Open Cloud based distributed Geo-ICT services

A Ph.D. Synopsis
Submitted to
Gujarat Technological University, Chandkheda

In Partial fulfilment
For the Award of
Ph.D. in Computer Engineering

by

Jhummarwala Abdul Taiyab Abuzar
Enrolment No: 129990907004

Under supervision of

Dr. M.B. Potdar,
Project Director,
BISAG – Gandhinagar

Index

1. Abstract

A Geographic Information System (GIS) basically 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 includes high resolution satellite sensors and imagery to simple derived data such as low resolution photographs uploaded on social networks by billions of internet users. The advancement of technology has made sensors cheap and it is easy to embed geographic location with data. Efforts such as 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 geo-spatial data by collecting terrestrial information. The data thus collected spans domains of weather forecasting, oceanography, forestry, climate, rural and urban planning, etc. Storing such large volumes of data is indeed a challenge but processing such volumes and deriving useful information which is required for planning purposes and accurate prediction required for decision making form the most important parts of the challenge. This analysis of big-geospatial data will not only provide current insights but will also enables to perform complex spatio-temporal analysis and operations to understand the underlying phenomenon.

2. Background

GIS has gone beyond typical tasks of mapping to actual application of geospatial sciences. Some of the most important routine applications include spatial analysis, digital elevation model (DEM) analysis such as line of sight and slope computations, watershed and viewshed analysis, etc. 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 include manual editing), etc. A collection of geospatial data for an organization may be stored in a geo-database for 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 format. The later is preferred for exchange of data between applications and on web while the former 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.

Publicly available dataset from OpenStreetMap project (OSM) represents vector features in XML format. It represents points, lines and polygons in the form of nodes, ways and relations. For the year 2016, this OSM data amounts to >800 GB and consists of more than 3.5 billion vector features. This is just one representation of massiveness of geo-spatial data. The analysis required to be performed on such huge datasets require large amount of storage, compute and memory and is no longer feasible with a single computer. The advancement and ease of managing IaaS (Infrastructure as a Service) resources in Cloud Computing to utilize remote infrastructure for distributed storage, processing and network capabilities 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 utilization of parallel/distributed systems and application of parallel/distributed processing techniques.

Our focus is on processing of large amount of Vector data which is available in form of Shapefiles (a collection of .prj, .shp, .shx, .dbf and other related files). The dataset available consists of more than 300,000 shapefiles and utilizing up to ~750 GB of storage. The development of a model distributed framework for processing of (vector) geo-data will enable the utilization of such large amount data for temporal analysis. The developed model framework will also relieve the geo-scientists from the complexity of distributed system and focus on insights derived from its processing.

3. Objectives of the Research Work

- On-demand access for the geo-scientists to distributed geo-processing services on request basis or programmatically without the need of specialized knowledge regarding parallel and distributed systems.
- Plan to successful implementation of Open Source and distributed GIS including a workflow interface for distributed workflows in Cloud environment

4. Hypotheses/Motivation

The root of our motivation lies in the absence of a complete distributed GIS and dependent analysis tools. GIS softwares and the accompanying libraries are desktop based and can only operate with in the limits of a single system. They neither provide required processing capability to work with huge amount of geo-spatial data nor are able to handle the volume of data available today. For e.g. the most widely used data format for storing vector data i.e. shapefile format cannot store more than 70 million points (and that is even restricted by its file size which cannot exceed 2 GB limit). It is indeed essential and important to break out from the boundaries of a single system available in terms of storage, memory and processing. There have been multi-decade research activities in the domain of parallel and distributed systems. GIS as an application domain has been lagging behind in the adoption of distributed processing. Due to the developments in programming languages, compilers and operating systems, it has now become far easier to adapt GIS for utilization of such systems and to reap the advantages of parallel and distributed systems.

There have been some developments recently in the field of distributed GIS but those mainly focus on transforming vector and raster data akin to the format that is understandable directly by the underlying distributed framework (e.g. Text for Apache Hadoop and several other similar projects). A huge amount of data is available in shapefiles. A shapefile is made up of several components of which the main shapefile (.shp) forms the most important part and stores the actual location information. Each main shapefile stores a particular type of vector entity which is either of point(s), line(s) or polygon(s) and their geographical location adhering to a particular SRS/CRS (Co-ordinate Reference Systems). Beside this main shapefile, the shapefile is also accompanied by several other files such as the index (.shx), the attribute database (.dbf), etc. The attribute database may be different for every shapefile and depends upon the standardization adopted by the creator of the shapefile. This heterogeneity in the attribute information, binary format of shapefiles and co-location of all the shapefile component files presents a huge challenge while adopting any GIS system over a distributed framework.

The utilization of Desktop GIS is mature which have yield to stable libraries such as GeoTools. Recent adaptations to MapReduce have considerably voided the requirement of a costly and specialized parallel system which presents its own challenges and limitations. Well tested set of functionality of GeoTools can be made available upon MapReduce and provisioned to users requiring analysis of large and complex geo-datasets. Our focus is to fulfill the same requirement for distributed processing of shapefiles, the most widely used geo-spatial vector data format. Apache Hadoop is best suited for development taking in view the development efforts and considering the large user-base who have also provided a plethora of extensions to support execution of variety of applications. 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 (dataset provided by BISAG).

5. Issues in utilizing Vector data in a distributed environment

Shapefiles: ESRI recommends to not include more than 1,000 features in the shapefile [2] (Refer Appendix for more information) and the file itself should not exceed 10 MB for web usage [3]. There is also a 2 GB size limit for any shapefile component, which translates to a maximum of roughly 70 million point features [4]. Practically due to large size of an HDFS block i.e., 64 MB, only a few shapefiles will ever need to be split and their data shuffled for the map reduce job but the scenario of shapefile and its components requires grouping the files (.shp, .shx and .dbf) so all of them would be available on a single node for a single task. Some techniques and extensions such as CoHadoop and Hadoop++ have been developed which allows co-locating files on Hadoop. As described further they do not serve our purpose.

