abc

by D P

FILE	THESIS.PDF (1.63M)		
TIME SUBMITTED	24-APR-2017 10:51AM	WORD COUNT	11751
SUBMISSION ID	803762916	CHARACTER COUNT	60082

Spatial Data: Crawling, Metadata Discovery, Publishing & Query Orchestration

Punjabi Deepak Bharatbhai

Roll No. 15IT60R17

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR

WEST BENGAL, INDIA

APRIL 2017

Spatial Data: Crawling, Metadata Discovery, Publishing & Query Orchestration

Geo-Service Portal

Foundation Platform

Thesis submitted to Indian Institute of Technology Kharagpur in partial fulfillment of the requirements for the award of the degree of

Master of Technology
in
Computer Science and Engineering
by

Punjabi Deepak Bharatbhai
15IT60R17

Under the guidance of

Prof. Soumya K. Ghosh

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

APRIL 2017

©2017 Deepak Punjabi. All rights reserved.

DECLARATION

I, **Punjabi Deepak Bharatbhai**, Roll No.**15IT60R17**, registered as a student of M.Tech. program in the Department of Computer Science & Engineering, Indian Institute of Technology, Kharagpur, India (hereinafter referred to as the 'Institute') do hereby submit my thesis, title: **Spatial Data Crawling, Metadata Discovery, Publishing & Query Orchestration** (hereinafter referred to as 'my thesis') in a printed as well as in an electronic version for holding in the library of record of the Institute.

I hereby declare that:

1. The electronic version of my thesis submitted herewith on CDROM is in PDF format.

2. My thesis is my original work of which the copyright vests in me and my thesis does not infringe or violate the rights of anyone else.

3. The contents of the electronic version of my thesis submitted herewith are the same as that submitted as final hard copy of my thesis after my viva voice and adjudication of my thesis on 28−04−2017.

4. I agree to abide by the terms and conditions of the Institute Policy on Intellectual Property (hereinafter 'Policy') currently in effect, as approved by the competent authority of Institute.

5. I agree to allow the Institute to make available the abstract of my thesis in both hard copy (printed) and electronic form.

6. For the Institute's own, non-commercial, academic use I grant to the Institute the non-exclusive license to make limited copies of my thesis in whole or in part and to loan such copies at the Institute's discretion to academic persons and bodies approved of from time to time by the Institute for non-commercial academic use. All usage under this clause will be governed by the relevant fair use provisions in the Policy and by the Indian Copyright Act in force at the time of submission of the thesis.

7. Furthermore

(a) I agree to allow the Institute to place such copies of the electronic version of my thesis on the private Intra-net maintained by the Institute for its own academic community.

(b) I agree to allow the Institute to publish such copies of the electronic version of my thesis on a public access website of the Internet should it so desire.

8. That in keeping with the said Policy of the Institute I agree to assign to the Institute (or its Designee/s) according to the following categories all rights in inventions, discoveries or rights of patent and/or similar property rights derived from my thesis where my thesis has been completed:

a. with use of Institute-supported resources as defined by the Policy and revisions thereof,

b. with support, in part or whole, from a sponsored project or program, vide clause 6(m) of the Policy.

I further recognize that:

c. All rights in intellectual property described in my thesis where my work does not qualify under sub-clauses 8(a) and/or 8(b) remain with me.

9. The Institute will evaluate my thesis under clause 6(b1) of the Policy. If intellectual property described in my thesis qualifies under clause 6(b1) (ii) as Institute-owned intellectual property, the Institute will proceed for commercialization of the property under clause 6(b4) of the Policy. I agree to maintain confidentiality as per clause 6(b4) of Policy.

10. If the Institute does not wish to file a patent based on my thesis, and it is my opinion that my thesis describes patentable intellectual property to which I wish to restrict access, I agree to notify the Institute to that effect. In such a case no part of my thesis may be disclosed by the Institute to any person(s) without my written authorization for one year after the date of submission of the thesis or the period necessary for sealing the patent, whichever is earlier.

Punjabi Deepak Bharatbhai
Department of Computer Science & Engineering,
Indian Institute of Technology, Kharagpur.

CERTIFICATE

This is to certify that this thesis entitled **Spatial Data: Crawling, Metadata Discovery, Publishing & Query Orchestration**, submitted by **Punjabi Deepak Bharatbhai** to Indian Institute of Technology, Kharagpur, is a record of bonafide research work carried under my supervision and I consider it worthy of consideration for award of the degree of Master of Technology of the Institute.

Dated:

Prof. Soumya K. Ghosh
Department of Computer Science & Engineering,
Indian Institute of Technology, Kharagpur.

ACKNOWLEDGEMENT

First and foremost, I would like to express my deepest gratitude and sincere thanks to my advisor, ***Prof. Soumya K. Ghosh*** for his inspiration, encouragement and able guidance all throughout the course of my M.Tech Project. It is because of his valuable advice from time to time and constant support that today I have been able to give shape to my work. It is indeed an honour and a great privilege for me to have worked under his guidance which has made my research experience productive. I learned an approach of humanity, perseverance and patience from him.

I sincerely acknowledge my deepest gratitude towards all faculty members of Department of Computer Science & Engineering for providing in-depth knowledge on various subjects over the past two years. It was a pleasure to learn and work with their co-operation.

Dated:

Punjabi Deepak Bharatbhai
Department of Computer Science & Engineering,
Indian Institute of Technology, Kharagpur.

ABSTRACT

In the recent era of technology data is everything. One of the very useful type of data is spatial data. The information about spatial data can be found to be much useful in many fields of industry. From remote sensing, mining to doing spatio-temporal analysis to doing flood or disease spread analysis, the applications are endless. However this data is not generally publicly easy to find. Spatial data is not handled well by the the general purpose search engines. The methods to access these data are also heterogeneous and many times require licensing and using proprietary technologies. Heterogeneity of the data poses another problem of ranking and combining A central catalog service can be built to keep track of this various repositories, also the data can be made available through simple unified calling mechanisms. With the availability of large amount of heterogeneous data from different multiple sources one can do many type of spatio temporal analysis. This catalog service can then behave as central node for all available geo-spatial data available in the web. Registry can be used to publish all the information to subscribers and open web. Using this registry an efficient query processing system can be built around it to generate information needed in the user friendly mode. Query processor can find data from various heterogeneous data stores and process the data to get one result from many data sources. Various ranking and cost matrices can be deployed to make indexing of returned data feasible.

Key words: Topical Crawling, Meta-Data Discovery, Meta-Data Publishing, Spatial Ranking, Spatial Cloud Computing, Spatial Query Orchestration

Contents

Declaration **iii**

Certificate by the Supervisor **v**

Acknowledgments **vi**

Abstract **vii**

List of Figures **x**

List of Tables **xi**

1 Introduction **2**

1.1 Motivation 2

1.2 Problem Statement 3

1.3 Objectives 3

1.4 Organization of thesis 5

2 Literature Survey **6**

2.1 Spatial Data & Spatial Web Services 6

2.1.1 Classification of Spatial Data 6

2.1.2 OGC Web Services 7

2.2 Spatial Web Crawler 9

2.3 Catalog Service for Geo-Spatial Data 10

2.4 Query Orchestration 10

3 Spatial Web Crawler **12**

3.1 Objectives 12

3.2 Architecture of the Spatial Web Crawler 13

3.3 Algorithm 16

3.4 Example Scenario 17

3.5 Advantages of Spatial Web Crawler 18

4 Spatial Metadata Discovery & Publishing **19**

4.1 Objectives 19

4.2 Architecture of Catalog Service 19

4.3 Implementation 21

4.3.1 Crawler Module 22

4.3.2 Database Setup 22

4.3.3 Catalog Service 23

4.3.4 Query Processing 24

4.4 Extending Implementation to Cloud 24

4.5 Results 26

5 Spatial Query Orchestration **32**

5.1 Quadtree based Indexing 32

5.1.1 Algorithm 33

5.1.2 Example Scenario 33

5.2 Ranking based on Quality Preferences 35

5.2.1 Algorithm 36

5.2.2 Example Scenario 37

6 Conclusion and Future Work **39**

6.1 Contribution of thesis 39

6.2 Future Scope 40

Bibliography **42**

List of Figures

1.1 3 - staged pipeline approach 4

3.1 Spatial web crawler architecture 13

3.2 XML response from the geo server 18

4.1 Architecture of Catalog Server 20

4.2 List of relations 23

4.3 Catalog Service as a mediator 24

4.4 load balancer implementation 25

4.5 load balancer preliminary results 26

4.6 GeoServer Implementation 27

4.7 Map of KGP BNK ROAD 29

4.8 Overlay of KGP BNK ROAD, Block HQ & Boundary Layers 30

5.1 arrangement of feature points 34

5.2 Ranking by quality preferences 37

5.3 Features ranked by neighbourhood quality 38

List of Tables

5.1 Output with overlap region 1 35
5.2 Output with overlap region 2 35
5.3 Final ranking for the features 38

List of Algorithms

1 *Extraction Algorithm* 16

2 *WMS Resolver Algorithm* 16

3 *Analysis & Indexing Algorithm* 17

4 *Quadtree Indexing Algorithm* 33

5 *Ranking based on quality preferences* 36

Spatial Data: Crawling, Metadata Discovery, Publishing & Query Orchestration

Chapter 1

Introduction

As the information around us grow exponentially, spatial context about such information is becoming more and more important and useful. For example, If one wants to go from one place to other, He or she always chooses the path that is spatially shortest. Another good spatially aware example is when a person wants to buy a house. Many factors affecting the decision to buy a house are inherently spatially aware, like how far is the hospital, how far is the school facility or is there any good restaurants available nearby. All these examples suggests that one of the crucial factor while buying a house is how qualitative is the spatial neighbourhood. Or we can say that many decisions occurring in everyday life of a person are spatially aware.

