'rotate-north ol-unselectable ol-control';
  element.title = 'Draw An Area To Download';
  element.appendChild(button);

  ol.control.Control.call(this, {
    element: element,
    target: options.target
  });

};

ol.inherits(app.RotateNorthControl, ol.control.Control);

//

```

```

// Create map,
//
var layers = [
  new ol.layer.Tile({
    source: new ol.source.TileWMS({
      url: 'http://demo.boundlessgeo.com/geoserver/wms',
      params: {
        'LAYERS': 'ne:NE1_HR_LC_SR_W_DR'
      }
    })
  })
];

var map = new ol.Map({
  controls: ol.control.defaults({
    attributionOptions:
      ({
        collapsible: false
      })
  }).extend([
    new app.RotateNorthControl(), mousePositionControl,
    new ol.control.ScaleLine({
      units: 'degrees'
    })
  ]),
  layers: layers,
  target: 'map',
  view: new ol.View({
    projection: 'EPSG:4326',
    center: [0, 0],
    zoom: 2,
    rotation: 0
  })
});

var projectionSelect = $('#projection');
projectionSelect.on('change', function() {
  mousePositionControl.setProjection(ol.proj.get(this.value));
});
projectionSelect.val(mousePositionControl.getProjection().getCode());

var precisionInput = $('#precision');
precisionInput.on('change', function() {
  var format = ol.coordinate.createStringXY(this.valueAsNumber);
  mousePositionControl.setCoordinateFormat(format);
});

$('#ol-rotate-north button[title]').tooltip({
  placement: 'right'
});

// a normal select interaction to handle click
var select = new ol.interaction.Select();
map.addInteraction(select);

```

```

var selectedFeatures = select.getFeatures();

// a DragBox interaction used to select features by drawing boxes
var dragBox = new ol.interaction.DragBox({
  condition: ol.events.condition.always,
  style: new ol.style.Style({
    stroke: new ol.style.Stroke({
      color: [0, 0, 255, 1]
    })
  })
});

//map.addInteraction(dragBox);

var infoBox = document.getElementById('info');

var geotiffCheckbox = document.getElementById('geodesic');
var netCDFCheckbox = document.getElementById('geodesic1');

dragBox.on('boxend', function(e) {
  // features that intersect the box are added to the collection of
  // selected features, and their names are displayed in the "info"
  // div

  var datetime= document.getElementById("date").value;

  var info = [];
  var extent = dragBox.getGeometry().getExtent();

  // for the (NetCDF/THREDDS) technique: the following URL is used to retrieve
  the data from this technique:
  if (geotiffCheckbox.checked) {
    url = "http://thredds.icos-
cp.eu/thredds/wcs/common/netcdf/dataDemo/tasmax_day_BCSd_rcp85_r1i1p1_ACCESS1
-0_2100.nc";
    url += "?service=WCS";
    url += "&version=1.0.0";
    url += "&request=GetCoverage";
    url += "&COVERAGE=tasmax";
    url += "&format=GeoTIFF";
    url += "&bbox=" +extent;
    url += "&time=" +datetime;
  }else {
    url = "http://thredds.icos-
cp.eu/thredds/wcs/common/netcdf/dataDemo/tasmax_day_BCSd_rcp85_r1i1p1_ACCESS1
-0_2100.nc";
    url += "?service=WCS";
    url += "&version=1.0.0";
    url += "&request=GetCoverage";
    url += "&COVERAGE=tasmax";
    url += "&format=NetCDF3";
    url += "&bbox=" +extent;
    url += "&time=" +datetime;
  }
}

```

```

// for the (Rasdaman/Petascop) technique: the following URL is used to
retrieve the data from this technique:
    if (geotiffCheckbox.checked) { // this code is inspired
from Geographic Information System Stack Exchange website.