XML formats: Vector data is also stored in XML variants such as KML (Keyhole Mark-up Language), GML (Geography Mark-up Language) apart from CSV and TSV formats. There is no limit on the number of features that can be stored in these formats but that imposed by the underlying storage capacity, file system and the OS. OpenStreetMap [3] started providing full planet dumps since 2012 (having 1.8 billion point features) and one of the openly available planet.osm (XML file) for year 2016 is larger than 50 GB (uncompressed: ~800 GB) and contains more than 3.5 billion point features. Iterating through this huge amount of data and processing is not only cumbersome and difficult using traditional desktop GIS applications but is also inefficient without an index. Most of the distributed GIS frameworks incorporate functionality for indexing of data but their indexing performance have neither been benchmarked nor compared with existing standalone tools capable of storing and indexing geo-data. Apache Hadoop provides an interest streaming interface for processing XML data. Only Hadoop GIS SATO from the distributed geoprocessing tools (mentioned further) utilizes Hadoop streaming but there is no functionality for processing OSM/XML data.

6. Problem Definition

Hadoop has been designed from the ground-up to process data from web crawlers which is text. Geospatial data such as those contained in shapefiles has to 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 frameworks such as Spatial Hadoop, Spatial Spark and GeoSpark. Additionally, most widely used relational database management systems such as MySQL, Postgres and SQLite incorporate spatial indexing using extensions and add-ons.

The development of our data processing model is based upon GS-Hadoop which in turn was developed and required for distributed processing of Shapefiles. The development of GS-Hadoop with an accompanied library ShapeDist to compute over our proposed Extended

Shapefile Format (SHPX) has also been discussed. The development of GS-Hadoop enabled the co-location and utilization of Shapefiles with GeoTools library. 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.

7. Review of related literature:

There have been several attempts at processing Geospatial data using Hadoop and MapReduce [7] [9] [10] [11] by first converting it into text. While each approach focuses on bringing the power of distributed and parallel computing to geo-computation, they also rely on the only text based processing capabilities of Hadoop. These approaches are application specific and focus on storage [12] [13], creating spatial indexes [14] and optimize execution of spatial queries such as joins [15], etc and cannot use shapefiles directly. Shapefiles like any other file can be split into blocks and stored on HDFS but being binary in nature, the individual blocks cannot be processed with Hadoop as the blocks individually don't convey meaningful information. There have been some extensions such as (CoHadoop, Hadoop++) which tweaks Hadoop and try co-locating similar files (in the same rack) but do not serve our purpose as processing of shapefiles require co-locating shapefile components on the same node rather than the same rack.

The best way to represent geographic data is in an interactive visual form rather than tables containing columns representing location co-ordinates. 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 (of shapefiles) from tens or hundreds of gigabytes of data to extract small subsets. Thus, distributed processing of geo-data needs to be complemented with extraction of required data from large datasets. Several systems such as SHAHED [16], TAGHREED [17] and TAREEG [18] have been proposed to deal with the issues.

8. Work Completed

1. **Setup of a Hadoop Cluster with Hadoop 1.2.1 and Hadoop 2.6.0 (50-200 nodes)**
   Included in Appendix

2. **Development of the Extended Shapefile Format (.shpx)**

   The related files .shp, .shx, .dbf (.prj and .sbn) should be available on the same host for the map-reduce task to efficiently utilize the index and related attributes. It would be possible to use a container such as .tar or .zip format to group the files and send it to the task but that will cause an additional overhead of the compression/decompression. To overcome this new extended shapefile format (.shpx) is proposed which is simple and allows to access .shp, .shx and .dbf files directly using the Memory mapped IO with java.nio package available in latest editions of Java without any overhead.

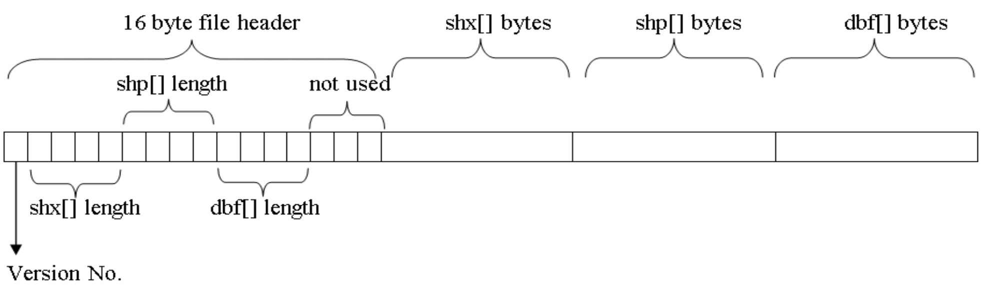
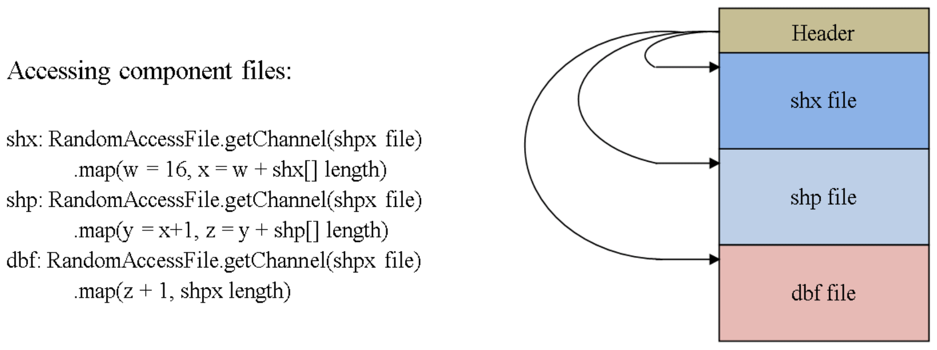


Fig. 1: Extended Shapefile Container Format (with header)


Accessing component files:

```
shx: RandomAccessFile.getChannel(shpx file)
        .map(w = 16, x = w + shx[] length)
shp: RandomAccessFile.getChannel(shpx file)
        .map(y = x+1, z = y + shp[] length)
dbf: RandomAccessFile.getChannel(shpx file)
        .map(z + 1, shpx length)
```

Fig. 2: Accessing shapefile components from an Extended Shapefile

3. Development of ShapeDist library

The issue of co-locating the shapefile components can be easily resolved using an appropriate container format such as Tar. Even after using such a format, the issue of splitting files into blocks remains and the default "copyFromLocal" or "put" will split the Tar file and upload it to HDFS. Fortunately, the recent versions of HDFS provide an API to dynamically decide the block-size while uploading the files on HDFS. ShapeDist library contains a User Defined format for .shpx files and can pass the contained .shp, .shx and .dbf file transparently to the GeoTools library.