It can be said that spatial data is another dimension for organizing information. With this, getting and organizing spatial data for yielding useful information becomes more and more useful. This creates a desired motivation for building such system to effectively collect and utilize spatial data.

1.1 Motivation

Currently available search engines are good for normal data such as text and web links, but when it comes to searching and ranking spatial data, none of the currently available search engines can provide efficient results. One of the reason for this can be that not all Geo-spatial data is publicly available. Other problems is that repositories containing spatial data are not generally indexed and the meta-data information about what data they contain is not easily available. Also the normally crawled data is not kept in a spatially aware manner. Our motivation here is to build a registry service that can crawl spatial data from the open web, avail data from national government agencies as well as

open source free repositories. Once the data is available various kinds of tasks can be built on top of that. This behaves as a foundation platform to provide basic and robust services upon which various other services can be built. We call this as Geo-Service Portal. One can easily build and perform various type of spatial queries like GetMap on the data available. Various graphical applications as well as application programming interfaces(APIs) can be provided on the available data to cater in use in various ways. A query interface can be built upon the data available from catalog service. This query interfaces can provide various views for user specific user queries. An spatial-temporal analytics can also be done using the catalog. Change of data can be tracked. Other use cases can be finding the spread of diseases or getting the area affected by flood. Many other applications can be generated.

1.2 Problem Statement

The aim of this project is to create a foundation platform for crawling, storing, maintaining and publishing meta-data information about spatial data and corresponding providers and to utilize this information to perform efficient query orchestration.

1.3 Objectives

1. **Build a topical crawler to crawl the web and store Geo-spatial metadata.**

   A topical crawler is a special type of crawler which only searches for information of a particular topic interest. Here the topical crawler crawls the open web, finds and filters URLs found by it. This module checks whether the server provided by the URL is capable of providing OGC compliant geo-spatial web services or not. If the URL is found to be a geo-server it stores all the metadata information about the Geo-server and about the data it contains into the database. It uses domain ontology knowledge to classify data into different domains. This information then later can be utilized to query and find useful information.

2. **Build an OGC compliant catalog service to publish and search accumulated metadata.**

   The aim here is to build a web service that can query and publish relevant metadata information from the database. The catalog service is real-time. The data stored in the catalog changes with time to reflect changes in corresponding data

from the providers. This can also be explored to examine spatio-temporal nature of the data. The catalog service should be built in such a way that queries can be processed in real time. For this purpose the data should be stored in database in structured manner rather than storing it in plain json or XML files. The responses provided by the spatial repositories are generally in XML, GML or shape format, catalog service parses the data to store data in a structured way in database so that various query operations can be supported in OGC compliant manner. One other crucial point for catalog service is to provide high availability. Registry is used to search as well as publish the available data.

Figure 1.1: 3 - staged pipeline approach

3. **Build Query orchestration service to perform real-time query with heterogeneous data sources and cost matrices associated with them.**

   One of the tasks of catalog service is to provide relevant information to the querying machine or person. The data that is stored by different repositories can be of heterogeneous form. Task of query orchestration is to find relevant information across all available data sources and perform the query using all relevant data. If the data is available from multiple data sources different type of ranking and cost matrices can be associated to provide most relevant information to the user. Query orchestration is also associated with ranking. While retrieving the data it should be presented to user in a ranked retrieval manner based on some metrics. Here we have discussed two techniques of ranked retrieval and indexing called Quad tree based indexing and ranking based on quality preferences.

1.4 Organization of thesis

This thesis is mainly organized into six chapters. The first chapter gives introduction about the subject matter, it clears the motivation and advantages of the project. It defines the problem statement and states as well as explains the objectives of the project. The second chapter namely literature survey, provides already available data about the subject and proposed solution methods. This section first explains about the data for the model to be work with. It explains the web services and APIs build for spatial data. Then it explains about three part of the foundation platform, Spatial web crawler, Catalog service and query orchestration. Chapter 2 also discusses about the previous work done in the respective fields.

Chapter 3, 4 and 5 are the core foundation blocks for Geo-service portal foundation platform. Chapter 3 is explains about the spatial web crawler. It explains the architecture and the algorithm used for the spatial web crawler. It also explains advantages of spatial web crawler. Chapter 4 contains information about spatial metadata discovery and publishing. It discusses objectives for the catalog service. It also give architecture and implementation details about catalog service for web and query processing. It also shows example results. It also extends the concept to cloud based paradigms.Chapter 5 gives a look into spatial query orchestration. It discusses about two broad strategies two index and rank spatial data.

The final chapter concludes the morals of the thesis and creates a pathway for better optimizing and extending the proposed solution for the future work.

Chapter 2

Literature Survey

The literature for this project can be divided mainly into four parts. The spatial data and spatial web services on which the framework is developed. Crawling: How to find that data. Catalog Service: Once the data is found, how to store that data. How to make data easily accessible to others. And at the last Query orchestration: how to perform simple queries on the available data.

2.1 Spatial Data & Spatial Web Services

The term Geo comes from geography. Geography stores all the information of location and shape of the object in the spatial data. Spatial data stores the relationships between these data. It can also be easily mapped to a map. Spatial data is stored as raster data or vector data. Geo-server provides various kinds of functionality to this type of data. Spatial data is the data that can be mapped.Spatial data is collection of spatial object that has topology and co-ordinates. Spatial data gives geographical and shape related attributes of the data.Geographical information system is used to retrieve and operate on spatial data. Different operations can be add, visualize, annotate etc. A Geographical Information System provides various extended operation on spatial data. Different examples of such software that offers spatial services are ESRI, Microsoft SQL Server, ERDAS imagine etc.

2.1.1 Classification of Spatial Data

Open Geo-spatial Consortium(OGC) defines Different types of object under the class geometry are as below. All the major Geo-spatial service providers and vendors provide this kind classification.The primitive four data types in spatial data are point, curve, sur-

face and GeomCollection.

- **Point**
  Point in a map is denoted by (x, y) co-ordinates. When we see kharagpur city on a scale of India it will be seen as a point. Point can be used to denote various objects like origin, city, end point, top of the mountain etc. It denotes a single point consisting of longitude and latitude.

- **Curve**
  Curve is used to denote collection of points. This can be a straight line or curve. For example, a road network can be represented with the help of line strings. Similarly, a river can be denoted as a curve. Curve can be made with help of two distinct points.

- **Surface**
  A surface is representation of an area or a polygon. When kharagpur is seen in the scale of west Bengal it is seen as surface or polygon. Polygon can be made using at least three non-linear points. Every polygon has a feature called boundary.

- **GeomCollection**
  Collection of basic building blocks defining a new type of geometry can be defined with the help of GeomCollection. GeomCollection is made with combining two or more geometry types.

2.1.2 OGC Web Services

Open Geospatial Consortium (OGC) is the worldwide standardization body for geospatial standards. OGC provides a standardized way of accessing this geospatial data. OGC mainly provides three kinds of web services.

- **Web Map Service (WMS)**
  Web Map service defines a way of accessing geospatial information across all geo servers in a standard format as image. This image can be raster image or a

vector image. Raster images are of type jpg, png or bmp. Vector images contains svg format extension images. It also provides a way to access metadata about the available information of the layers. This information can contain type and no of layers.

Some of the well-defined operations in this layer are GetCapabilities, GetMap and DescribeLayer. GetCapabilities operation returns capabilities of the server for example, what kind of services the server offers. This operation may be useful to check if a server is geo server or not by examining the kind of services it offers. GetMap service returns map images for the queried data. multiple such layer of images can be queried and then can be overlaid on top of each other. for example satellite view, terrain view, traffic view etc. Each of this layer gives additional information to the map. DescribeLayer operation describes what are the available layers and what is the metadata about the layers.

- **Web Feature Service (WFS)**

  WFS allows direct access to features contained in the map. WFS uses SOAP based interface. SOAP is used to minimize the overhead into the communication and verify cross platform stability. For exchanging data between client and server WFS uses Geographical mark-up language(GML) which is based on XML. Some well-defined operations in WFS are query or get feature, which returns the feature stored on the server. We can add the feature in the repository by add feature. We can delete feature by delete feature. Also we can update feature stored in the repository by update feature. We can also lock certain feature to disallow modification of it while sharing via lock feature. Locked resources and features wont be accessible to change to other clients. Example of feature are the objects in the map like building or petrol pump.

- **Web Coverage Service (WCS)**

  WCS offers multi-dimensional coverage of the geo spatial data.It can provide originally retrieved data or provide processed data. It provides spatio-temporal context to the given geographical data. For example, it can show the flow of the river changing over the span of years. Thus we can say that WCS provides richer coverage of spatial data than WFS or WMS. Coverage data is mainly used for scientific calculations.

