



A Spatio-Temporal Model for Massive Analysis of Shapefiles

Dr.Prabha Shreeraj Nair

Dean Research, Tulsiramji Gayakwade Patil College of
Engineering and Technology, Nagpur

ABSTRACT

Over the course of time an organization working with geospatial data accumulates tons of data both in the form of vector and raster formats. This data is a result of co-ordinated processes within the organization and external sources such as other collaborative organizations, projects and agencies, crowd sourcing efforts, etc. The massive amount of data accumulated as a result and the recent developments in the distributed processing of geo-data have catalyzed the development of our spatio-temporal data processing model. Our model is loosely based upon GS-Hadoop and uses dataset consisting of more than 300,000 shapefiles (a vector data format), which was accumulated over a span of several years from various sources and creation of custom geo-portals for government departments. The developed model provides access to visual representation, extraction of features and related attributes from over more than 800 GB of shapefile data containing ten of billions of features. In this paper, we model a spatial data infrastructure for processing such huge amount of geo-data.

Keywords: *Shapefiles; spatio-temporal processing; spatial processing; feature extraction;*

I. INTRODUCTION

A Geographic Information System (GIS) consist of a collection of applications which operate upon geographic data and are utilized for planning purposes. Geographic data is collected from many sources which include high resolution satellite sensors and imagery to low resolution photographs uploaded on social networks and tweets by billions of internet users. The advancement in semiconductor devices and electronic components has made sensors small, cheap,

power efficient and easier to embed geographic location with captured data [1]. Satellite systems have been long used for navigation purposes and there have been numerous additions to global positioning system (GPS) with comparative and counterpart efforts on regional scale by India's NAVIK and on global scale by Russia's GLONASS, China's BeiDou and Europe's Galileo. These efforts not only provide simple and free access to geographic position but are also used commercially for application requiring high precision up to a centimeter. Apart from the positioning systems, recent launches by ISRO such as SCATSAT-1, INSAT 3DS, Cartosat-2C and numerous other satellites of Earth Observation System from NASA gather and continuously generate geospatial data by collecting terrestrial information [2]. The data, thus collected, spans domains of weather forecasting, oceanography, forestry, climate, rural and urban planning, etc. Storing such large volumes of data in a structured manner is one of the challenges. Processing such volumes and deriving useful information which is required for planning purposes and accurate prediction required for decision making forms one of the most important part of the challenges.

Geospatial data, for these applications, is mostly collected in raster form which is then transformed to a more usable vector format after application of image processing techniques (which can include manual editing, etc). A collection of geospatial data for an organization can be stored in a geo-database for multiple simultaneous access, security reasons and centralization purposes. Keeping geo-databases aside, most if not, all of vector data for the geographic datasets is stored in shapefiles (.shp) and XML formats. Shapefile is the legacy vector data format

developed by ESRI (Environmental Systems Research Institute) and is the de facto format for most widely used desktop GIS software such as QGIS, products from ESRI, etc while the later is preferred for exchange of data between applications and on web (e.g. using web-services).

Apart from the remotely sensed data, a large amount of geospatial data also comes from digitization of maps, field surveys conducted for planning purposes, engineering drawings (CAD), etc. There have also been numerous crowd sourcing efforts which include Google Map Maker, Wikimapia, OpenStreetMap and others [3]. Most of these online aggregators have strict and restricted licensing requirements for commercial use of the data and derived works. OpenStreetMap (OSM) is operated by the OpenStreetMap Foundation which does not own the project or the community sourced data. OSM has a very non-strict license which permits the use of the data (except for the raster tiles) for any purpose, creative, educative or commercial. Free, open and publicly available dataset from OSM consisting only of vector features is available in XML format (*OSM XML*) and binary (*.pbf*) format. The OSM format represent vector features such as points, lines and polygons in the form of nodes, ways and relations as shown in Figure 1. The vector data from OSM is openly available since 2012 while for the year 2016 it amounts to >800 GB of uncompressed XML consisting more than 3.5 billion vector features. This is just one representation of massiveness of geo-spatial data.