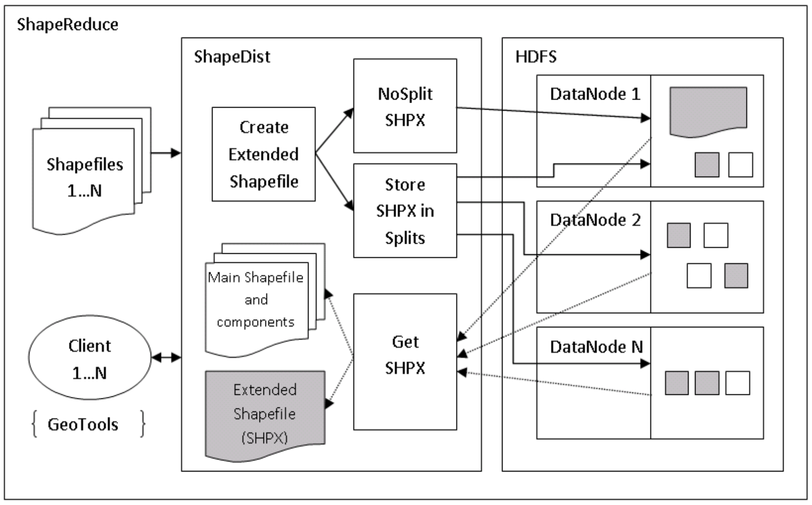


Fig. 3: Usage of the ShapeDist library

4. Developing MapReduce programs in Java to Access GeoTools functions

Included in Appendix

5. Performance and Benchmarks of GS-Hadoop and Indexing Methods

Using the ShapeDist library several MapReduce runs were executed on the cluster varying the number of Active nodes (Live nodes). The processing for the sample input of 3072 extended shapefiles (~11.3 GB) took 02:34:35 (hh:mm:ss) on a standalone computer. The sample input was processed in ~18 minutes on a cluster of 50 nodes. No considerable performance improvement was found as compared to the deployed resources for nodes > 30 for the dataset size of ~11.3 GB. Hadoop benchmarks have been included in Appendix E. The following table summarises the time taken (average in minutes) for the sample input on the Hadoop Cluster which was deployed on Hyper-V server.

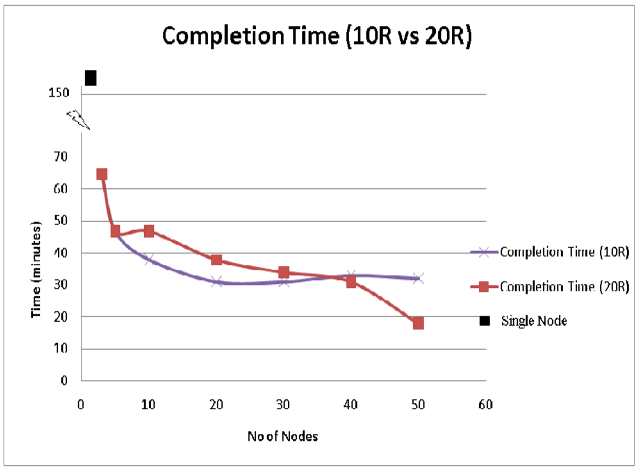


Fig. 4: Completion Time for 10 vs. 20 simultaneous Reducers (R)

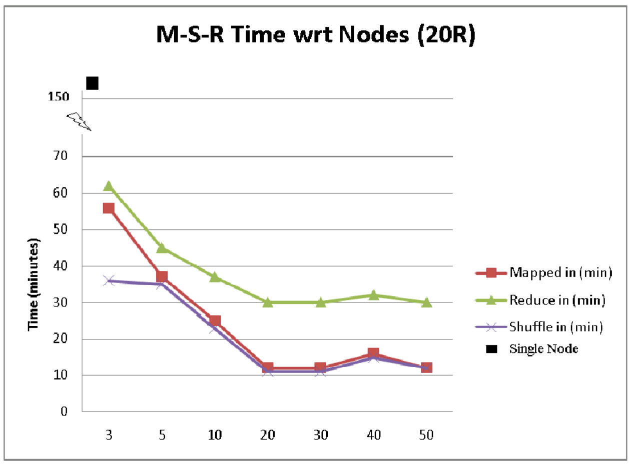
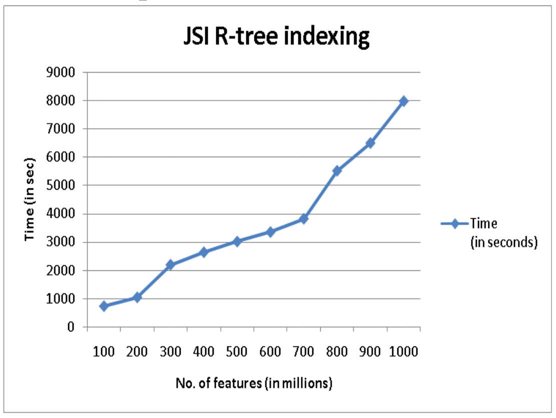


Fig. 5: Map (M), Shuffle (S) and Reduce (R) Timing w.r.t. Nodes having 20 simultaneous reducers (R)

For use with our proposed GS-Hadoop framework, several geo-data indexing libraries and frameworks were also evaluated. The following represents the indexing performance of various tools and frameworks in terms of time, memory and storage required with respect to the number of features to be indexed.

JSI (Java Spatial Index)



Spatial Hadoop

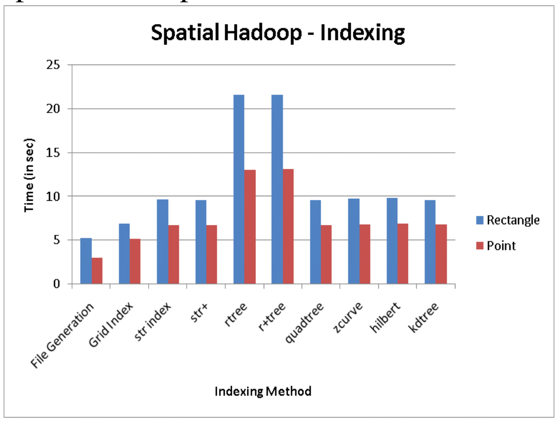
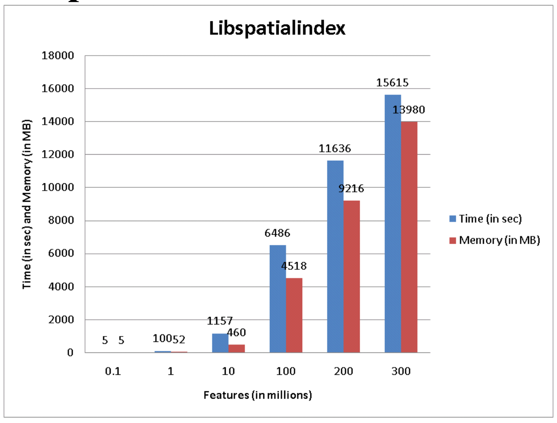