2.2 Spatial Web Crawler

Wenwen Li[2] says that if a server is offering geo-spatial capabilities then it can be either of the WMS, WFS or WCS server. The geo-servers must comply to OGC standards for geospatial data. This can be checked using GetCapabilities service request. We can check whether a given server offers WMS offerings or not can be checked using special suffix string added to the URL of a geo-server. More details are covered in the geo-spatial crawler part below. If the server is indeed a WMS server, It replies with and XML files containing all the data and operations it can offer. We call this as services.xml file. Once We know that the server is a geoserver many kinds of operations can be performed on it. WMS geoserver offer various operations such as GetMap, DescribeLayer, GetCapabilities etc. WMS capabilities file contains all these operations mentioned above. This is to check if a particular server is geoserver or not. If we want to crawl through the open web, then we must repeat this process. For this purpose standard operation of crawlers are performed. We maintain a buffer of frontiers which divides the boundaries of crawled web and open un-crawled web. Wenwen Li[2] also suggests to make a auto updating crawler to periodically check whether there is any addition or removal or geoserver from the existing available data. This geoserver(s) also add new data to them and remove obsolete data from the data store. Automatic updates allows this operations to be done periodically and maintain the overall consistency of the data.

Marc Najork[3] suggests that the task of crawling the web for WMS servers can be done parallel. He suggests to use parallel FIFO queues to carry out the operation efficiently. To check whether the URL already exists or not in the crawled set, he suggests to use efficient data structures to check set membership such as HashSet. He also suggests to use priority in crawling based on various methods such as PageRank.

Lopez-Pellicer[4] suggests that there might be already available geoservers and catalog services on the web. We must efficiently identify this catalog services to different type of OGC compliant services. So that different kind of queries can be forwarded to different geo servers or catalog services. Dirk Ahlers[5] suggests an architecture to crawl and parse available geoserver(s) and store the available data efficiently. They also suggests to index the data so there data can be retrieved efficiently. This methods is described as focused crawling.

Li, W., et al.[6] suggest a method to semantically crawls the web. In his approach he maintains a hierarchy of geospatial objects to identify parent children relationship. For

example, river is a type of water-body. This kind of type of relationships can be used to manage and index available data and geoservers efficiently. JIANG Jun[7] suggests similar methods for WFS servers.

2.3 Catalog Service for Geo-Spatial Data

In this section we will look at two types of available catalog services for geospatial data. Python catalog service for web(PyCSW)[8] offers a model architecture for implementing a OGC compliant web service catalog portal. In this model, series of module are used to invoke a hierarchy such that decomposition of the functions and use case is efficient and easy. In this architecture many of the modules can work in parallel. Important modules are crawler, WMS capabilities files, Database to store geospatial data and a server to provide interface to functions. All the functions are implemented and run via programs on a server. This wsgi server acts as a geospatial catalog which accumulates data from various other geospatial repositories and catalogs available on the open web. Using the PyCSW interface many types of applications can be invoked. PyCSW provides various APIs for OGC compliant operations. For example, using GetMap a user or application can get all the data from all the crawled repositories in a single place. We can know the information available for the layers and from where this information is available. The detailed implementation of this architecture is discussed in the later chapters.

GeoServer[10] also offers such services but it also offers powerful management tools and admin console to function via graphical user interface. It provides services to work with various data formats such as shape files or GML format, it supports various database options for storage purpose and it is extensible to add more services. In our approach, we will mainly use GeoServer as our catalog service. However it should be noted that GeoServer is more resource intensive than PyCSW as it provides more services than the latter. It provides integration with various programming languages like Python, Java, Ruby and PHP.

2.4 Query Orchestration

Query Orchestration provides query operations on the spatial data from the database. Various operations are available as OGC standards to operate and query on the data. Query orchestration also deals with indexing and ranking. Here we have used Quad

tree for indexing spatial data. Quad tree[13] is particularly standard data structure for indexing spatial data. Another interesting concept is ranking by quality preferences[11]. In this concept we take the cumulative property of spatial neighbourhood to calculate ranking for the spatial objects.

Once the catalog service is built it can directly act as a single point for querying multiple repositories. The catalog server in-fact acts as a mediator while getting the queried data from other repositories. Different type of example queries are listed below.

- **Service metadata information**
  Describes what kind of OGC compliant web services are offered by the catalog server or geo-server.
- **GetLayers**
  An interface can be built to provide information about list of available layers.
- **GetOperations**
  List of available operations can be provided that gives the information about the list of operations that are offered by the geoserver or catalog service.
- **GetBoundingBox**
  We can get the information about the bounding box of the layers or map that covers the total area.
- **GetMap**
  The map of particular layer can be retrieved.
- **DrawMap**
  The map of particular layer/data can be retrieved and placed onto another map. This is useful to find area of interest in the particular map by superimposing the queried data to generic map.

Chapter 3

Spatial Web Crawler

Our first step to building the platform Geo-Service portal is to gather the required spatial data. This can be done by finding the available data sources for geo-registries from the web via crawling or adding the sources manually. Spatial data is not indexed and available as compared to normal image or text data. The crawler module crawls the open web to find OGC compliant geoservers and stores their metadata information. It uses ontology or hierarchical mapping or spatial data to classify the crawled features. This features are permanently mapped into ontology to help in query processing.

3.1 Objectives

1. Build a spatial web crawler which crawlers through available geo-servers which offers WFS based OGC compliant services.

2. Build a domain specific vocabulary(ontology) for this features which can be helpful to compare found features with wanted features.

3. Perform semantic matching of found features from crawled web-pages with given ontology for filtering the correct features and storing them in the permanent repository.

4. Perform an evaluation of the given spatial web crawler using metrics and test URL seed sets.

3.2 Architecture of the Spatial Web Crawler

Our crawler contains three modules for crawling spatial features through the world wide web. Some of the definition needed for understanding the working of spatial web crawler are:

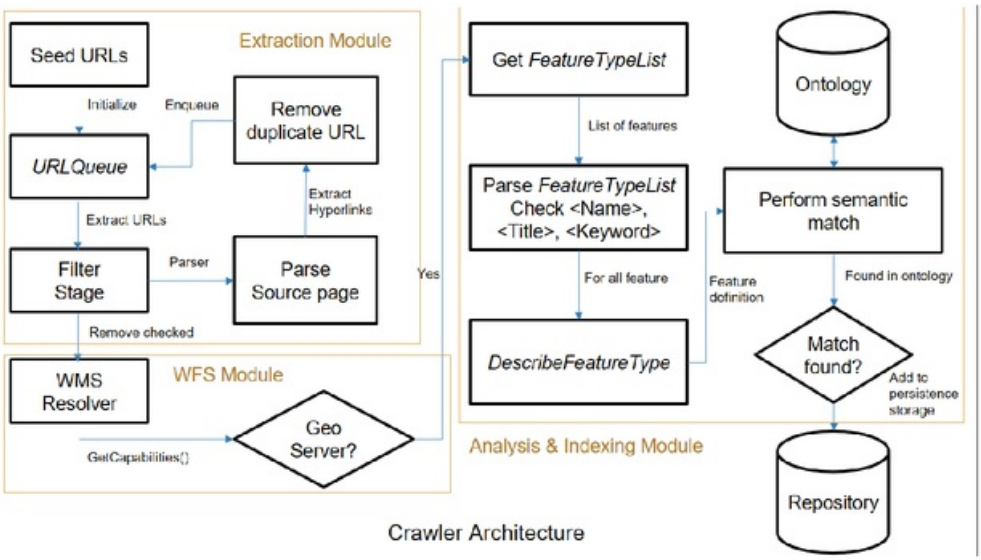

Figure 3.1: Spatial web crawler architecture

- **Seed set**
  A set of test URL(s) to initialize the queue for crawling the web. The crawler starts with the URLs given in the seed URL set. Seed set URL(s) are processed and parsed to obtain other URL(s), which are added to the queue for later crawling.

- **URLQueue**
  A queue that contains set of URLs to be crawled. The crawler module takes URLs one after the another from this queue. It contains list of all the valid URL(s) to be crawled. It contains seed set initially. URL(s) are taken one by one by the crawler module for processing and deleted from the URLQueue. Similarly, newly found URL(s) are added to the queue.

- **Frontiers**
  A set of URLs from the URLQueue that are currently being crawled. This are

called frontiers because they reside between the known web and the unknown web. This are active URL(s) currently participating in the process of crawling. Frontiers are checked by the WMS resolver module for their geo-capabilities and parsed by extraction module to get new set of URL(s) to add in URLQueue.

- **WMS resolver**
  WMS resolver is a module that checks that if a server is WMS server or not given the URL from the URLQueue. It uses special request URL to check whether the server replies to the URL in an OGC compliant manner.

- **Parser**
  Parser is a module that downloads the webpage from the given URL and parse the HTML webpage looking for pre-specified tags and names. After parsing it gets a set of tags and its contents. In our implementation we parse the webpage and look for the anchor tags within it and store all the hyperlinks from the crawled webpage. This links are then stored in the URLQueue for further processing.

- **Ontology**
  Ontology is a semantic dictionary containing all the features its type and relationship hierarchy between features. This is used while mapping features to it's corresponding super-type and while query processing that contains more than one feature type. For example, getting overlap or intersection between two feature types.