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
  <bounds minlat="54.0889580" minlon="12.2487570"
maxlat="54.0913900" maxlon="12.2524800"/>
  <node id="298884269" lat="54.0901746"
lon="12.2482632" user="SvenHRO" uid="46882"
visible="true" version="1" changeset="676636"
timestamp="2008-09-21T21:37:45Z"/>
  <relation id="56688" user="kmvar" uid="56190"
visible="true" version="28" changeset="6947637"
timestamp="2011-01-12T14:23:49Z">
    <member type="node" ref="294942404" role="" />
    <tag k="ref" v="123"/>
    <tag k="route" v="bus"/>
    <tag k="type" v="route"/>
  </relation>
  ...
</osm>
```

Fig 1: A sample OSM XML (consisting of *nodes*, *relations* and *members*)

The analysis required to be performed on such huge datasets require large amount of storage, compute and memory which is not available with a single computer. The advancement of utilizing remote

infrastructure for distributed storage, processing and network capabilities and ease of managing IaaS (Infrastructure as a Service) resources in Cloud Computing compels their utilization for temporal analysis of large amounts of geo-spatial data. It is further not possible to process and utilize such huge amount without the aid of parallel/distributed processing techniques such as Map Reduce.

A. Shapefile

Vector data was historically stored in a format known as coverage. Coverages are data stores which stores the vector features (point, line, arc, polygon, etc.) including their topology. The associated attributes for the features are further linked with the coverage in tables stored in separate files. Coverages are used for geospatial data requiring topological relationships between the features. In coverage connecting and adjacent features will share a boundary and multiple features cannot overlap each other ensuing topological correctness. The complexity of representing data in coverage (and maintaining the topological correctness of data) led to their utilization only for complex mapping and was not widely adapted. This made way for a new simple vector file format known as shapefiles.

Shapefile format (published in 1998) from ESRI (Environmental Systems Research Institute) is an Open Specification which is simple to implement and it does not store the topological information for the features [4]. Moreover, it utilizes DBF file (dBASE® format), a database format widely used at the time, for storing the associated attribute information. A Shapefile can only store one of the vector feature geometry (either point or line or polygon) at a time and can have multiple features overlapping each other as compared to coverage. This topologically insensitive and simplistic view of representing vector features led to wide adoption of the format. Today, most of the desktop GIS softwares support shapefile as the default format. The Shapefile is a collection of multiple files, each serving a different purpose. The main shapefile (*.shp*) in addition to storing the features also stores the information regarding the actual extent (The Bounding Box having *Min (X, Y)*, *Max (X,Y)*) of the shapes in the file header. The file header is followed by the geometric data for the shapes. The shapes stored in the main shapefile are indexed in another file; shape index file (*.shx* or *.sbn* or *.sbx*) indexes the vector features using R-Tree

indexing. The shapefile index is independent of the main shapefile. The shapefile format is not strict for the use of the related index file and as a result desktop softwares such as QGIS and MapServer index the main shapefile using Quad-Tree algorithm in a separate index file (.qix). All of the components of the shapefile are limited in size of 2GB, a restriction imposed by the proposed format. This restriction also limits the capacity of the shapefile and it can at most store 70 million point features. Shapefile is also complemented by a projection (.prj) metadata file which stores the co-ordinate and projection information and is either geographic (longitude, latitude represented by GEOGCS) or is projected (X, Y represented by PROJCS) for the shapefile. Softwares from ESRI represent uses a Projection Engine (PE) [5] to generate a projection which is stored in the .prj file representing units, datums, and spheroids using Extended Backus Naur Form (EBNF). The format is also supported by most desktop GIS softwares. A sample representation of a PE string is shown in the Figure 2.

```
GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137,298.257223563]],PRIMEM["Greenwich",0],UNIT["Degree",0.017453292519943295]]
```

Fig 2: PE String stored in (.prj) files can have geographic (GEOGCS) or projected (PROJCS) representations.