Fig. 6: (Left) JSI R-tree indexing performance and (Right) Spatial Hadoop indexing technique support and performance

Libspatialindex



Hadoop GIS SATO

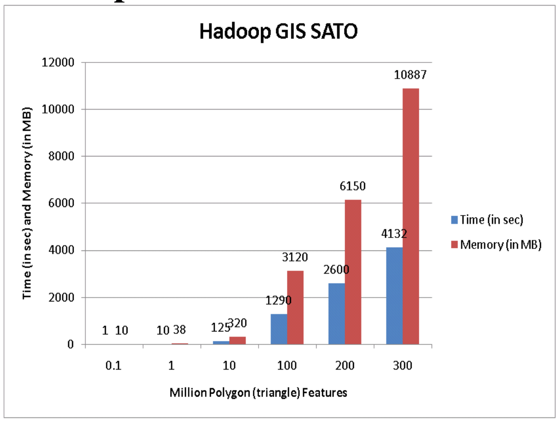
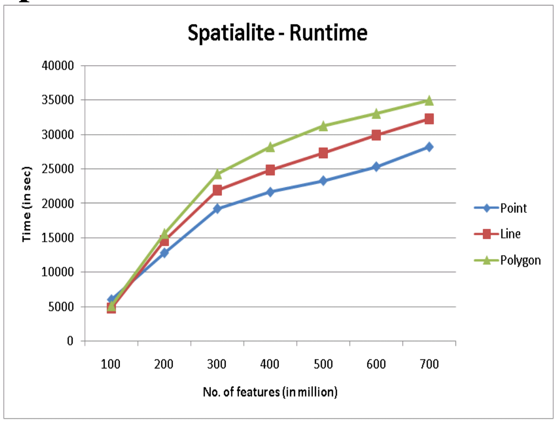


Fig. 7: (left) libspatialindex indexing performance and memory requirement and (right) Indexing performance and memory requirement of Hadoop GIS SATO

Spatialite



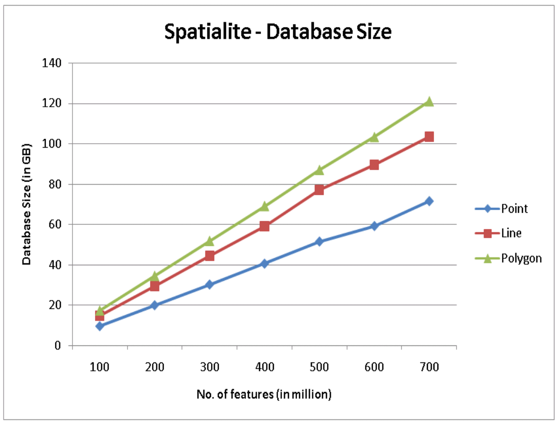


Fig. 8: (Left) Generating Index (running time) with respect to number of features in database and (Right) Growth in Spatialite database with respect to the number of features

9. Proposed Spatial Data Processing Model

A model has been devised which provides access to visual search and extraction of features and related attributes from over more than 800 GB of shapefile data containing about a million files containing more than 330,000 main shapefiles and totalling to ten of billions of features. We have also modelled a spatio-temporal data infrastructure based on the model and provide detailed solutions for problems that may arise in development of such spatial data infrastructure and processing such huge amount of heterogeneous geo-data. The main phases in the model are as described in the following figure:

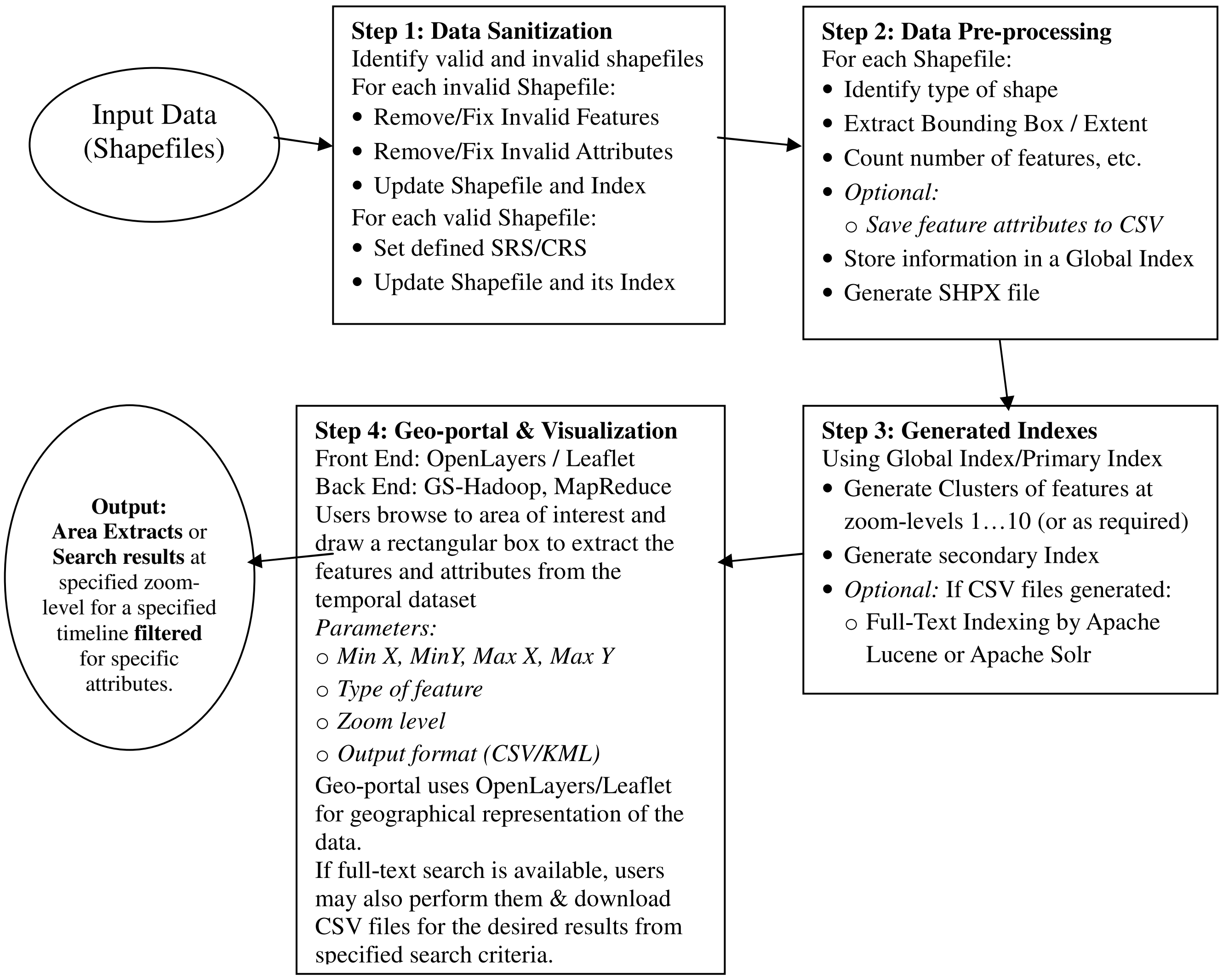

Input Data (Shapefiles)
Step 1: Data Sanitization
Identify valid and invalid shapefiles
For each invalid Shapefile:
• Remove/Fix Invalid Features
• Remove/Fix Invalid Attributes
• Update Shapefile and Index
For each valid Shapefile:
• Set defined SRS/CRS
• Update Shapefile and its Index
Step 2: Data Pre-processing
For each Shapefile:
• Identify type of shape
• Extract Bounding Box / Extent
• Count number of features, etc.
• *Optional:*
○ *Save feature attributes to CSV*
• Store information in a Global Index
• Generate SHPX file
Step 3: Generated Indexes
Using Global Index/Primary Index
• Generate Clusters of features at zoom-levels 1…10 (or as required)
• Generate secondary Index
• *Optional:* If CSV files generated:
○ Full-Text Indexing by Apache Lucene or Apache Solr
Step 4: Geo-portal & Visualization
Front End: OpenLayers / Leaflet
Back End: GS-Hadoop, MapReduce
Users browse to area of interest and draw a rectangular box to extract the features and attributes from the temporal dataset
Parameters:
○ *Min X, MinY, Max X, Max Y*
○ *Type of feature*
○ *Zoom level*
○ *Output format (CSV/KML)*
Geo-portal uses OpenLayers/Leaflet for geographical representation of the data.
If full-text search is available, users may also perform them & download CSV files for the desired results from specified search criteria.
Output: **Area Extracts** or **Search results** at specified zoom-level for a specified timeline **filtered** for specific attributes.

Fig. 9: Proposed Spatial Data Processing Model

Data Sanitisation 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. While processing with our dataset we identified about 37,521 shapefiles with EPSG (a standard format for CRS/SRS) listed in table below. 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. The EPSG identifiers and their areas for the Indian Subcontinent have been tabulated in Appendix while their corresponding geographic cover in represented in the following figure.

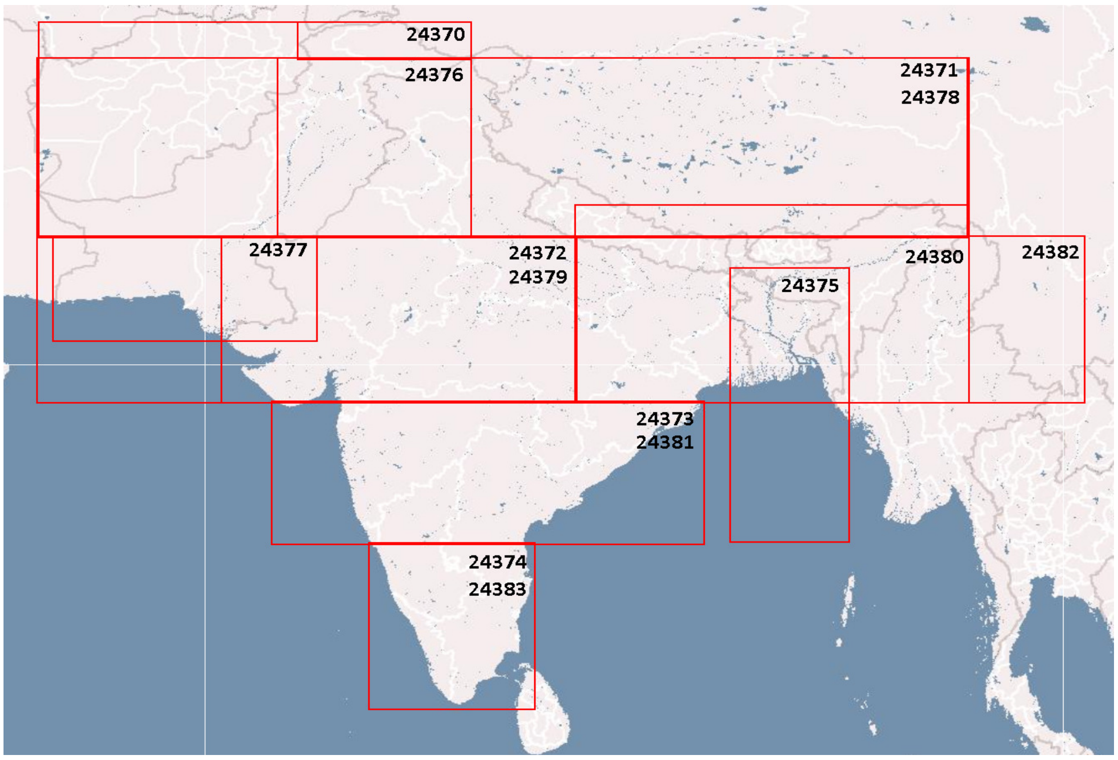
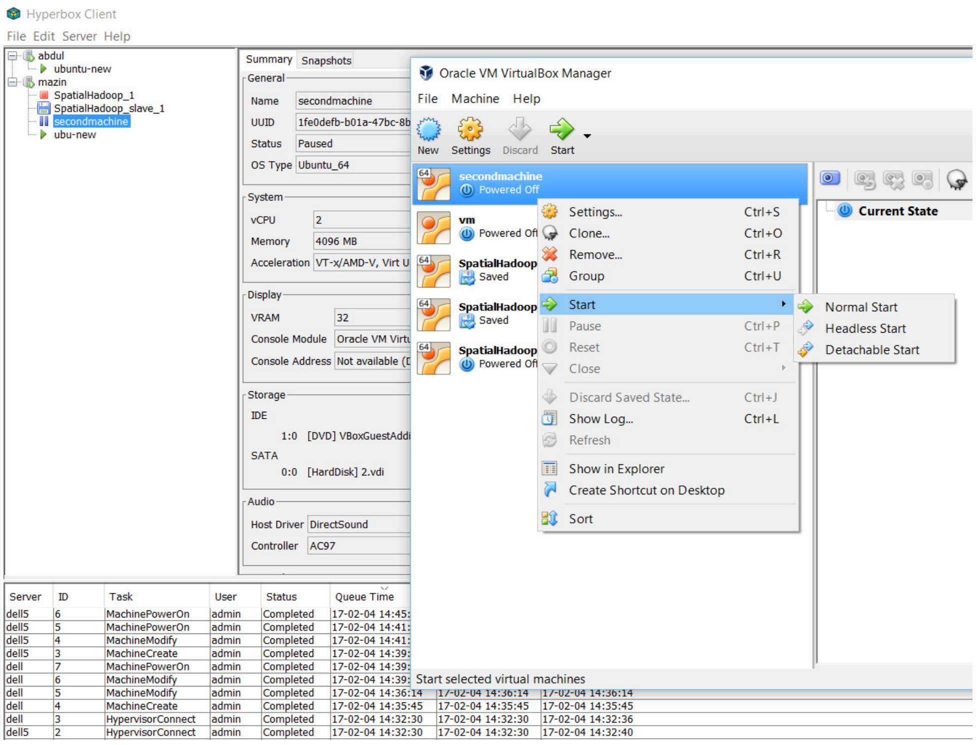