Spatial Web Crawler is designed to contain three modules. This modules are namely Extraction module, WFS module and Analysis & Extraction module. They are designed in a manner that they can work independently, which will come useful while implementing this as a extension in cloud based environments. :

- **Extraction module**
  Our algorithm starts with a set of seed URLs contained in the seed set. We initialize the URLQueue with these seed URLs. Initialization is done manually. It is required to choose these seed set URL(s) carefully, to get better results in crawling. Ideally, seed URLs should be good authorities, they should point to large number of good geoserver(s). Depending on the seed set, the crawler module may find more and less no of good spatial repositories in less or more amount of time. The crawler takes URL from the URLQueue and feeds it into filter stage. The filter stage takes the URL and checks whether the given URL is already crawled or not. After filtering such URLs, they are sent to the parser. The parser downloads the

web-page from the given URL and parses it for finding hyperlinks contained in it. These hyperlinks again go to filter stage for finding whether the given URLs are already crawled or not. After filtering these URLs are added to the end of URLQueue. The filtered URLs are also passed to the WFS module.

- **WFS module**
  Once the URL is fed into WFS module for the examination, WFS module sends a GetCapabilities() request to that server. We do this by appending the request to the URL.

$$?service = wfs\&version = 1.1.0\&request = GetCapabilities$$

The server replies for this request. If the reply contains WFS_Capabilities tag, then it offers WFS service. We parse the received response for finding the WFS_Capabilities tag. The server replies in XML format usually called capabilities.xml file, this files contains all metadata information about the feature services offered by the WFS server like number of layers, type of operations supported, feature description, etc. If the given server is WFS server then the given URL is passed to analysis and indexing module.

- **Analysis and Extraction module**
  In this stage server response is parsed for the tag FeatureTypeList. FeatureTypeList contains list of features. This features are stored under the tag FeatureType. FeatureType tag contains set of keyword, title, name tags. Each of this tags are checked to see if it contains any word from the ontology. For each of such tag found, DescribeFeatureType request is appended to the URL.

$$?service = WFS\&version = 1.1.0\&request = DescribeFeatureType\&typename = " + keyword$$

Here the keyword is the name of the feature. Each of this retrieved feature is checked again the ontology, if similar feature is found in the ontology then it is added to permanent storage in the repository. Matching is done semantically via ontology.

3.3 Algorithm

Algorithm 1 *Extraction Algorithm*

Input: set of URLs to start crawling

Output: set of WMS capable URLs

```
Initialize:
  for URL in Input do
    URLQueue.insert(URL)
  end for
  for URL in URLQueue do
    if ! filter(URL) then
      preUrlSet ← parser(URL)
      urlSet ← filterDuplicate(preUrlSet)
      URLQueue.insert(urlSet)
    end if
    WMSresolver(URL)
  end for
```

Algorithm 2 *WMS Resolver Algorithm*

Input: URL to check for OGC compliant services

Output: WMS capabilities

```
Response ← GetCapabilities(URL)
if Response contains WMS_Capabilities then
  Analyze(URL)
end if
```

Algorithm 3 *Analysis & Indexing Algorithm*

Input: WFS capable URL

Output: WFS records

```
FeatureTypeList ← GetFeatureTypeList
for featureinFeatureTypeList do
    name ← feature.name
    description ← DescribeFeatureTypeList(feature)
    if (name, description) matches in ontology then
        Store record in repository
    end if
end for
```

3.4 Example Scenario

Suppose we search for a example query after building the system. Let this query be *river and snow*. Now if we search this on normal search engines like Google, it will show us results containing either river or snow. But as we have information about spatial context in this crawler. Repositories have stored the information about which features are in near proximity to each other.

Other than this, we have a object ontology. Which is a hierarchical graph based on the spatial features. This graph contains generalized and specialized features. For example it suggests that river comes under the type stream which intern comes under the geometry type waterbody. This type of hierarchical classification helps us to understand the geospatial data and their features and relationships correctly. Our query is river and snow. The crawler searches the repository for finding both features river and snow or their similar representation by doing a semantic matching over ontology. Once found it can see which is the geo server which offers both of the geo services and feature types. Based on its finding from the repository we conclude our results and send the reply back to the user.

```xml
<?xml version="1.0"?>
- <root>
  - <FeatureType>
      <Name>prov_land</Name>
      <Title>Canadian Land</Title>
      <SRS>EPSG:42304</SRS>
      <LatLongBoundingBox maxy="83.8009" maxx="-11.9603" miny="35.8775" minx="-173.537"/>
    </FeatureType>
    -
  - <FeatureType>
      <Name>land_fn</Name>
      <Title>US Land</Title>
      <SRS>EPSG:42304</SRS>
      <LatLongBoundingBox maxy="89.8254" maxx="179.94" miny="31.8844" minx="-178.838"/>
    </FeatureType>
</root>
```


Figure 3.2: XML response from the geo server

3.5 Advantages of Spatial Web Crawler

- Allows search in web pages that are not generally searchable from the normal search engines. This is because of the spatial context awareness of the spatial web crawler.
- It provides more up-to-date results from the results. Spatial crawler is more sensitive to changes of spatial data on the web and automatically crawls through the changed features with the help of automatic update module.
- Provides improved accuracy in the search of spatial features and operations.
- Provides extra features such as bounding box of spatially crawled data and other spatial features. This bounding box can then be used for visualizing the crawled data.
- Spatial data mining and analysis kind of tasks can be performed. So that we can infer and predict new results and patterns.

Chapter 4

Spatial Metadata Discovery & Publishing

Once the geo-servers are crawled from the open web, we need to organize this data into a well organized manner so that it can be retrieved efficiently. The operation and query we perform on this data must be performed in a efficient manner. The aim of Spatial metadata catalog server is to make crawled spatial metadata available for public.

4.1 Objectives

1. Parse the crawled metadata and store it into a permanent database in a structured manner.
2. Publish metadata repository in OGC compliant manner to make it available as a service.

4.2 Architecture of Catalog Service

There are various data repositories available on the web. However this repositories are not indexed. Spatial web crawler crawls through open web and stores corresponding metadata information about available data into the xml files. Crawler uses *GetCapabilities()* OGC request to find if a given server is geoserver or not, if the given URL is indeed a geo-server then crawler module requests to retrieve *capabilities.xml* file from the server and stores it into warehouse. XML files contains the services information provided by the geo-servers found on the web. They are usually called services.xml.

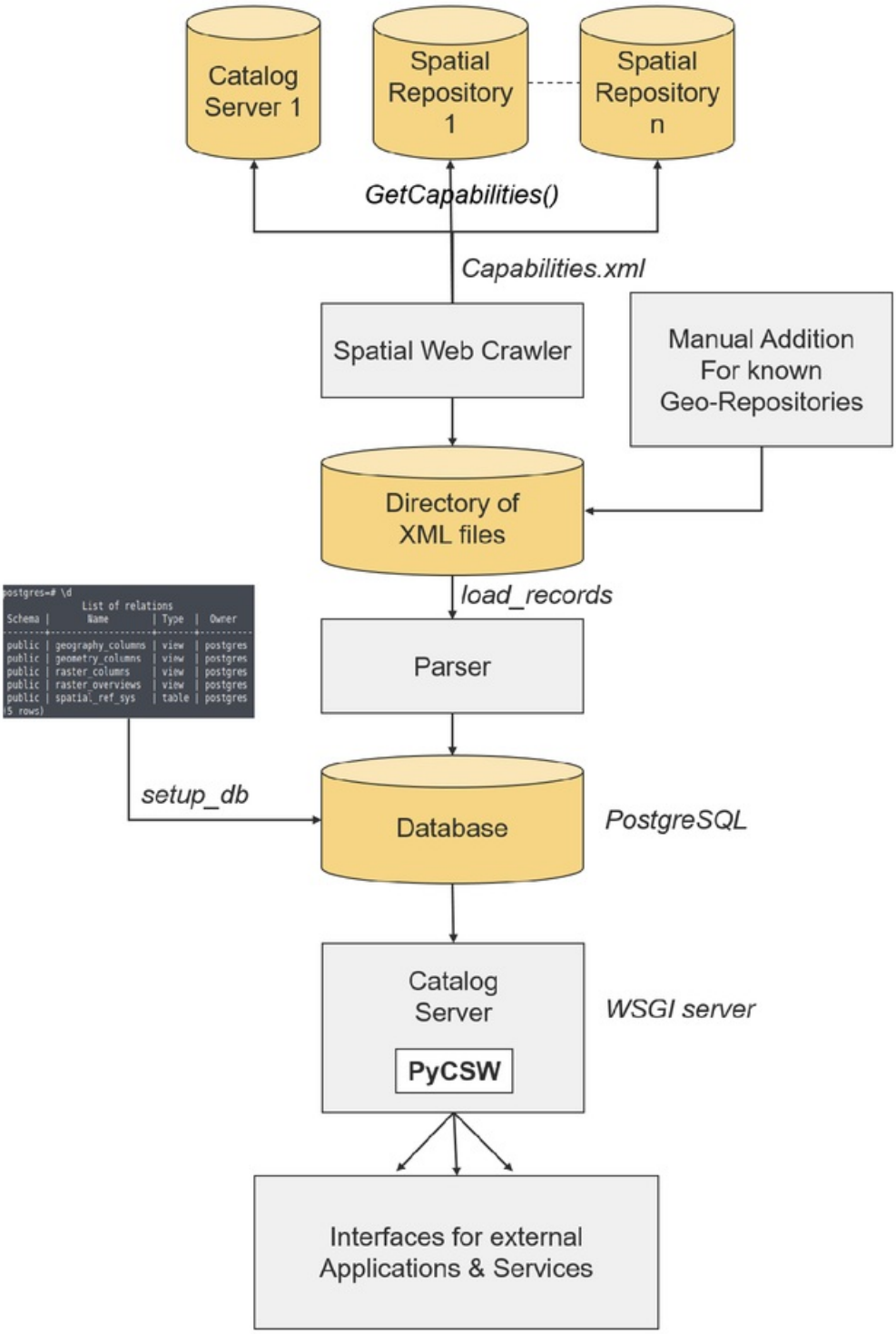

Figure 4.1: Architecture of Catalog Server

This information contains available operations, available layers and other useful information. This xml files are gathered in a folder containing xml files.