B. Hadoop (MapReduce) and Distributed GIS

Apache Hadoop, one of the most widely used distributed processing frameworks now-a-days, has been designed from the ground-up to process data from web crawlers which is in the form of plain text. As a middle layer, it enables aggregation and utilization of distributed computing, storage and network resources. Hadoop supports MapReduce applications from the ground up and has HDFS (Hadoop Distributed File System) which provides a fault tolerant distributed file system scalable to thousands of compute and storage nodes providing access to petabytes of storage capacity.

Geospatial data such as those available in the form of shapefiles can be converted to a text format such as CSV or TSV for processing on Hadoop. Moreover, as spatial data cannot be indexed using traditional B-tree structures used by RDBMS, several libraries such as

JSI (Java Spatial Index), libspatialindex and SpatiaLite are available for spatial indexing using advanced data structures such as R-tree, Quad-tree and R*-tree. These indexing mechanisms have also been natively incorporated in distributed processing frameworks such as Spatial Hadoop, Spatial Spark and GeoSpark [6]. These adaptations to MapReduce have considerably voided the requirement of a costly and specialized parallel or distributed system which presents its own challenges and limitations. These recent developments in the field of distributed GIS mainly focus on transforming vector and raster data akin to the format understandable directly by the underlying distributed framework (e.g. Text for Hadoop and Spark).

II. PROBLEM DEFINITION AND RELATED WORKS

There are hundreds of different file formats which have been developed for use with variety of applications for storing geo-data in both raster and vector formats. Nevertheless, huge amount of vector geo-data is available in form of shapefiles. The structure of each shapefile with its related attribute table can be unique and it also depends upon the standardization adopted by the creator of the shapefile (or it may not be standardized at all). This heterogeneity in the attribute information, binary format of shapefiles and co-locating all the shapefile components at a single location presents a huge challenge while adopting any GIS system over a distributed framework such as Map Reduce and Hadoop. Map Reduce requires data in a pre-defined format and by default it assumes a collection of (Key, Value) pairs as shown in Figure 3. It is not possible to create a single input format representing a number of shapefiles available in heterogeneous format which can be further be followed in Map Reduce as (Key, Value) pairs. If intended this transformation will become the bottleneck of the system and will increase the complexity of the actual processing code. This in turn will increase the development time and wastage of precious computing resources.

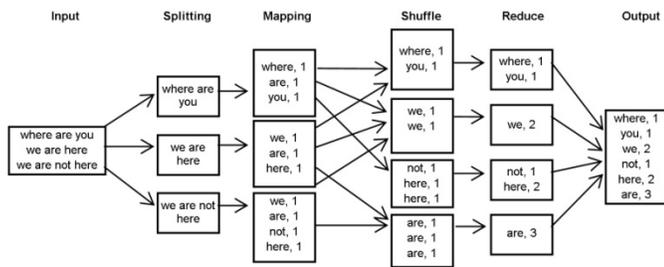


Fig 3: Different phases in MapReduce from Input to generation of (key, value) pairs and computation of final output for a word counting program.

There have been decades of development efforts in geo-processing algorithms, methods and techniques which have yielded to highly stable libraries such as GDAL/OGR (Geospatial Data Abstraction Library), GRASS GIS, GeoTools and alike which support nearly all raster and vector representation formats. It is interesting to know that GDAL itself supports 221 different raster and vector file formats [7]. GeoTools is an Open Source Java code library which implements nearly all the OGC specifications for working with geospatial data and can form the base for development of GIS applications. GeoTools is used by GeoServer, uDig and others. Several other libraries, tools and framework are available under the umbrella of OSGeo/OSGeo4W (Open Source Geospatial for Windows) and FOSS4G (Free and Open Source for GeoSpatial) [8]. This expansion of representation of geographic information and maps over the web has led to development of several Open source Web-GIS projects such as MapServer and GeoServer. To reduce the development efforts several options such as MapBox, CartoDB, QGIS Cloud and LizMaps are commercially available for hosting GIS applications (using their mapping API) in the cloud. All of these libraries consist of a comprehensive collection of functions suited for development of any GIS application. Well tested set of functionality from libraries such as GeoTools and akin can be made available upon MapReduce and provisioned to users requiring analysis of large and complex geo-datasets.