    url = " http://localhost:8080/rasdaman/ows/wcs";

    url += "?service=WCS";
    url += "&version=2.0.1";
    url += "&request=GetCoverage";
    url += "&COVERAGEID=tasmax";
    url += "&format=image/TIFF";
    url += "&subset=i(+extent.lng)";
    url += "&subset=j(+extent.lat)";
    url += "&subset=k(+datetime)";
    }else {
        url = " http://localhost:8080/rasdaman/ows/wcs";
    url += "?service=WCS";
    url += "&version=2.0.1";
    url += "&request=GetCoverage";
    url += "&COVERAGEID=tasmax";
    url += "&format=application/NetCDF";
    url += "&subset=i(+extent.lng)";
    url += "&subset=j(+extent.lat)";
    url += "&subset=k(+datetime)";

    }

    info.push(url);
    if (info.length > 0) {
        infoBox.innerHTML = info.join(', ');
        window.open(url, "_self")
    }
});

// clear selection when drawing a new box and when clicking on the map
dragBox.on('boxstart', function(e) {
    selectedFeatures.clear();
    infoBox.innerHTML = '&nbsp;';
});
map.on('click', function() {
    selectedFeatures.clear();
    infoBox.innerHTML = '&nbsp;';
});
</script>

```

Appendix D- THREDDS Configuration Catalog file in importing data

This XML document shows how the name of the data and their URLs are configured into the THREDDS environment. It also lets users define the name of their desired service.

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog
xmlns="http://www.unidata.ucar.edu/namespaces/thredds/InCatalog/v1.0"
xmlns:xlink="http://www.v3.org/1999/xlink"
version="1.0.2">
<dataset>
<service name="WCS" serviceType="WCS" base="/thredds/wcs/common/netcdf/"
/>
<dataset name="Yearly_fluxes"
urlPath="dataDemo/yearly_1x1_fluxes_limited.nc">
<serviceName>WCS</serviceName>
</dataset>
<dataset>
<service name="WCS" serviceType="WCS" base="/thredds/wcs/common/netcdf/"
/>
<dataset name="CO2_EUROPE_LSCE" urlPath="dataDemo/co2_EUROPE_LSCE.nc">
<serviceName>WCS</serviceName>
</dataset>
<dataset>
<service name="WCS" serviceType="WCS" base="/thredds/wcs/common/netcdf/"
/>
<dataset name="Tair_daily_WFDEI_201211"
urlPath="dataDemo/Tair_daily_WFDEI_201211.nc">
<serviceName>WCS</serviceName>
</dataset>
<dataset>
<service name="WCS" serviceType="WCS" base="/thredds/wcs/common/netcdf/"
/>
<dataset name="NASA_tasmax_day_BCSD_rcp85_rlilp1_ACCESS1-0_2100"
urlPath="dataDemo/NASA_tasmax_day_BCSD_rcp85_rlilp1_ACCESS1-0_2100.nc">
<serviceName>WCS</serviceName>
</dataset>
<catalogRef
xlink:href="http://thredds.icos-cp.eu/thredds/catalog.xml" />
</dataset></catalog>
```


Seminar Series

Institutionen för naturgeografi och ekosystemvetenskap, Lunds Universitet.

Student examensarbete (Seminarieuppsatser). Uppsatserna finns tillgängliga på institutionens geobibliotek, Sölvegatan 12, 223 62 LUND. Serien startade 1985. Hela listan och själva uppsatserna är även tillgängliga på LUP student papers (<https://lup.lub.lu.se/student-papers/search/>) och via Geobiblioteket (www.geobib.lu.se)

The student thesis reports are available at the Geo-Library, Department of Physical Geography and Ecosystem Science, University of Lund, Sölvegatan 12, S-223 62 Lund, Sweden. Report series started 1985. The complete list and electronic versions are also electronic available at the LUP student papers (<https://lup.lub.lu.se/student-papers/search/>) and through the Geo-library (www.geobib.lu.se)