No of spatial repositories in the web is significantly lower than other type of services and servers. It can happen that it might take a significant amount of time to gather practically good number of geo-servers. For this reason, this files can be added by the spatial web crawler or it can be added manually.

This xml files are then loaded into the database for faster and efficient retrieval operation. For database we can use either of MySQL, PostgreSQL or SQLite. But MySQL and SQLite does not offer inherent data structures and operations support for spatial data. SQLite is also meant for light weight databases, whereas here the no of request-reply and data transfer can be very high. For these reasons it is better to choose PostgreSQL database system to store data for the catalog service. PostgreSQL offers high scalability and offers inherent support of spatial data types and operations.

Catalog Service is hosted on *Web server gateway interface (WSGI)* server. We can also use other deployment servers like Apache. Main reason for choice of server as WSGI is that, WSGI integrates well with python. It is build on python. It is useful for request-reply type ordinary web interfaces and it is efficient for file transferring on other protocols. A python program admin.py is hosted on WSGI server, which takes the metadata information from the database and replies in response to GetCapabilities() request. When working on a particular request for a repository like GetMap(), the catalog service acts as a mediator between client and the repository on which the data is hosted. It converts the request taken by WSGI server converts it into standard OGC compliant WMS or WFS call, and accesses data from original repository from where data is hosted.

4.3 Implementation

There are two types of implementation done to implement catalog service module. First implementation covers *PyCSW* python catalog service for web and second implementation makes use of *GeoServer* open source tool for sharing spatial data. Difference between two type of tools is while PyCSW offer great functionalities and most of the OGC compliant services, GeoServer also offers more data type support to load repository from and a management console for admin to maintain Geo-registry. We will take

a look at both type of solutions here.

4.3.1 Crawler Module

In first stage implementation, we first crawl through open web using the spatial web crawler discussed in the above chapter. The crawler checks if the given server provides geospatial services or not. If the server is a geospatial repository then it collects all the metadata information about the repository and stores it in a form of a xml files. Standard OGC compliant service calls are used for collecting metadata information about found registry. Here in the figure we can see there are three repositories which have been crawled by the crawler. The information about these repositories are stored in a xml files called services.xml. The no of geoservers and registries are very less in the open web compared to other kind of services, for this reason it might take a long time to crawl and find significant amount of registry services for our catalog service. To handle this problem, manual addition option is used. We can manually add popular geoservers and registry services to the services.xml files. We know that geoservers reply in xml response when called for a *GetCapabilities* request. We can use this property and manually add popular geo-registries beforehand to the set of xml files. So in the summary, the first stage crawls the web and stores corresponding set of services files into one place.

For this, python based crawler module program is used. It takes a set of seed URLs, checks if the URL can built object of OGC compliant WMS or WFS object. If it can build OGC WMS object then, crawler requests for capabilities file, and stores it in set of xml files.

4.3.2 Database Setup

In second step, These set of xml files are used to populate database. Here various type of databases can be used. I have tried databases like MySQL, SQLite and PostgreSQL. In our approach I have used PostgreSQL because it offers geospatial properties and services inherently. Other databases do not offer inbuilt services for spatial data. PostgreSQL database offers PostGIS extension, which are a set of libraries to extend PostgreSQL database into spatial database. For adding PostGIS to PostgreSQL, go to PostgreSQL shell psql and create a extension, PostGIS.

CREATE EXTENSION POSTGIS;

Once this is done, import all of the services,xml files into the database. Create structured tables and schema for maintaining spatial data. For this purpose *setup_db* command is used from catalog service module.

```
postgres=# \d
                List of relations
 Schema |        Name        | Type  |  Owner
--------+--------------------+-------+----------
 public | geography_columns  | view  | postgres
 public | geometry_columns   | view  | postgres
 public | raster_columns     | view  | postgres
 public | raster_overviews   | view  | postgres
 public | spatial_ref_sys    | table | postgres
(5 rows)
```

Figure 4.2: List of relations

Here four tables are built to store and index spatial data efficiently. Those are, $geometry_columns, records$, $spatial_ref_sys$ and $spatial_ref_sys_srid_seq$. This tables are then used by the catalog server to publish and query available data. When a external entity queries catalog server for Get Capabilities request, the catalog server replies with the services file which contains all the information about all the geo-servers it has crawled till now. Thus it contains all the layer information and data crawled from the Internet. This makes it a single point of resource access. Data from xml files is parsed and stored into respective relations. For this *load_records* command is used from admin.py module within PyCSW. These tables are used to store parsed spatial features from the spatial repositories.

4.3.3 Catalog Service

Catalog service takes all available data from the database and creates a metadata capabilities file. It acts as a mediator between repositories which actually contains the data and the client. When a client requests for a query or a metadata service, catalog service checks if the database contains the information. If metadata information is requested, it replied directly to the client with the necessary files or response.

In case of query requiring actual data instead of the metadata, the catalog service resolves the source from which the data is available and brings the data back to the requester client. In this case, the catalog service acts as a mediator between client and the data source, acting just as a catalog. Here the catalog service is implemented via

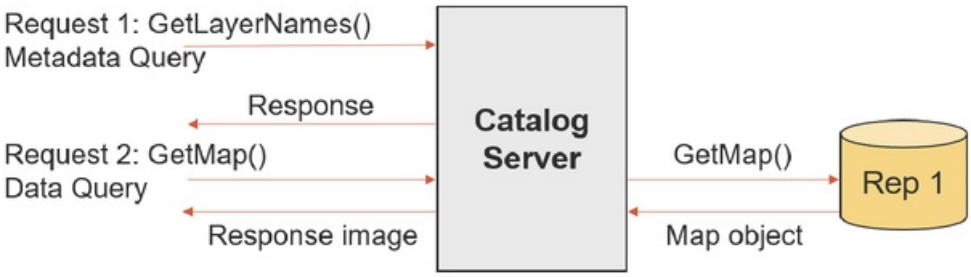

Figure 4.3: Catalog Service as a mediator

GeoServer. GeoServer takes the crawled layers from web map service and loads it into catalog.

4.3.4 Query Processing

When a query or request for the data is processed by the catalog service, it acts as a mediator between client and the repository from which the data is intended. It creates a object of the required service and requests for the object on behalf of the client. Client authorizes to the catalog service and catalog service authorizes itself to the data repositories for the transfer of data. Catalog service takes the object response from the repository and send it in response to client in client specified format. For metadata queries the catalog service retrieves data from database via object relational mappings in programming paradigms, this helps when having a multiple references to same object or feature. This method is used for retrieval of the features. How the featured are retrieved and presented are discussed further in the chapter 5.

4.4 Extending Implementation to Cloud

A key logical step to scale catalog service for the larger audience is to make a cloud based implementation. Cloud based implementation has characteristics like scaling horizontally, high availability and dynamic dispatch of machines. To achieve the effects of scalability and high availability, we can make use of *load_balancer*, *load_balancer* is a machine that acts as a middleware between client request and actual catalog servers. It forwards the request to one of the servers based on some qualitative features.

In our approach, we have started by building a *load_balancer* which balances the requests between multiple available given geoservers. These geoservers are cloned to

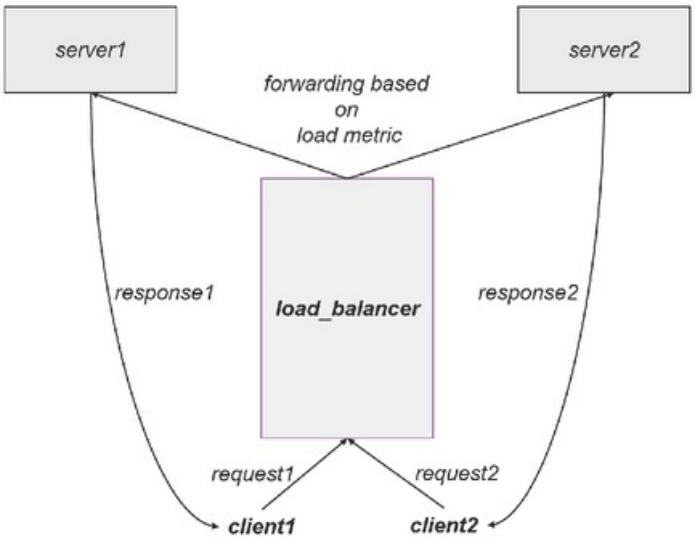

Figure 4.4: load balancer implementation

provide similar functionality but high availability. *load_balancer* detects which server has low number of requests compared to others and forwards the request to particular server. Actual servers are transparent to the end users. Interface between client and the geoservers is the *load_balancer*. It acts as a gateway for connecting with the geoservers. In our test implementation, we have created two instances of geoservers and run similar repositories on each of them. The client connects to the *load_balancer* which acts as a gateway for the actual geoservers. It gets the reply back from the actual servers and sends it to requesting clients. *load_balancer* can also restrict or allow certain users grant on certain function and not all of them, thus here *load_balancer* also acts as a control panel and authorization entity for the access control.