The best way to represent geographic data is in an interactive visual form rather than tables containing columns representing location co-ordinates in form of text. Many a times, it is also required to extract a subset of geo-data from such large volumes. It is not possible to parse through each and every individual record/feature (of shapefiles) from tens to hundreds of gigabytes of geo-data either to visualize it or to

extract small subsets (using relational query) for a specific purpose. A single computer system or a server, no matter the amount of computation power it possess, cannot process and provide the required output without years of specialized development efforts and creating a specialized system.

Several systems have been proposed to deal with the issue of using an existing distributed processing framework to work with geo-spatial data. One of such proposed system; SHAHED supports querying, mining, and visualization of NASA LP DAAC archived data. The data available is structured and represents Temperature, Crop and Ocean/Water information. The system is inclined towards cleansing of uncertainty from the data and generating spatio-temporal heat maps and videos for time ranges and parameters selected by the user from the pre-built index. The indexes of the data which is downloaded regularly from NASA are generated daily, monthly and yearly by Spatial Hadoop. The system is tightly coupled with HDF data format available from LP DAAC. Another system TAGHREED [10] for efficient and scalable querying, analyzing, and visualizing geo-tagged micro-blogs, such as tweets, etc is available. The system is scalable enough to supports high arrival rates of record such as tweets from a platform and can manage billions of records while maintaining the generated index in-memory. Parts of the index are flushed back to disk indexes as and when required. The system is tightly coupled with pre-defined attributes which includes the geo-location of the record and is specifically targeted towards tweets. It is targeted towards structured data, while the location is only in the form of points when compared with shapefiles which has additional types of lines and polygons. The system cannot be extended to support data format of shapefiles due to the heterogeneity of multiple shapefiles and the variety of attributes present. MapReduce-Based Web Service for Extraction of Spatial Data from OpenStreetMap dumps is available over the web [11]. The system makes it efficient and easy to extract OpenStreetMap data (geospatial data) which can be used for research and development activities and testing experiments. As data extracted from OSM will provide actual (real life) details regarding features such as the road networks, rivers, buildings, parks, etc., it becomes a de-facto choice for testing of any GIS application or library. The system translates the data from OSM XML to CSV/TSV format which can further be used by a distributed spatial geo-processing system such as

Spatial Hadoop. The system is similar to the OSM extracts provided by OpenStreetMap Data Extracts [12] but differentiates in terms of user selectable area and a fixed type of feature entities which can also be specified by the user. The system does benefit by distributed processing and storage provided by Hadoop but it does not employ any indexing method for the OSM XML. This lead to the repeated parsing of the data and the process becomes a bottleneck for multiple extract required by multiple concurrent users.

Our focus is on creation of a spatio-temporal data processing model which will provide near real-time and visual access to geo-data from hundreds of thousands of shapefiles. Apache Hadoop is best suited for development taking in view the development efforts and considering the large user-base who have tested it and deployed many applications over it. The user base also provides a plethora of extensions to support variety of user defined data-types stored in variety of file types and execution of a variety of applications [13]. As HDFS is tightly integrated with Hadoop, it is the most suited distributed file system for storage of huge amount of geo-spatial dataset consisting of a million files. The development of our data processing model is based upon GS-Hadoop which enables co-location and utilization of Shapefiles with GeoTools library on Hadoop utilizing data from HDFS [14]. The Shapefile dataset utilized consisted of more than 300,000 shapefiles. This data was accumulated over a span of several years (~9 years) from various departmental projects which required digitization of paper maps and creation of custom and online geo-portals. The developed model framework will also relieve the geo-scientists from the complexity of design and development of a distributed system and focus on insights derived by performing complex spatio-temporal analysis and operations.

