Cloud based implementation for Geo Crawler

Deepak Punjabi, Bhumi Faldu and Mayank Gautam
School of Information Technology
IIT Kharagpur

***Abstract*—In recent times, the increase in spatial data and services is vastly increased. To deal with this huge amount of data some kind of indexing is required. But this data is heterogeneous in nature and there are other problems as the scale of the web.Web crawlers are the programs that helps us to find useful things in the web.We try to deal with these problems and try to implement a crawler based on WFS standards of OGC web services. Issue arrises when we deal with the scale of the data that to be crawled. Storage, Indexing and heavy computations are required for these operations. For elasticity we try to move our crawler implementation to the cloud.In this paper we try to build cloud based architecture of Geo services crawler.Performance evaluation is an important aspect to judge the semantics used for the system.**

I. Introduction

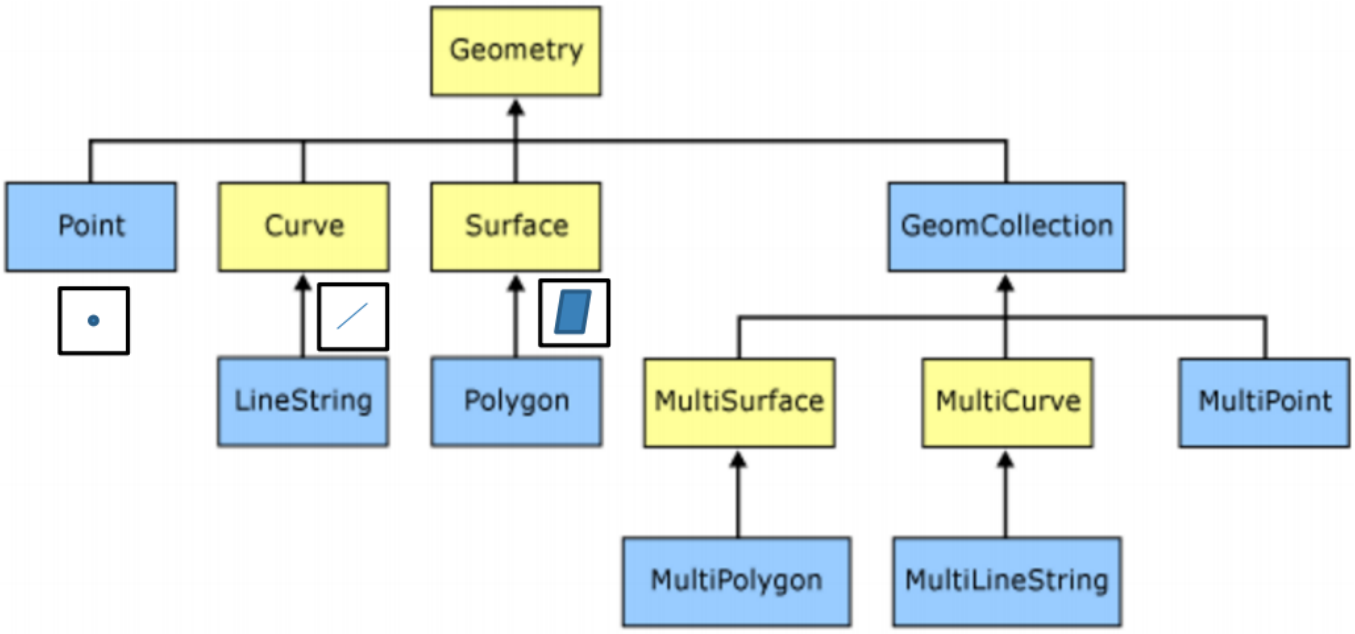
INTERNET is not the same as it was in it's initial days. It has grown many folds. Finding an accurate and quality information from such vast amount of information is a crucial task. Also, topical search results are based on the domain knowledge. It is very hard to utilize general tools for domain specific operations. One such class is geographical information. Finding and utilizing geographical information from web is a non-trivial task and when we do find such information it is not bound by the quality of service parameters. Time to crawl such information is huge and takes large computational cost. The based way to find the information from the web of this scale is via crawler but normal web based crawler fails to provide services that are dynamic in nature. That means changing the parameters to look for or the important information, we need to remake the crawler. Scaling can not be done via normal web based crawler. In this paper we give methodologies to built a cloud based web crawler for finding feature information from the Geo-spatial web servers. We filter and rank the feature services for the need and store them in a permanent repository