Fig. 10: EPSG for the Indian Subcontinent

In contrast with the previous implementation, this model has been deployed on private cloud environment created using Open Source HyperBox servers and client. The following figure shows a sample HyperBox client console. HyperBox uses VirtualBox API to provide a single console for multiple hosts and it relies on VirtualBox's Headless mode to run the virtual machines.



General
Name secondmachine
UUID 1fe0defb-b01a-47bc-8b
Status Paused
OS Type Ubuntu_64

System
vCPU 2
Memory 4096 MB
Acceleration VT-x/AMD-V, Virt U

Display
VRAM 32
Console Module Oracle VM Virtu
Console Address Not available (D

Storage
IDE
1:0 [DVD] VBoxGuestAddi
SATA
0:0 [HardDisk] 2.vdi

Audio
Host Driver DirectSound
Controller AC97

Server	ID	Task	User	Status	Queue Time		
dell5	6	MachinePowerOn	admin	Completed	17-02-04 14:45:		
dell5	5	MachinePowerOn	admin	Completed	17-02-04 14:41:		
dell5	4	MachineModify	admin	Completed	17-02-04 14:41:		
dell5	3	MachineCreate	admin	Completed	17-02-04 14:39:		
dell	7	MachinePowerOn	admin	Completed	17-02-04 14:39:		
dell	6	MachineModify	admin	Completed	17-02-04 14:39:		
dell	5	MachineModify	admin	Completed	17-02-04 14:36:14	17-02-04 14:36:14	17-02-04 14:36:14
dell	4	MachineCreate	admin	Completed	17-02-04 14:35:45	17-02-04 14:35:45	17-02-04 14:35:45
dell	3	HypervisorConnect	admin	Completed	17-02-04 14:32:30	17-02-04 14:32:30	17-02-04 14:32:36
dell5	2	HypervisorConnect	admin	Completed	17-02-04 14:32:30	17-02-04 14:32:30	17-02-04 14:32:40

Fig. 11: HyperBox Client with VirtualBox Manager

The visual interface of the Geo-portal allows a user to view the complete dataset with a compatible web-browser (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 requirements. The user can navigate to the region of interest, explore and export the desired features. The following figures represent region extents from more than 80,000 distinct shapefiles and it takes less than a minute to render using a desktop computer with a decent configuration.