III. PROPOSED SPATIO-TEMPORAL DATA PROCESSING MODEL

A model has been devised which provides access to visual search and extraction of features and related attributes from shapefile data scalable for millions of files totaling to tens of billions of features. We have also modeled a spatio-temporal data infrastructure based on the model and provide detailed works needed to be done in development of such spatial data infrastructure for processing such huge amount of heterogeneous geo-data. The spatio-temporal data

processing model consist of major four steps/phases namely 1) Data Sanitization; 2) Data Pre-processing; 3) Data Indexing and 4) Filtering and Visualization at the Geo-portal interface. These phases in the model are depicted in the Figure 4.

A. Data Sanitization

This is the most important and time consuming phase of the model as it goes by every bit of the geospatial data (Shapefiles) and provides clean input data for the other phases. This phase is similar to the data cleaning phase in a data-mining model which cleans and filters all the incorrect information. Invalid and corrupt Shapefiles are identified according to a given criteria such as with vector features outside the bound extents of the shapefiles, shapefiles with invalid bound extents or shapefiles with invalid projection, etc. It is also possible to filter the Attribute information or structure the attributes according to the output requirements at this phase. While processing with our dataset we identified about 37,521 shapefiles with EPSG¹ (a standard format for CRS/SRS²) incorrect for the region (Gujarat) for which the shapefiles were produced.

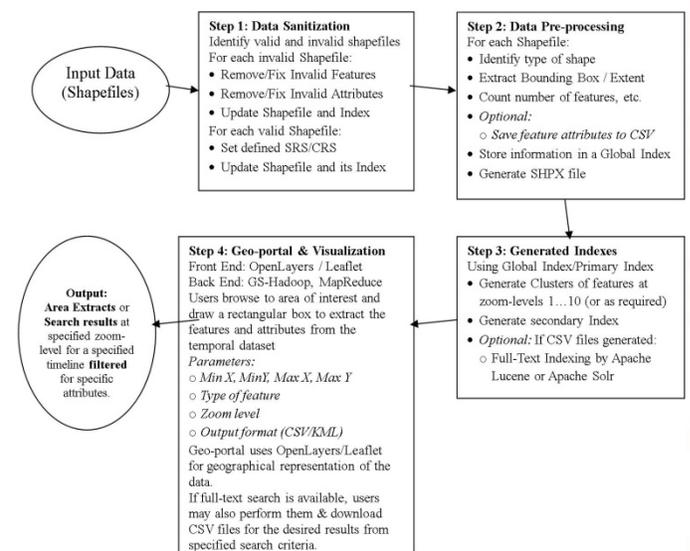


Fig 4: Major Steps/Phases in the proposed Spatio-temporal data processing model.

As these have been incorrectly marked and will not represent correct geographic location, measure of length and area they have to be in the correct EPSG that have been defined for the Indian Subcontinent or the region in consideration. EPSG Codes (EPSG: 24370 ... EPSG: 24383) for the Indian Subcontinent regions have been marked and shown in Figure 5. For

an invalid shapefile, it might also be required to remark all the vector features for accurate results in the subsequent phases. If desired to be transformed automatically, data marked for one region on earth with an incorrect projected system may not transform accurately for another region on earth. With our dataset, we fixed shapefiles with invalid features, bounds and attributes removing the invalid features and setting a correct required projection.

B. Data Pre-processing

This phase will intake shapefiles (.shp, .shx, .dbf) and provide extended shapefiles (.shpx) for use with GS-Hadoop. We also generate an index with the shapefile properties such as its bounds, type and number of features, the attribute table, etc. This information is stored in a database such as MySQL, PostgreSQL, etc. for faster regional queries and locating features having specific attributes. The attribute table is also parsed to a CSV/TSV file and is provided to Apache Lucene/Solr for full text indexing.