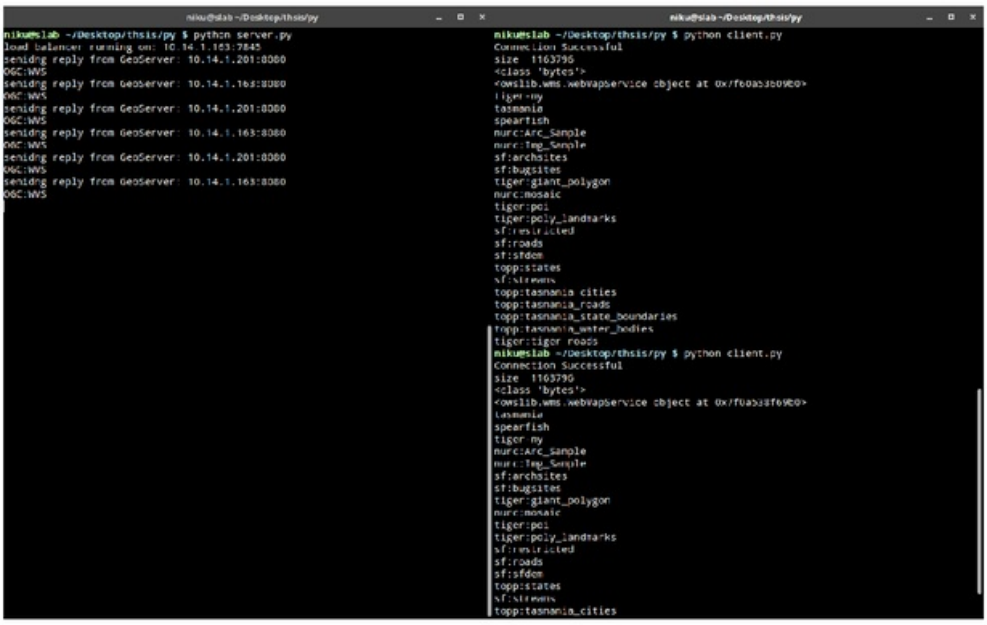

Figure 4.5: load balancer preliminary results

4.5 Results

Here are some results from the projects that explains and elaborates what kind of information can be get from catalog service. We have connected to GeoServer instance that is locally hosted. Note that many more richer services can be added to the platform to extend it's functionality.

- Information about service metadata : OGC:WMS
  This shows that registry is capable of providing OGC compliant web map service type of service to the user.

- Title : GeoServer Web Map Service
  Title gives information about title of the server hosting web map service.

- List of available layers

  1. kgp:POPULATION
  2. kgp:bnk_block_boundary
  3. kgp:bnk_block_hq

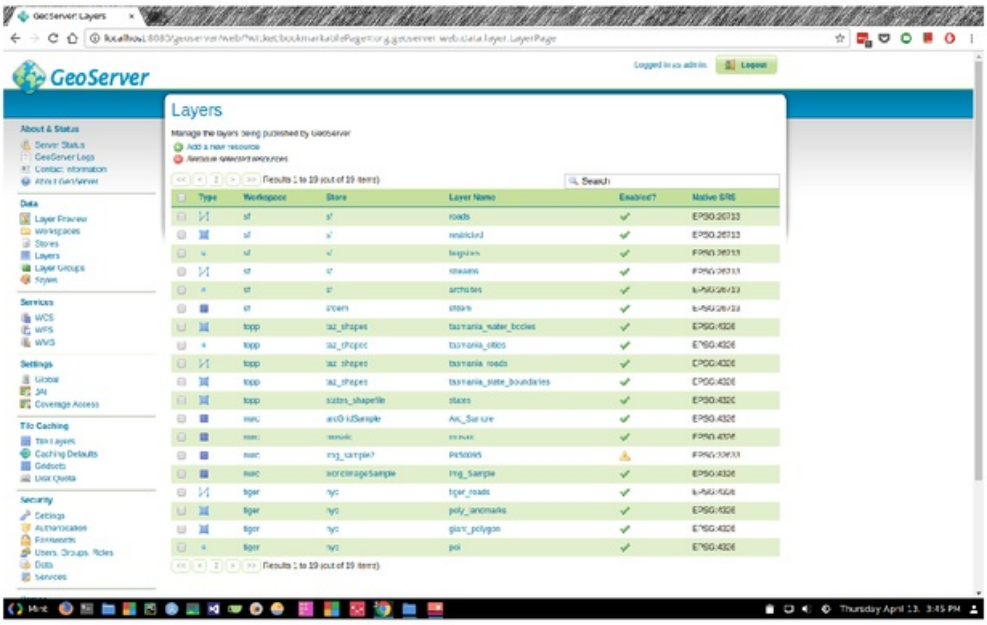

Figure 4.6: GeoServer Implementation

4. kgp:bnk_district_boundary
5. kgp:bnk_drainage
6. kgp:bnk_grampanchayat_boundary
7. kgp:bnk_mouza_boundary
8. kgp:bnk_road
9. tasmania
10. spearfish
11. tigerny
12. nurc:Arc_Sample
13. nurc:Img_Sample
14. sf:archsites
15. sf:bugsites
16. tiger:giant_polygon
17. nurc:mosaic
18. tiger:poi

19. tiger:poly_landmarks
20. sf:restricted
21. sf:roads
22. sf:sfdem
23. topp:states
24. sf:streams
25. topp:tasmania_cities
26. topp:tasmania_roads
27. topp:tasmania_state_boundaries
28. topp:tasmania_water_bodies
29. tiger:tiger_roads

This shows the list of available data in the form of layers. Each layer represents data of a geographic location from a view point. Multiple layers can be overlapped on each other to better understand the data. Each layer contains multiple features in itself.

- Available operations (WMS)

  1. GetCapabilities
  2. GetMap
  3. GetFeatureInfo
  4. DescribeLayer
  5. GetLegendGraphic
  6. GetStyles

  Gives details about the operations that can be performed by the geoserver. For example, DescribeLayer provides information about a particular layer. GetMap returns a map of layer in multiple available formats.

- GetMap

Figure 4.7: Map of KGP BNK ROAD

Returns a map of particular layer. Can be superimposed to another map for better visualization. GetMap supports multiple options like name of layer(s), styles, transparency, bounding box, size and format like jpeg, png or pdf.

- DrawMap
  DrawMap overlays different map images on top of each other to better visualize and analyze the effect of desired operation. For example, it can be used to find area affected by flood, or to find division of regions based on religion.

Figure 4.8: Overlay of KGP BNK ROAD, Block HQ & Boundary Layers

User can specify options to get the result image in desired format.
Options:
Layers = { kgp:bnk_road, kgp:bnk_block_hq, kgp:bnk_block_boundary }
Width = 768
Height = 679
Format = image/png

- Information about specific layer:
  - Title — POPULATION
  - Name — kgp:POPULATION
  - Is Queryable — 1
  - Is Opaque — 0
  - Bounding Box —
  - minx — 68.52669525146484
  - miny — 8.086045265197754
  - maxx — 97.3387680053711
  - maxy — 35.8697509765625

Provides information about a particular layer, for example, name of the layer, title, bounding box information etc.

- Similar information can be found for web feature service, for example, supported operations in WFS.

  1. GetCapabilities
  2. DescribeFeatureType
  3. GetFeature
  4. GetGmlObject
  5. LockFeature
  6. GetFeatureWithLock
  7. Transaction

Chapter 5

Spatial Query Orchestration

As much as it is important to find and save data for the later usage, retrieval of the data in a efficient and logical manner is also a crucial task. It is required that data retrieved from the database in in some logical order. This becomes helpful specially when query result contains large number of output results. User might not look at each and every retrieved data for finding useful information. Most of the time user will use top 5 or 10 entries of the result to make further use. For example, if a user wants to know which are good Chinese restaurants nearby him. A query might select a bounding region and retrieve all the chinese restaurants and show the result to the user. In this case user will not look at each and every restaurants and see their other quality to decide which restaurant he or she wants to go to, rather he will look at merely top 5 restaurants and decide where he wants to go. It is crucial that in this case this top 5 restaurants have some logical qualities that are suitable while retrieving the results. This is where Indexing and ranking of spatial data comes into play, with indexing and ranking of spatial data we can retrieve the useful information from the data in some predefined manner which ranks data into some metrics to gain useful information.

5.1 Quadtree based Indexing

Quad tree is a special kind of tree data structure. In this each node is divided into four child nodes. In node is then either divided into four nodes or kept as it is. Here the leaf nodes cannot be divided further and contains objects and features to be stored in a tree. Geometrically, each node here represents a square, which is further divided into four equal squares. If one cell contains more than one features than it is divided further. If the cell contains one feature only than it becomes leaf node and is not further divided. Quadtrees are highly used in domain such as image processing and spatial indexing.

Quadtree are specially used for spatial division of data. In this spatial features are drawn on a grid and the quadtree divides the grid such that each of the cell only has one feature in it. Many databases use quadtree for spatial indexing. Indexing can be useful not only in retrieving or select query but also in other type of queries which require data in turn to be executed.

5.1.1 Algorithm

In this algorithm, quadtree is created based on the number of points and area of indexing also known as bounding box. We first define the bounding box to create a area that defines the points to be inserted. Any points that lie in the bounding box will be added to the tree. Insert all the points into the tree and then select area of intersection. Area of intersection is the area which contains the output features. We can choose different overlap regions to find different features that reside in that area. Accessing the intersection from the quadtree, we can than display various information about the features residing in that area like id, bbox etc.