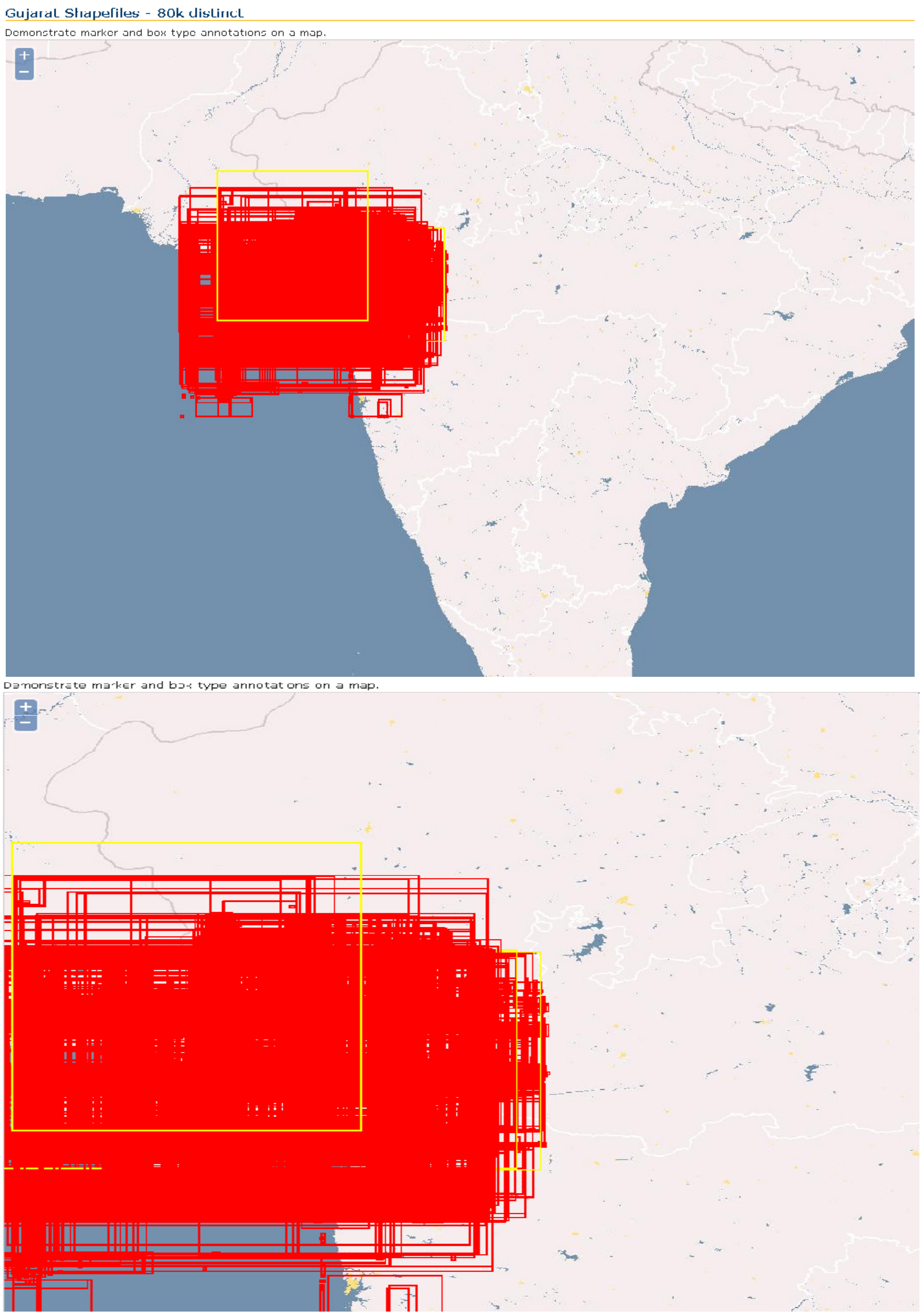

Gujarat Shapefiles - 80k distinct
Demonstrate marker and box type annotations on a map.
Demonstrate marker and box type annotations on a map.

Fig. 12: Extents of dataset (more than 80 thousand) for Gujarat

10. Performance evaluation of the model

The front-end of the Geo-portal which provides visual access to the temporal data utilizes two of the most widely adopted libraries OpenLayers and Leaflet for representation. We have also compared the performance of both the libraries for rendering the data. The data was input in the form of JSON and GeoJSON. Across the browsers used, Mozilla Firefox was the most reliable. Chrome and Opera were capped at 4GB RAM usage even for 64 bit version of the applications. 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 features. The client utilized for measuring the performance was configured with 16 GB of RAM while the CPU clocked at 3.6 GHz. All of the readings have been averaged over 10 consecutive runs. The features render performance (CPU time utilized using a single thread) and the amount of memory required for various use cases have been depicted below:

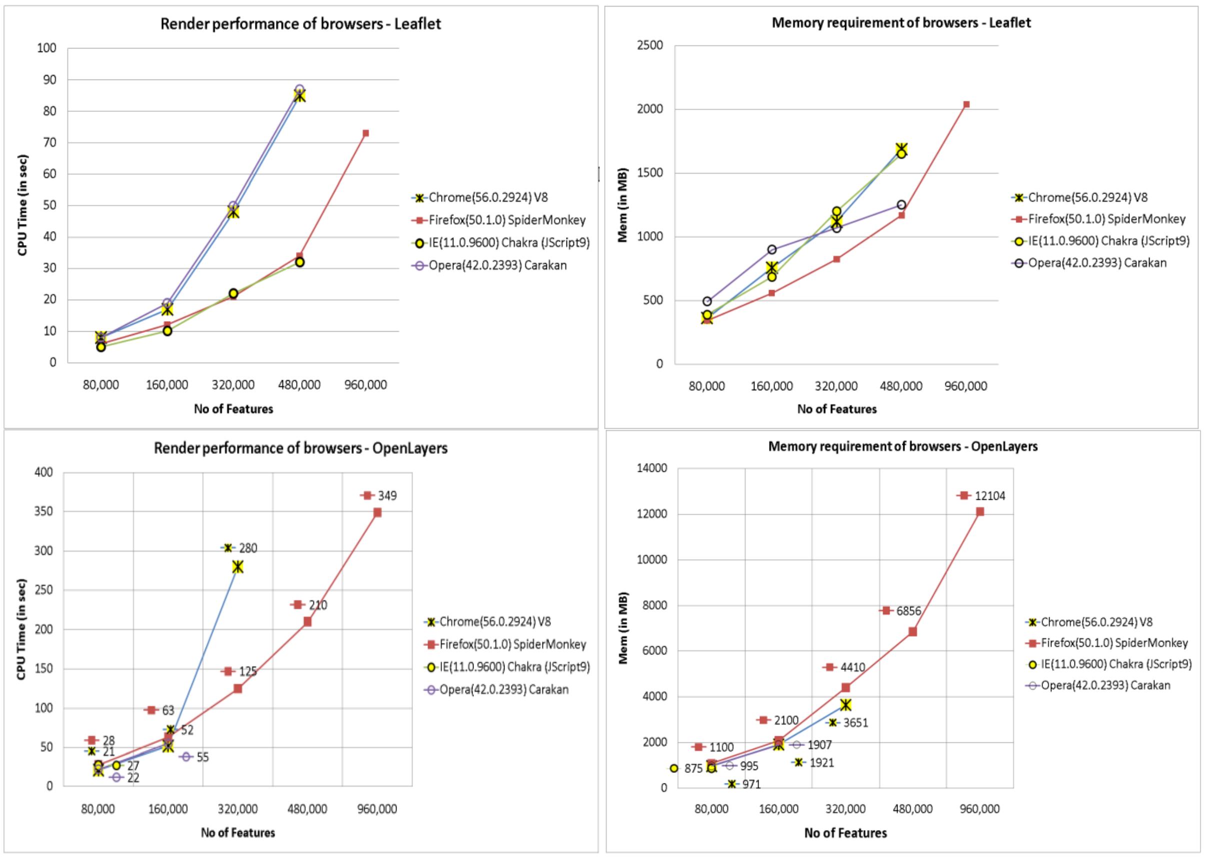


Fig. 13: Render performance and Memory requirements of OpenLayers and Leaflet for various browsers

OpenLayers did not scale up to render a million features even after utilizing a high amount of memory (using Firefox) while Leaflet easily supported rendering a million features utilizing just 2.1 GB of RAM (using Firefox). Appropriate filters need to be provided by the user to reduce the unnecessary data from Geo-portal for optimal rendering and responsiveness from the system. The proposed system provides a temporal analysis framework without requiring any installation on the clients and multiple users can access the system using a web-browser.