C. Multi-level index generation

Multi-level indexing is required owing to the requirement of indexing of millions of shapefiles. A Single index if used will

¹ EPSG: European Petroleum Survey Group has provided recommendations for geographic and projected coordinate systems, unit of measurements, etc for various regions on earth.

² CRS/SRS: Coordinate or Spatial Reference System refers to the standardized EPSG Codes.

Become a bottle neck for querying due to its size. Multi-level indexes from very coarse to very fine are required due to the availability of geo-data at various spatial resolutions. An index with the corresponding spatial resolution (zoom level) required by the user will be utilized for efficient querying and visual representation of data for the region in consideration. The attribute tables in form of CSV/TSV files from the previous phase are full-text indexed by Apache Lucene/Solr. This full text indexing provides near real-time for search queries for locating features with certain attributes. All of these 3 phases are executed for any new input data stored for the spatio-temporal system.

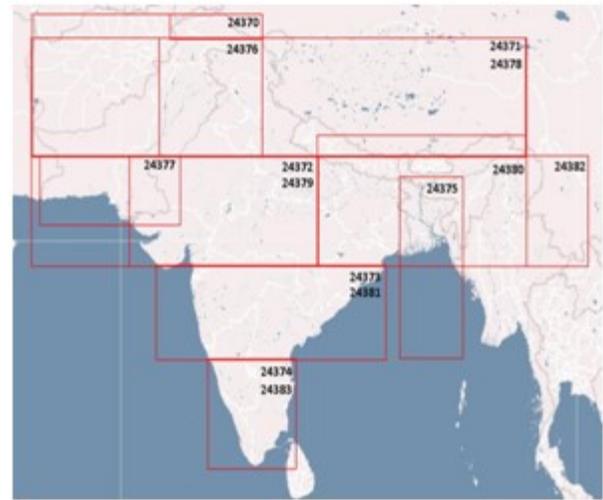


Fig 5: EPSG codes and the applicable region for the Indian Subcontinent

D. Geo-portal and Visualization

As said earlier, the best way of representing geo-spatial information is using web-maps. The visual interface of the Geo-portal allows a user to view the complete dataset with a compatible web-browser (currently supporting OpenLayers 2/3 and Leaflet). We have used OSM layers for the base map. Functions such as panning, zooming, etc are available as with OpenLayers/Leaflet and can be customized according to the visual requirements.

The user can navigate to the region of interest, explore and export the desired features. Using the index generated in Phase 2 and multi level indexes in Phase 3, only the required extended Shapefiles will be processed for visualization. For extraction, the user can specify the granularity of the desired information in form of data to be extracted for the number of zoom levels. The Figure 6 represent region extents from more than 80,000 distinct shapefiles and which takes less than a minute to render with OpenLayers and Leaflet using a desktop computer with a decent configuration.

IV. PERFORMANCE EVALUATION OF THE MODEL

The model has been deployed on private cloud environment created using HyperBox servers and clients using 10 numbers of desktop nodes. All the nodes are configured with 16 GB of RAM and have Core i7 with 8 cores clocked at 3.6 GHz base

frequency. Depending on the size of the data available to the

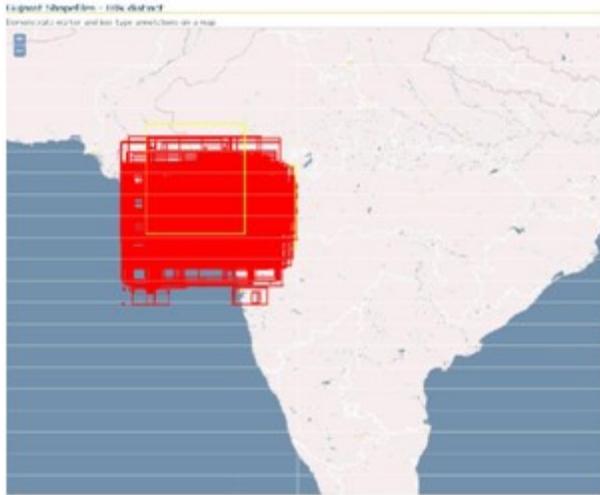


Fig 6: Bounding extents of shapfiles (more than 80 thousand) for Gujarat

We evaluated the performance of the front-end of the Geo-portal which provides visual access to the temporal data utilizing two of the most widely adopted libraries OpenLayers and Leaflet for representation. We have compared them for rendering information provided by our system in form of, JSON (Java Script Object Notation) and GeoJSON objects, representations used by the libraries.