A. Document Structure

In this paper we start with introducing the information need in the internet. We see what are the crucial parts in finding important and useful information from the web. We see how scale , feature and applications of spatial data in increasing as ever. In part II we see what is spatial data. In which industrial systems we can find spatial data. What are the different types and objects of spatial data. Then we see what are the different services that are available on the internet to access spatial data in various forms. We see traditional approaches of searching spatial data and how they don't work. We then focus our discussion to web crawler to efficiently retrieve data. We see what the features it should and must have. Then we see what are the different types of available web crawlers. What is the objective of building spatial web crawler is disscused in the later part. Then we see a comprehensive detailed architecture of the spatial web crawler and understand it's parts. We discuss what are the advantages and challenges in building this spatial web crawler.
We then move our discussion to building cloud based architecture for the spatial web crawler. We see two approaches for implementing crawler that is based on the cloud and also the hybrid approach. We discuss what are the advantages we gain by implementing this cloud based approach.
After building the system, we see what are the different performance measures through which we can evaluate the performance of the system.

II. Spatial Data

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. 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 spartial data. Different operations can be add, visualize, annotate etc. Different examples of software that offers spatial services are ESRI, Microsoft SQL Server, ERDAS imagine etc.

A. Classification of Spatial Data

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, surface 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.

NASA has a satellite called Earth Observation Satellite, which takes map images of earth and sends it to observatory. It provides 3 terabytes(TB) of data on the daily basis as per the NASA. This calculates to 90 TB data over a month and over a 1000 TB of data in a year. This data is quite huge, it is unstructured in nature and it is un-indexed. Finding relevant information out of this data is a non-trivial task. In the next section we see how the spatial data

is accessed from the internet.

B. 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 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 GetCapabilties, GetMap and DescribeLayer. GetCapabilities operation returns capabilities of the server ie 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 overlayed 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 command. 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.

C. Searching of spatial data

There are previously known two popular and trivial approaches for searching spatial data.

1) **Catalogue approach**
   Service provider registers their services to the registry. The registry contains various type of registered services and their corresponding geo servers. But there are some problems with this approach. First one is that in many cases the registry is not up to date. Many times the latest services are not yet bound to any registry. One other problem is related with incorrect classification of services. This is specially an issue because user might be searching in the other part of the catalogue where it cannot find the particular service even it is there. Last kind of problem can occur because not all kind of providers registers all kind of services. Registry may be biased to some kind of services.

2) **Utilization of popular search engine**
   In an another approach we can also utilize popular search engines like Google, Yahoo, Bing or DuckDuckGo to find spatial data. But the problem with this approach is that it takes all data as general data and not spatial or other kind of unordinary data. Because of this we lose many spatial operations and features.

Another problem is the popular search engines use some kind of ranking of pages which is not based on the quality of the page(QoS). For example, google uses PageRank to determine the result webpages but it is not dependent on the quality of the resultant webpage but merely a measure of from how many other webpages the result webpage is addressed.

III. Crawler

As we have seen, the above two methods fail to provide proper spatial context when searching for the geospatial data. We need something that searches the whole web searches and preprocess the data available in the internet and give optimal suggestions about available spatial service providers. A crawler is such kind of agent. It is a program that browse through the world wide web in some systematic way and creates an index of the searched data. This index can be later used for faster access for web-page or categorization of the web-page. Crawled web pages are stored in permanent repositories. Crawler is a basic and essential building block of search engines. Popular examples of crawler programs are bingbot, googlebot and polybot.