- 335 Fei Lu (2015) Compute a Crowdedness Index on Historical GIS Data- A Case Study of Hög Parish, Sweden, 1812-1920
- 336 Lina Allesson (2015) Impact of photo-chemical processing of dissolved organic carbon on the bacterial respiratory quotient in aquatic ecosystems
- 337 Andreas Kiik (2015) Cartographic design of thematic polygons: a comparison using eye-movement metrics analysis
- 338 Iain Lednor (2015) Testing the robustness of the Plant Phenology Index to changes in temperature
- 339 Louise Bradshaw (2015) Submerged Landscapes - Locating Mesolithic settlements in Blekinge, Sweden
- 340 Elisabeth Maria Farrington (2015) The water crisis in Gaborone: Investigating the underlying factors resulting in the 'failure' of the Gaborone Dam, Botswana
- 341 Annie Forssblad (2015) Utvärdering av miljöersättning för odlingslandskapets värdefulla träd
- 342 Iris Behrens, Linn Gardell (2015) Water quality in Apac-, Mbale- & Lira district, Uganda - A field study evaluating problems and suitable solutions
- 343 Linnéa Larsson (2015) Analys av framtida översvämningsrisker i Malmö - En fallstudie av Castellums fastigheter
- 344 Ida Pettersson (2015) Comparing Ips Typographus and Dendroctonus ponderosus response to climate change with the use of phenology models
- 345 Frida Ulfves (2015) Classifying and Localizing Areas of Forest at Risk of Storm Damage in Kronoberg County
- 346 Alexander Nordström (2015) Förslag på dammar och skyddsområde med hjälp av GIS: En studie om löv- och klockgroda i Ystad kommun, Skåne
- 347 Samanah Seyedi-Shandiz (2015) Automatic Creation of Schematic Maps - A Case Study of the Railway Network at the Swedish Transport Administration
- 348 Johanna Andersson (2015) Heat Waves and their Impacts on Outdoor Workers – A Case Study in Northern and Eastern Uganda
- 349 Jimmie Carpmann (2015) Spatially varying parameters in observed new particle

- formation events
- 350 Mihaela – Mariana Tudoran (2015) Occurrences of insect outbreaks in Sweden in relation to climatic parameters since 1850
- 351 Maria Gatzouras (2015) Assessment of trampling impact in Icelandic natural areas in experimental plots with focus on image analysis of digital photographs
- 352 Gustav Wallner (2015) Estimating and evaluating GPP in the Sahel using MSG/SEVIRI and MODIS satellite data
- 353 Luisa Teixeira (2015) Exploring the relationships between biodiversity and benthic habitat in the Primeiras and Segundas Protected Area, Mozambique
- 354 Iris Behrens & Linn Gardell (2015) Water quality in Apac-, Mbale- & Lira district, Uganda - A field study evaluating problems and suitable solutions
- 355 Viktoria Björklund (2015) Water quality in rivers affected by urbanization: A Case Study in Minas Gerais, Brazil
- 356 Tara Mellquist (2015) Hållbar dagvattenhantering i Stockholms stad - En riskhanteringsanalys med avseende på långsiktig hållbarhet av Stockholms stads dagvattenhantering i urban miljö
- 357 Jenny Hansson (2015) Trafikrelaterade luftföroreningar vid förskolor – En studie om kvävedioxidhalter vid förskolor i Malmö
- 358 Laura Reinelt (2015) Modelling vegetation dynamics and carbon fluxes in a high Arctic mire
- 359 Emelie Linnéa Graham (2015) Atmospheric reactivity of cyclic ethers of relevance to biofuel combustion
- 360 Filippo Gualla (2015) Sun position and PV panels: a model to determine the best orientation
- 361 Joakim Lindberg (2015) Locating potential flood areas in an urban environment using remote sensing and GIS, case study Lund, Sweden
- 362 Georgios-Konstantinos Lagkas (2015) Analysis of NDVI variation and snowmelt around Zackenberg station, Greenland with comparison of ground data and remote sensing.
- 363 Carlos Arellano (2015) Production and Biodegradability of Dissolved Organic Carbon from Different Litter Sources
- 364 Sofia Valentin (2015) Do-It-Yourself Helium Balloon Aerial Photography - Developing a method in an agroforestry plantation, Lao PDR
- 365 Shirin Daneshpash (2015) Evaluation of Standards and Techniques for Retrieval of Geospatial Raster Data - A study for the ICOS Carbon Portal