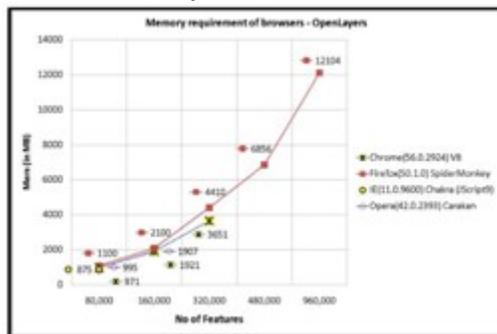


Fig 7: Memory requirement for browsers when using OpenLayers rendering library

We tested the results on four of the most commonly used web-browsers which serve as the visualization clients. The latest available versions of the browsers were used as OpenLayers and Leaflet depend on the recent functionalities provided by JavaScript, HTML5 and CSS3. Across the browsers used, Mozilla Firefox was the most reliable. Internet Explorer did not scale beyond 100,000 features for rendering features with OpenLayers while with Leaflet Internet Explorer was able to handle around 500,000 rectangle extents (features). Chrome and Opera were capped at 4GB

RAM usage even for 64 bit version of the applications. All of the readings have been averaged over 10 consecutive runs. The feature render performance (CPU time utilized using a single thread) and the amount of memory required for various use cases have been depicted in the Figures 7, 8, 9 and 10.

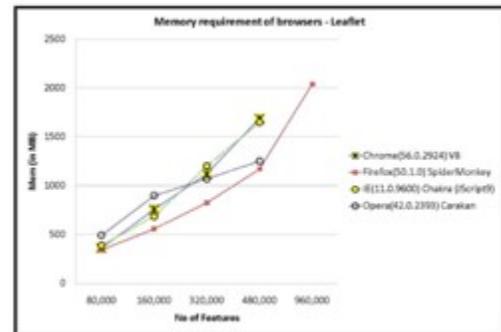


Fig 8: Memory requirement for browsers when using Leaflet rendering library

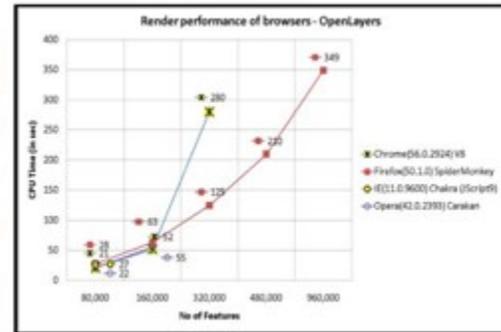


Fig 9: Rendering performance of OpenLayers library upto a billion features

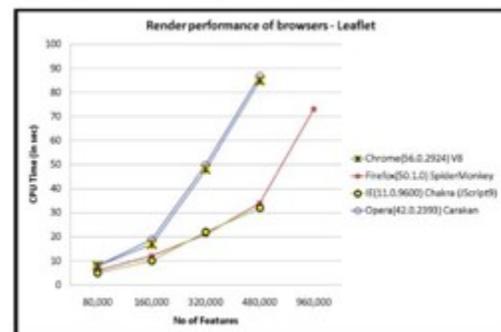


Fig 10: Rendering performance of Leaflet library upto a billion features

CONCLUSION AND FUTURE WORK

In this paper, we proposed a spatio-temporal model scalable to process millions of shapefiles totaling to billions of features. The model also provides a visual interface for near-real-time browsing of geo-data and has capabilities for extracting subsets from multiple

shapefiles for the desired region. The user can also extract features by providing filters such as known attribute information. The key part of this model is that it supports processing of shapefiles, all of which can have heterogeneous structure in form of attributes. The performance evaluation of the libraries also provide insights into the capabilities of the web-browsers and the rendering libraries which cap at certain number of features and are bound by their internal memory management in spite of much amount of memory available with the visualization client.