Main goal of Web crawling is to collect effective and useful information(pages) from the web, where web contains more than trillions of pages, it is sometimes called as spider. There are some of the features, crawler must provides and some of the features crawler should provide. The basic start operation of crawler is to visit one or more URLs in seed set. Fetched page is then parsed and text of web page and links in web page are divided. Text is added into text indexer and links are fed into Index Frontier, Initially Index Frontier contain only link of seed set and as any page from frontier is fetched and parsed then that page is deleted from frontier but if crawler add fetched page to back of frontier then its called continuous crawling. crawling process may be looked like a web graph recursive traversal.

Crawler must have features :

- **Robustness**: Web contains some of the server which mislead crawler and lead crawler into infinite loop of fetching web pages. Its sometimes called as spider traps. Crawler must come to know about such kind of traps.
- **Politeness** : Politeness policies that decide rate of traffic directed to any web server. There are many policies, one of the basic policy is no to concurrent call to same web server, another one is to decide time between two subsequent call of same web server.

Crawler should have features :

- **Distributed**: Crawler is a big activity which is very easily divided into independent tasks and so each those individual and independent tasks are distributed over multiple machines and parallel activity of tasks make crawling fast.
- **Scalability**: Scale up feature in crawler is done by adding more machines in task or by adding more bandwidth.
- **Performance and Efficiency**: Efficient way to use processors, storage space and network bandwidth.
- **Quality**: For a given query, how much useful pages are fetched by crawler consider as a quality.
- **Freshness**: Change into any page contents takes some time to reflect on result of crawler. In such case, crawler works continuously to fetch newly updated and added pages.
- **Extensible**: Crawler architecture must be modular in terms of new data formats, new fetch protocols and many more.