Algorithm 4 *Quadtree Indexing Algorithm*

Input: Set of spatial point features: features
Output: Arrangement of points when retrieved

```
bbox ← set bounding box For index
create index with bbox
for feature in features do
    insert feature in index
end for
overlap ← select area of interest
outputFeatures ← index.intersect(overlap)
for feature in outputFeatures do
    display feature
end for
```

5.1.2 Example Scenario

Let's look at a example to understand indexing based on a quadtree. Here we take 16 point features in a rectangular region. This points are uniformally distributed over the region in this case, but may not be in the real world. Bounding box for area is (0,0,80,80) and location of the point features are shown in the figure. We can take different overlap

regions to find the features which reside in those intersections.

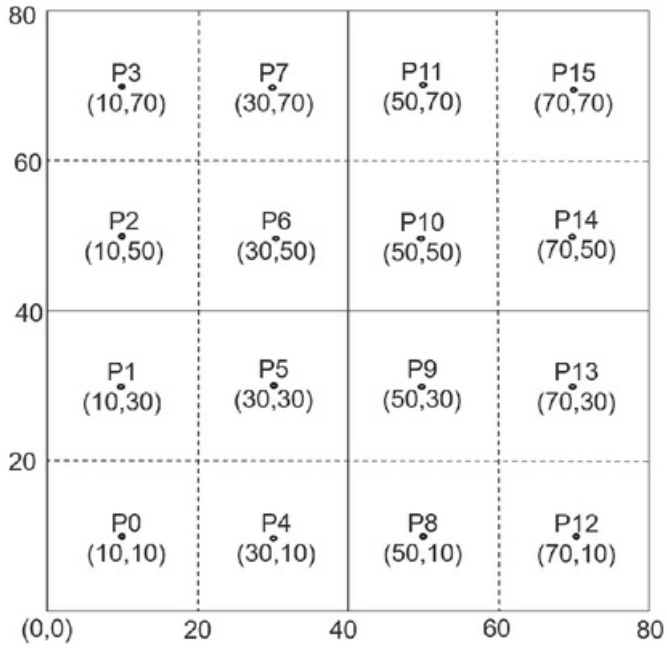

Figure 5.1: arrangement of feature points

Here we take a look at two different intersection regions, $olap1 = (0, 20, 40, 60)$ and $olap2 = (40, 20, 80, 80)$. When we intersect this overlaps with index it gives the points which resides in the intersection as below.

As we can see, quadtree can extract and list features from area of interest.

p2	(10,50)
p1	(10,30)
p6	(30,50)
p5	(30,30)

Table 5.1: Output with overlap region 1

p14	(70,50)
p15	(70,70)
p13	(70,30)
p9	(50,30)
p10	(50,50)
p11	(50,70)

Table 5.2: Output with overlap region 2

5.2 Ranking based on Quality Preferences

There are other type of advanced ranking mechanism available when more data about the features to be indexed is available. One of such method is ranking based on quality preferences. In this method, we look at some overall mathematical quality of spatial neighbourhood to determine the quality of the feature. There can be more than one methods to rank spatial neighbourhood based on mathematical functions.

For example, If one wants to buy a house, he considers many factors besides the attributes of the house itself. Although qualities of house such as area, no of rooms and price are crucial factors, other neighbourhood qualities such as distance from good quality restaurants or availability of general hospital are also an important factor. One always wants to have his or her home nearby good facilities like hospitals, restaurants and cinemas. The quality of each of this feature in the spatial neighbourhood of house is the determining factor while considering to buy the house.

Quality of each of the facilities is also an important factor. This is why we have defined spatial neighbourhood quality as a parameter of quality of each type of facility and not the quantity of feature types. One can also use the total sum of quality of all features in spatial neighbourhood. That will give the quantitative measure of quality features in the neighbourhood. For example, if there are more number of features that are low in quality, that neighbourhood can still be ranked higher than a spatial neighbourhood where there are less number of high quality features. Other types of mathematical measures can also be employed such as average quality of each feature or sum of average

quality of each feature type in spatial neighbourhood.

5.2.1 Algorithm

This algorithm ranks the quality of spatial neighbourhood for every feature to be ranked. Set of input features with information about other qualitative features are required for this purpose. We also need to set neighbourhood radius for examining spatial neighbourhood. The algorithm takes the input and finds the ranking of the input features. To do this, it first scans the spatial neighbourhood around each features. This is done by creating a bounding box of specified radius around the feature point. We find all the qualitative features in the spatial neighbourhood and find the maximum quality of each type of feature from the neighbourhood. The sum of each type of quality feature gives a score to a particular input feature. Input feature are ranked by decreasing order of their neighbourhood score.

Algorithm 5 *Ranking based on quality preferences*

Input:
features: Set of spatial point features
qualitative_features: Set of features with quality
radius: Spatial neighbourhood radius
Output:
Ranking of input features based quality of spatial neighbourhood

```
1:  Initialize:
2:      boundary_box ← Create bounding box based on radius and
        feature bounding box
3:  for feature in features do
4:      qfeatures ← Get all qualitative_features in the boundary_box of feature
5:      feature.quality ← 0
6:      for qtype in qualitative_feature types do
7:          qtype.quality ← extract maximum quality for the type in bondary_box
8:          feature.quality ← feature.quality + qtype.quality
9:      end for
10: end for
11: ranked_list ← features ranked by feature.quality
12: for feature in ranked_list do
13:     Display(feature)
14: end for
```

5.2.2 Example Scenario

In our example, we have taken four feature points to rank. They are $feature0(20,20)$, $feature1(20,60)$, $feature2(60,20)$ and $feature3(60,60)$. There are two types of neighbourhood features that we are interested in called box features and star features. Locations of star features are x1(10,70), x2(30,70), x3(10,30), x4(30,30), x5(10,10) and x6(50,30). Locations of box features are y1(10,50), y2(50,70), y3(70,70), y4(70,50), y5(70,30), y6(70,10). Quality of each box and star features are shown in the figure 5.2.

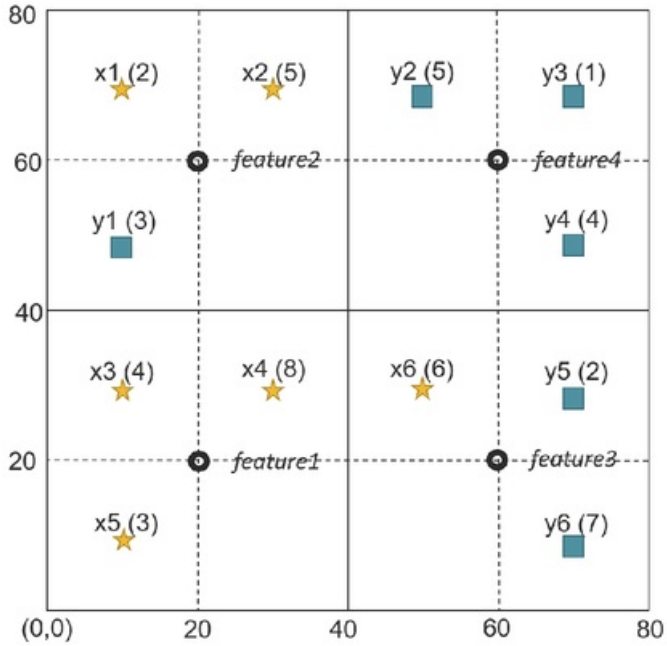

Figure 5.2: Ranking by quality preferences

In this approach, we first iterate on the feature points, here $feature1, feature2, feature3$ and $feature4$. For each feature we take a neighbourhood region corresponding to it. Here we have taken neighbourhood region to be of 20 distance buffer from the feature point. We find all the relevant features from this buffer and we find maximum quality of star feature and maximum quality of box feature from this buffer. This selected features represents the buffer and in transitive manner the feature itself. We can apply various operations to determine the quality of spatial neighbourhood by applying various mathematical functions on top of it. Here we have chosen the sum function for the

demonstrative purposes.

For each feature, the sum of other qualitative features in neighbourhood gives the final score of a feature. The higher the score, lesser the rank. In this example, our final ranking becomes.