Currently our model is not capable of performing full-text search. In the future, we would like to extend the proposed model to support full-text search by deploying a Lucene/Solr cluster. The model currently uses Shapefiles transformed into Extended Shapefiles and uses GS-Hadoop as the backend framework. While the extended shapefile format does not include *.prj* (projection file), the format is extensible enough to include any type and any number of files for which only the header needs to be expanded. The header in the extended shapefile can further be put at the end of the file (just like ZIP) to enable easy appending of files. Our proposed framework can be extended to support such modifications and if desired to use data from shapefiles after transformation in to text, RDD's and Apache Spark can be used for faster querying.

ACKNOWLEDGMENT

We are thankful to Shri T.P. Singh, Director, Bhaskaracharya Institute for Space Applications and Geo-informatics (BISAG), Department of Science and Technology, Government of Gujarat for his support and encouragement. We also wish to express our most sincere gratitude to the system and network administrators at BISAG for providing us with the needed storage and computing facilities.

REFERENCES

- [1] Hancke, Gerhard P., Gerhard P. Hancke Jr., "The role of advanced sensing in smart cities," *Sensors* 13.1, pp. 393-425, 2012.
- [2] List of Earth Observation Satellites, <http://www.isro.gov.in/spacecraft/list-of-earth-observation-satellites>
- [3] Sehra, S.S., Singh, J., Rai, H.S., C., "A systematic study of OpenStreetMap data quality assessment," 2014 11th IEEE International Conference on Information Technology: New Generations (ITNG)," pp. 377-381, 2014.
- [4] ESRI E., "Shapefile technical description. An ESRI White Paper", 1998.
- [5] "Projection Engine (PE)," <http://support.esri.com/technical-article/000001897>
- [6] Tang, M., et al., "Locationspark: a distributed in-memory data management system for big spatial data," *Proceedings of the VLDB Endowment* 9.13, pp. 1565-1568, 2016.
- [7] "GDAL Autotest Status," <https://trac.osgeo.org/gdal/wiki/AutotestStatus>
- [8] Brovelli, M. A., et al., "Free and open source software for geospatial applications (FOSS4G) to support Future Earth," *International Journal of Digital Earth*, pp. 1-19, 2016.
- [9] Eldawy, A., Mokbel, M. F., Alharthi, S., Alzaidy, et al., "Shahed: A mapreduce-based system for querying and visualizing spatio-temporal satellite data," 31st IEEE International Conference on Data Engineering (ICDE), pp. 1585-1596, 2015.
- [10] Magdy, A., Alarabi, L., Al-Harhi, S., Musleh, M., Ghanem, et al., "Taghreed: a system for querying, analyzing, and visualizing geotagged microblogs," 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 163-172, 2014.
- [11] Alarabi, L., Eldawy, A., Alghamdi, R., et al., "TAREEG: a MapReduce-based web service for extracting spatial data from OpenStreetMap," 2014 ACM SIGMOD International Conference on Management of data, pp. 897-900, 2014.
- [12] "OpenStreetMap Data Extracts," <https://download.geofabrik.de/>
- [13] Grover, M., Malaska, T., Seidman, J., Shapira, G., "Hadoop application architectures," O'Reilly Media Inc., 2015.

- [14] Abdul, J., Alkathiri, M., Potdar, M. B.,
“Geospatial Hadoop (GS-Hadoop): An efficient
MapReduce based engine for distributed
processing of shapefiles,” IEEE International
Conference on Advances in Computing,
Communication, & Automation (ICACCA), pp. 1-
7, 2016