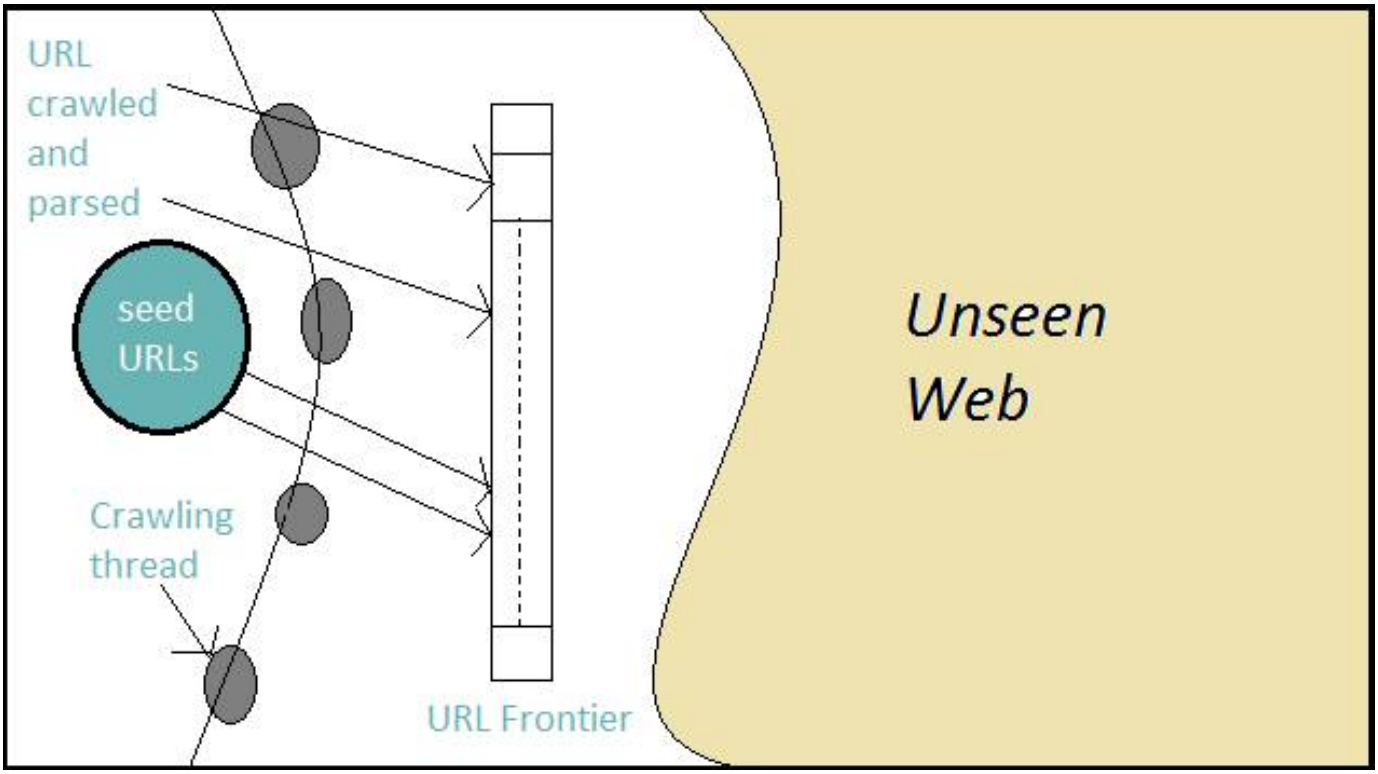
A. Types of web crawler

There are mainly two types of web crawlers.

1) Universal crawler
2) Preferential crawler
   - Focused crawler
   - Topical crawler

Universal crawler is the crawler that treats all kind of webpages as same or given them same weight. Other kind of crawler is a preferential crawler which prefers one kind of webpage over another. One of the type of topical crawler is topical crawler. One type of topical crawler is spatial crawler to which we will look into detail.

- **Universal crawler** :
  Universal crawler is also known as Giant search engine. It includes huge cost in terms of availability, communication bandwidth, storage and processing power. Theres mainly two major