Feature	*Rank*
2	13
0	8
1	8
3	5

Table 5.3: Final ranking for the features

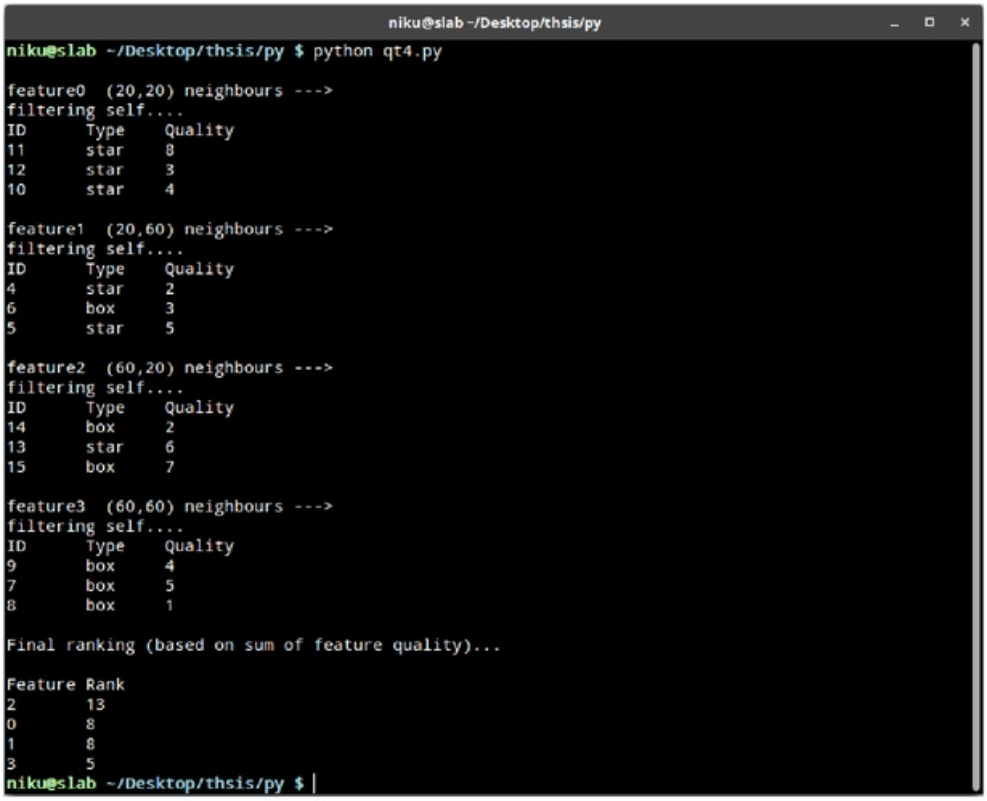

Figure 5.3: Features ranked by neighbourhood quality

Chapter 6

Conclusion and Future Work

Geo-service portal acts as a underlying framework or foundation for various kind of higher level use cases. Main advantage of this registry service is that we can avail data from various repositories that has been crawled and indexed for others to use. Basic querying facilities can be provided directly on top of registry service. Using geo-spatial crawler we can get the latest geo-spatial data available on the web. Once this data is got from various geo-servers, various kinds of applications can be built on top of that. Building an OGC compliant web service catalog can also be beneficiary as already available software and services can use the registry for various kinds of services with little to no modification of their original code-base.

Geo-service portal can be used for many kind of applications. This applications and use cases include wide variety such as just getting the relevant spatial information for the application to doing in-depth spatial or temporal or spatio-temporal analysis. Other use cases can include finding the spread of the disease on large geographic area. Which of the areas can be affected by flood can also be founded using data from catalog. All this services can work if underlying foundation of catalog service is built.

6.1 Contribution of thesis

Geo-Service portal builds ground work for spatial metadata gathering and publishment. It provides this foundation platform in a modular fashion, such that all of these modules work in a pipeline. Output of one module is feeded as input to other module. This gives independence to update modules irrespective of each other as it has loose coupling.

Spatial web crawler allows to crawl the open web and find existing geo repositories

and catalog servers to gather spatial data and store available metadata. This allows better availability of spatial features and data then conventional services.

Spatial catalog server allows us to maintain and publish spatial metadata and provides various OGC compliant services on top of it. This allows not only storage and retrieval of spatial data but also known services and methods attached to retrieval of spatial data and extraction of information from spatial data.

Spatial query orchestration allows retrieval of spatial features in and indexed and ranked manner, which can be useful while working with large number of features. It uses indexing based on quad tree which groups feature based on their proximity from the region of interest. Another advanced method to rank spatial features is by quality preferences, in which quality of feature and it's neighbourhood define the importance of such feature.

6.2 Future Scope

There is significant scope of improvements in the current framework for Geo-service portal as it is designed in a modular way. Extensions and higher level use cases can be provided to any of the modules or newer modules can be added. Some of the advancements to the current features are listed below:

Spatial web crawler module can scrap and crawl other things to make use of other qualities of data the catalog service or repositories that are hosting. This can be useful while ranking the quality of a whole registry with respect to other registries. A score can be provided to crawled registries for the spatial data they contain, size and rarity of the data, by no of times they are referenced and many other such metrics. As spatial data is time driven, catalog service can provide refresh rate, a time after which the catalog service is reloaded to check if their is any change in the data it hosts from the crawled sources. This is useful to find most up to date data from available geo repositories. Analyzing this behavior over a time can also give spatio-temporal behaviour of the data provided by a given repository. For example, spread of city in given range of time and rate of growth can be known. Ranking mechanism can be improved to give scores to repositories and in turn the data crawled from the repositories.

In today's era, when creating a service reliability, scalability and availability is very crucial things to consider. If the data is hosted on one server and single server architecture is implemented then there might be a high chance of failure on high load. Also the growth of the users and query load on the server cannot be predicted. For this reasons it would be better to consider a cloud based implementation for the above stated architecture. Cloud based implementation for crawler, registry service, and query processor can scale very well in the situations for high load. Cloud based implementation can also be good for availability because on situations of high load, infrastructure can be scaled up to cater the need of high load, in situations of low load the infrastructure can be scaled down to save power and investments. The registry itself can be replicated in the cloud at various geographical places to avail above stated high availability and reliability of the operations. Changes are also be easily and more efficiently done in the cloud because we don't have to shutdown and start the instances again. Cloud instances can be replaced in-place without shutting them down. This ensures versioning can be done of the software and eventual upgradations and roll-out of updates can be provided without hiccups.

Dynamic deployment of catalog server is the key to achieve high scalability and high availability time. In this approach we can dynamically deploy instances of GeoServer catalog server(s) to accommodate high demand from users. To make it transparent to users, *load_balancer* can be used. Similarly we can scale down in case of low demand from users. With these advancements, access control can also be achieved via *load_balancer*, because it can work as a single point of access for the users.

Bibliography

[1] Patil, Sonal, Shrutilipi Bhattacharjee, and Soumya K. Ghosh. *A spatial web crawler for discovering geo-services and semantic referencing with spatial features.* International Conference on Distributed Computing and Internet Technology. Springer International Publishing, 2014.

[2] Li, Wenwen, Chaowei Yang, and Chongjun Yang. *An active crawler for discovering geospatial web services and their distribution pattern - A case study of OGC Web Map Service.* International Journal of Geographical Information Science 24.8 (2010): 1127-1147.

[3] Najork, Marc. *Web crawler architecture.* Encyclopedia of Database Systems. Springer US, 2009. 3462-3465.

[4] Lopez-Pellicer, Francisco J., et al. *Discovering geographic web services in search engines.* Online Information Review 35.6 (2011): 909-927.

[5] Ahlers, Dirk, and Susanne Boll. *Location-based Web search.* The Geospatial Web. Springer London, 2009. 55-66.

[6] Li, W., et al. *Semantic-based web service discovery and chaining for building an Arctic spatial data infrastructure.* Computers & Geosciences 37.11 (2011): 1752-1762.

[7] Jiang, Jun, Chong-jun Yang, and Ying-chao Ren. *A Spatial Information Crawler for OpenGIS WFS* Sixth International Conference on Advanced Optical Materials and Devices. International Society for Optics and Photonics, 2008.

[8] *http://geopython.github.io/pycsw-workshop/*

[9] *https://geopython.github.io/OWSLib/*

[10] *https://github.com/geoserver/geoserver*

[11] Yiu, Man Lung, et al. *Ranking spatial data by quality preferences.* IEEE Transactions on Knowledge and Data Engineering 23.3 (2011): 433-446.

[12] Hjaltason, Gisli, and Hanan Samet. *Ranking in spatial databases.* Advances in Spatial Databases. Springer Berlin/Heidelberg, 1995.

[13] *https://github.com/karimbahgat/Pyqtree*

abc

ORIGINALITY REPORT

%9	%7	%2	%8
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.cc.gatech.edu Internet Source	%4
2	Submitted to Indian Institute of Technology, Kharagpure Student Paper	%1
3	gnu.inflibnet.ac.in Internet Source	%1
4	"Bibliographic Listings", Journal of Planning Literature, 2013. Publication	<%1
5	Submitted to Indian Institute of Technology, Kanpur Student Paper	<%1
6	Submitted to Indian Institute of Technology Guwahati Student Paper	<%1
7	Submitted to Indian Institute of Technology, Bombay Student Paper	<%1
8	Lecture Notes in Computer Science, 2014. Publication	<%1

9	spatial.umaine.edu Internet Source	<%1
10	www.kochi-tech.ac.jp Internet Source	<%1
11	dblp.dagstuhl.de Internet Source	<%1
12	Submitted to Higher Education Commission Pakistan Student Paper	<%1
13	Li, Wenwen, Miaomiao Song, Bin Zhou, Kai Cao, and Song Gao. "Performance improvement techniques for geospatial web services in a cyberinfrastructure environment – A case study with a disaster management portal", Computers Environment and Urban Systems, 2015. Publication	<%1
14	Submitted to Middle East Technical University Student Paper	<%1
15	www.inderscience.com Internet Source	<%1
16	Submitted to King Saud University Student Paper	<%1
17	Submitted to University College London Student Paper	<%1
	docplayer.net	

18	Internet Source	<%1
19	Submitted to Institute of Technology, Nirma University Student Paper	<%1
20	Submitted to City University of Hong Kong Student Paper	<%1
21	Wenwen Li. "An active crawler for discovering geospatial Web services and their distribution pattern - A case study of OGC Web Map Service", International Journal of Geographical Information Science, 08/2010 Publication	<%1
22	www.science.gov Internet Source	<%1

EXCLUDE QUOTES	OFF	EXCLUDE MATCHES	< 3 WORDS
EXCLUDE BIBLIOGRAPHY	ON		