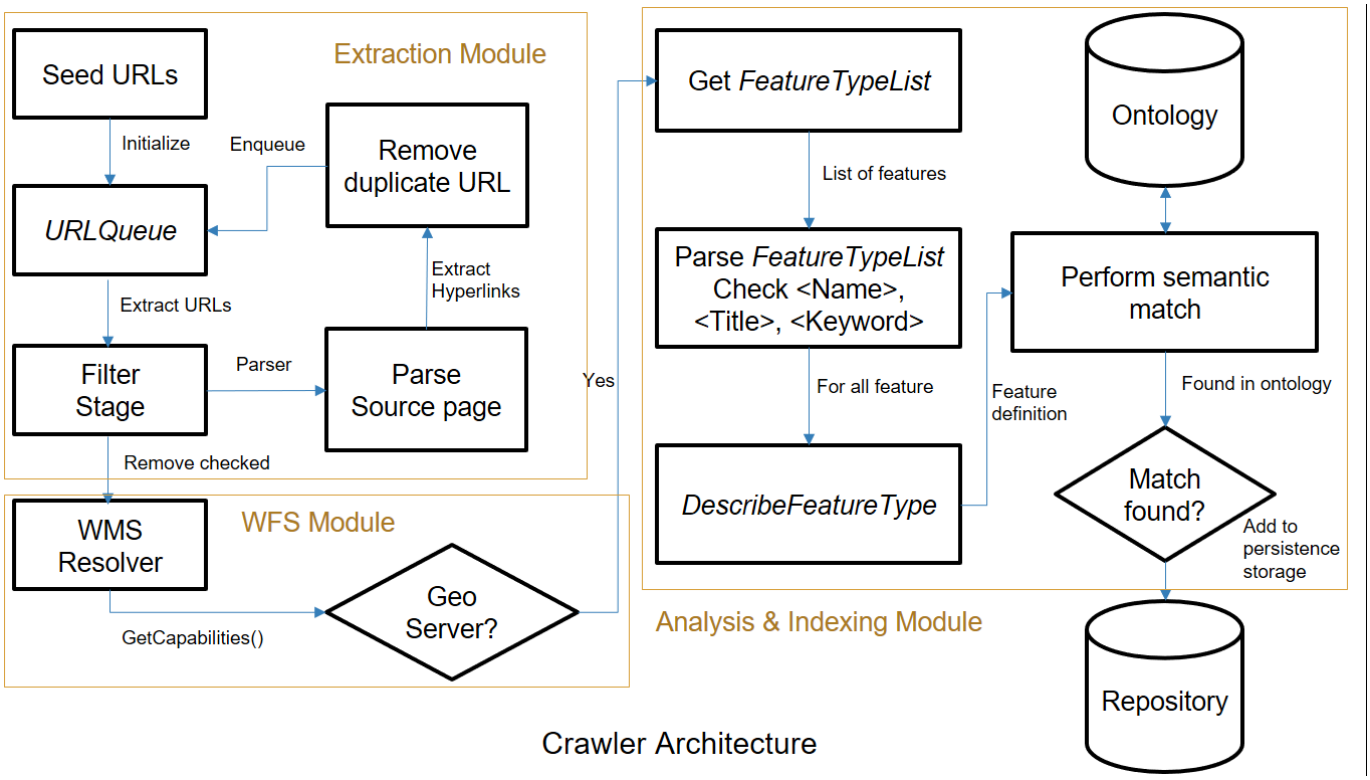


issues in large scale Universal crawler. First one is Performance, need to scale up our infrastructure to trillions of pages. Second one is about coverage, freshness and bias. Coverage check how quickly new pages are added to crawler. Freshness is to decide time duration in between of two fetches of same page. All pages are not equally important so that factor is decided by Bias parameter Ex. News portal has high probability for newly added and updated data.

- **Preferential crawler** :
  In Preferential crawler, each page has its own importance factor I(P). User wants to visit pages in decreasing order of I(P), this specification is implements in Frontier as a priority queue sorted by I(P). Importance of pages depend totally on application requirements, Ex. Pages that are closest to seed pages, Pages that are most central based on pagerank. Example of preferential crawlers are Breadth first web-graph search, Best-N-first, Pagerank, Sharksearch and Infospider. In breadth first algorithm of crawler, pages are visited in order of encountered. Best-N-first algorithm works on concept of top N webpages in frontier, sorted by similarity. In Pagerank algorithm, pages in Frontier will be sorted based on keywords. Sharksearch crawler, sorting in priority queue is based on similarity with immediate parents and anchor text. Infospider crawler works on adaptive distributed algorithm.
- **Focused crawler or Topical crawler** :
  Best-N-first is one of the Examples of Preferential crawler which filter out relevant pages based on topics. Crawling starts with seed pages of given topic. Belief behind Focused crawler is that relevant pages contain relevant links and that lead to crawling function but as depth of search is increased, relevance of page towards any topic decrease so till relevant documents are found on depth search will be continue. Focused crawler require less money and effort than other crawler because theres no unrelated page crawling.

B. Spatial Web Crawler: Objectives

1) Build a spatial web crawler which crawlers through 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.



Crawler Architecture

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.

C. 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:

- **Seed set**: a set of test URLs to initialize the queue for crawling the web. The crawler starts with the URLs given in the seed URL set.
- **URLQueue**: A queue that contains set of URLs to be crawled. The crawler module takes URLs one after the another from this 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.
- **WMS resolver**: WMS resolver is a module that checks that if a server is WMS server or not given the URL from the URLQueue.
- **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.
- **Ontology**: semantic dictionary containing all the features its type and relationship hierarchy between features

Our crawler contains mainly three modules:

- **Extraction module**
  Our algorithm starts with a set of seed URLs contained in the seed set. We initialize the URLQueue with these seed URLs. The filter stage takes URL one after the other 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 webpage 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.

```
<?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>
```

Fig. 1. XML response from the geo server

- **WFS module**
  Once we have a URL for the examination, we send a GetCapabilities() request to that server. We do this by appending the request to the url. $services?REQUEST = GetCapabilities\&version = 1.1.0\&service = WFS$ 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. 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 the feature is found in the ontology then it is added to permanent storage in the repository.

D. 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.

E. 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 spatialized features. For example it suggests that river comes under the type stream which intern comes under the geometry type waterbody. This type of hierachical 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.

F. Challenges

There are many challenges for practically implementing a crawler in the real world. Scale of the web is so huge that the simple implementation of the theoretical crawler will be very poor. One other problem is that many pages generate, add or delete on the dynamic bases. We need to reflect this changes into the crawler. This is a crucial parameter for setting the update period or refresh rate. Another problem arises because of heterogeneity in the available data on the world wide web. How to register and store this data into the single repository depends on the implementation of the algorithm. Major problem is related to the scale of the web. Crawling through this huge web is a time and computation consuming task and high performance computing and parallism is a must need in this type of task.

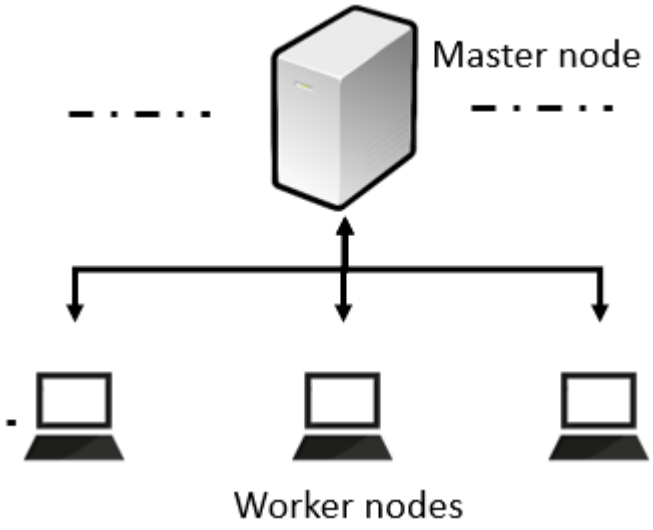
IV. Cloud based Geo-Crawler

According to survey report by *internetstats* in 2012 2.4 billion people were actively using the internet. By this estimation the global usage traffic will reach 14 GB per user in the year 2018. This clearly stats that the data and information need is going to rise as ever. Finding correct information from the web is a time consuming and costly task for web crawlers.Traditional web crawlers do not deal with the scale of the web efficiently. Traditional web crawler has the task to take the url download the web-page and parse the web page for other hyperlinks. These are the tasks which can be parallelized. Different servers for optimally doing different tasks can be accommodate via geographic distribution over the glob. This particular servers can deal with the specific task efficiently. Most of the big companies use multi server and distributed environment for working for several reasons. The task of downloading the webpage, filtering a web link, parsing the web page and finding useful links, doing some domain specific semantic matching, these are the tasks that can be run in parallel. Different geographically distributed servers can be accommodate to do specific tasks efficiently.In this paper we define a cloud based approach to crawl through geo spatial information available on the internet. Find the servers which offers web feature type of spatial services and store these web pages in the permanent repository to use in later tasks e.g. searching. This implementation make use of *high performance computing architecture (HPCA)* to speed up the computation of the tasks.

A. Master-worker approach

As the name suggest, Master-worker approach for crawling Geo-Data divides tasks between different workers for balancing load. parallel concept in crawling data actually speed up the process of our search engine. As we already discuss traditional approach of Geo Crawler, the main task in the crawler are fetching the url from the urlqueue, parsing the web page to extract hyperlinks, checking whether the particular server offers geo services or not and maintaining a permanent repository for indexing and accessing previously crawled data.

As we can see most of this tasks are parallelizable, they can work independent of each other. We take advantage of this phenomenon to produce a master worker based approach. There is one master node to manage and distribute tasks over other nodes. Master node is used to synchronize between works of worker nodes. We have multiple worker nodes that takes a url from the url-queue. Multiple other nodes takes this input and downloads



the web-page. Worker nodes parse this web-page to extract hyperlinks and add them again in the urlqueue. Part of the workers find if the url server offers geoservers functionality or not. If the server offers such functionality the server is added to the permanent repository.

We can add other worker nodes based on the load on the system. This can be done through allocating more Virtual machines to the task. On low load situation we can delete un-used virtual machines or we can migrate the task to more powerful machine and deallocate the current virtual machine.

B. Map-Reduce based Approach

In the map reduce based approach we have multiple mapper nodes and multiple reducer nodes. The Task of crawling is divided into the this infrastructure. We can divide the task of fetching the urls into the mapper tasks. The mapper nodes takes the url and crawls through all of its links and store into a intermediate mapper representation. This representation is sorted over different kind and domain of web servers.The mapper nodes are working in parallel. So multiple mapper nodes crawls through urls given to them, parse the links from the web page and store them into the intermediate representation. In this phase we also check whether the link is already crawled or not. The filter stage is applied for this. We check this by checking the intermediate representation, if the url is found in the mapper phase, it is not added again.

In the reducer phase, we take the url from the mapper nodes. Here multiple mapper node can crawl through the same domain. The reducer node takes input from all such nodes. The reducer nodes are in contact of permanent repository. They check if the url is already classified as a geo server in the repository. If so, no processing is done further. Otherwise the reducer node checks if the available url is of a geo server which offers a web feature service. For this standard methods are applied as mentioned in the standard web crawler architecture. If the server is found to be a geo server it is added into the permanent repository and indexed accordingly to features it is offering. Multiple such reducer nodes are implementing the same things. High degree of parallelization is achieved through this problem and high speed up acquired respect to the traditional geo crawler.

For implementing this we use hadoop framework. Using hadoop we can provide dynamic scaling in the infrastructure by dynamically adding servers at run time. The primary task is divided into smaller modules and each module performs a specific task. Synchronization between the nodes is achieved via some master nodes. In implementation part we use ad hoc implementation combining both master slave and map reduce based approach.

C. Advantages of Cloud Implementation

- **On-demand scaling**
  Scaling of the web can be easily accommodated in the cloud based implementation of the web crawler. We can dynamically add and remove servers from the cloud based systems. for example, if we want to speed up the task of fetching the urls we can introduce extra servers for the same task. Other parts of the system need not be changed and a local optimization can be performed.This allows the web crawler the dynamically add sub modules to the given crawling system for high load situations and dynamic scale up.Similarly in case of low load situations we can de-allocate virtual machines and agents to reduce the cost of the operation of the web crawler. This helps very much in economy of scale. This kind of crawler can perform very efficiently in any type of load situation whether high or low load.

- **Distributed processing**
  Multiple geographically distributed servers can be accommodated to globalize the coverage of the Geo crawler. It provides reliability in the operation. Means if one of the node fails still the operation continues uninterrupted for the crawling task. It also provide low network latency and high response time for the clients. We can increase the no of server agents in the locations nearby to the high load clients. This drastically improves response time to the for the client. The crawling agent is nearer to the client because of that the service response time for the client increases. If one server has failed system can redirect the client to other crawling agents. This can be also utilized for distributed repositories to reduce access time available for retrieving the data form the repositories. This can also be utilized to crawl locally relevant services.

- **Cheaper and faster for small businesses**
  For small businesses and start-ups initial cost is very crucial parameter. In this cases building own crawler for finding domain specific services like geo services will be a very costly task. Cloud based implementations provide on demand and pay per use services. Client only has to pay for the usage and not the whole software and service. This rent based models help a lot to small businesses for finding and retrieving domain specific web feature services. Major cloud and service providers can implement this kind of services for small businesses through their reliable and faster crawlers.

- **Ubiquitous access**
  Cloud based resources are available over the internet. This resources can be accessed via any device provided you have proper bandwidth and internet access. It means that clients are not constrained to use the service via any constrained devices and platforms. As cloud based services can be accessed anywhere via the internet the nature of our geo crawler service is ubiquitous. Cloud based paradigm increases availability of the services far better than the traditional client server services.

V. Evaluation of the system

The performance of the system is measured taking various seed URLs, running the algorithm against given sample data set in which correct results are already known. There are three kind of measures used for measuring the performance of the spatial web crawler. Performance is found using various LCS parameters, In this type we change the size of the sub-string that we want to match and check the results at various points.

- **Precision**
  Precision is found by dividing the no of geo servers found by the spatial web crawler by the total no of geo servers found. If we increase the length of the LCS it means we are comparing words and features as it is, because of this error in the result gets reduced. Hence average precision increases.

$$precision = \frac{no\ of\ relevant\ Geoservers}{total\ no\ of\ servers} * 100\% \tag{1}$$

- **Recall**
  Recall is found by taking a division of no of geo servers found by the spatial web crawler by actual set of existing geo servers. If we increase the LCS length threshold, the value of recall increases until some point and then it starts decreasing.

$$recall = \frac{no\ of\ relevant\ Geoservers}{total\ no\ of\ Geoservers} * 100\% \tag{2}$$

- **F1 measure**
  To normalize the values received by precision and recall parameters we generally use more robust F1 measure. It is seen that also in F1 measure, the value of measure increases until some point and then it starts decreasing.

$$F1 = 2\ *\ \frac{precision\ *\ recall}{precision\ +\ recall} \tag{3}$$

Final evaluation is done by taking this measure for all the feature types and averaging it over all the features. We need to find the optimal LCS length by experimentation to see at which point the algorithm gives optimal performance.

VI. Conclusion

Geo-spatial data is ever growing in todays era. This type of data is hard to search from the world wide web and index it for further processing. Different kind of services are available for searching the spatial data from the web but non is proven efficient and reliable enough to use for domain specific applications. Our algorithm suggests a way for building a WFS based Geo-crawler that efficiently crawls and indexes founded features into the permanent repository. This repository can then be used in many applications like search engines or data mining. This can help in many applications such as transportation & navigation, urban planning and emergency response planning. For better scaling we use cloud based approach to provide additional features for our system like dynamic scale up and on demand pay for use access. Cloud based implementation provide more robust and efficient approach for building crawler that searches, indexes and provides extra operations based on geo services.

References

[1] Patil, Sonal, Shrutilipi Bhattacharjee, and Soumya K. Ghosh. "A spatial web crawler for discovering geo-servers and semantic referencing with spatial features." Distributed Computing and Internet Technology. Springer International Publishing, 2014. 68-78.

[2] Li, Wenwen, Chaowei Yang, and Chongjun Yang. "An active crawler for discovering geospatial web services and their distribution patterna case study of OGC web map service." International Journal of Geographical Information Science 24.8 (2010): 1127-1147.

[3] 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.

[4] Marc Najork. Web crawler architecture. Microsoft Research.

[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] Bahrami, Mehdi, Mukesh Singhal, and Zixuan Zhuang. "A cloud-based web crawler architecture." Intelligence in Next Generation Networks (ICIN), 2015 18th International Conference on. IEEE, 2015.

[8] Suakanto, Sinung, et al. "Building crawler engine on cloud computing infrastructure." Cloud computing and social networking (ICCCSN), 2012 international conference on. IEEE, 2012.