Note: All of the browsers utilize different JavaScript engines which form the execution base for the OpenLayers and Leaflet libraries.

11. Outcomes/Deliverables

1. Automated Scripts for managing Hadoop Cluster
   a) Managing DNS (Domain Name Service) and DHCP (Dynamic Host Configuration Protocol) information for VM's in Windows Server 2008 R2 Enterprise
   b) Scripts using PowerShell Hyper-V API for automated deployment of cluster on Windows Server 2012 R2 Datacenter Edition
   c) Scripts using PowerShell Hyper-V API for automatically managing the storage and networking of the Hadoop cluster.
2. GS-Hadoop (Geospatial Hadoop): an efficient MapReduce based engine for distributed processing of shapefiles.
   a) Extended Shapefile Format (.shpx)
   b) ShapeDist library
      1. Application for .shp to .shpx and .shpx to .shp format conversion
      2. Application for distributed processing of shapefiles
      3. Web admin for (1) and (2)
3. OpenStreetMaps OSM format to NASA HDF5 format (Available at: https://github.com/abdultz/osm2hdf5)
4. Scripts for generating synthetic data for JSI, SATO, libspatialindex, spatialite and SpatialHadoop and evaluating their indexing performance.
5. Distributed Processing Model for Spatial Data Processing and temporal visualisation.

12. Publications

1. "GeoDigViz: A spatiotemporal model for massive analysis of Shapefiles" by Jhummarwala Abdul, Sumit Prajapati, Dr. M.B. Potdar. *3rd International Conference on Computing Communication and Automation 2017, May 05-06, 2017.* IEEE.
2. "Geospatial Hadoop (GS-Hadoop): An efficient MapReduce based engine for distributed processing of Shapefiles" by A. Jhummarwala, M. Alkathiri, M.B. Potdar in *2nd International Conference on Advances in Computing, Communication and Automation, Nov 2016.* IEEE.
3. "Comparative evaluation of various indexing techniques of Geospatial vector data for processing in distributed computing environment" by A. Jhummarwala, M. Alkathiri, M. Karamta, M.B. Potdar in *ACM Compute, Oct 2016.* ACM.
4. "Geo-spatial Big Data Mining Techniques" by M Alkathiri, A Jhummarwala, M.B. Potdar in *International Journal of Computer Applications Vol.135 issue 11, pp. 28-36, Feb 2016.* Foundation of Computer Science (FCS). *doi>10.5120/ijca2016908542.*
5. "Big-Geo Data Processing using Distributed Processing Frameworks" by Shruti Thakker, Jhummarwala Abdul, M.B. Potdar in *International Journal of Scientific and Engineering Research, Vol. 6, Issue 5, May 2015.* IJSER.
6. "GeoProcessing Workflow Model for Distributed Processing Frameworks" by S Thakker, A Jhummarwala, M.B. Potdar in *International Journal of Computer Applications Vol. 113, issue 1, pp. 33-38, March 2015.* Foundation of Computer Science (FCS).
7. "Parallel and Distributed GIS for processing Geo-data: An Overview" by A Jhummarwala, M B. Potdar, P Chauhan in *International Journal of Computer Applications Vol. 106 issue 16, pp. 9-16, Dec 2014.* Foundation of Computer Science (FCS).

13.References

1. "Parallel and Distributed GIS for processing Geo-data: An Overview" by A Jhummarwala, M B. Potdar, P Chauhan In *International Journal of Computer Applications 106 (16), 9-16, 2014.*
2. ESRI Shapefile Technical Description; Address: *https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf*
3. Shapefiles- ARCGIS Online Help; Address: *https://doc.arcgis.com/en/arcgis-online/reference/shapefiles.htm*
4. Geoprocessing considerations for shapefile output; Address: *http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Geoprocessing_considerations_for_shapefile_output*
5. "CoHadoop: Flexible Data Placement and Its Exploitation in Hadoop" by Mohamed Y. Eltabakh et al. In *Proceedings of the VLDB Endowment, Vol.4 Issue 9, 575-585, 2011.*
6. "Hadoop++: making a yellow elephant run like a cheetah (without it even noticing)" by Jens Dittrich et al. In *Proceedings of the VLDB Endowment, Vol.3 Issue 1-2, 515-529, 2010.*
7. "A demonstration of SpatialHadoop: an efficient MapReduce framework for spatial data" by Ahmed Eldawy and Mohamed F. Mokbel In *Proceedings of the VLDB Endowment VLDB Endowment, Vol. 6 Issue 12, 1230-1233, 2013.*
8. Using differencing disks; Address: https://technet.microsoft.com/en-us/library/cc720381%28v=ws.10%29.aspx
9. "MRGIS: A MapReduce-Enabled High Performance Workflow System for GIS" by Q. Chen, L. Wang, Z. Shang In *eScience '08. IEEE Fourth International Conference on eScience, 646 – 651, 2008.*
10. "SATO: a spatial data partitioning framework for scalable query processing" by Azza Abouzeid et al. In *Proceedings of the VLDB Endowment VLDB Endowment Vol. 2, Issue 1, 922-933, 2009.*
11. "Haggis: turbocharge a MapReduce based spatial data warehousing system with GPU engine" by Ablimit Aji, George Teodoro, Fusheng Wang In *Proceedings of the 3rd International Workshop on Analytics for Big Geospatial Data, 15-20, 2014.*
12. "Hadoop GIS: a high performance spatial data warehousing system over mapreduce" by Ablimit Aji et al. In *Proceedings of the VLDB Endowment VLDB Endowment, Vol. 6, Issue 11, 1009-1020, 2013.*
13. "HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads" by Hoang Vo, Ablimit Aji, Fusheng Wang In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, 545-548, 2014.*
14. "HQ-Tree: A distributed spatial index based on Hadoop" by Feng Jun et al. In *China Communications, Vol. 11, Issue 7, 128 – 141, 2014.*
15. "SJMR: Parallelizing spatial join with MapReduce on clusters" by Zhang, S., et al. In *IEEE International Conference on Cluster Computing and Workshops, 1-8, 2009.*
16. "SHAHED: A MapReduce-based System for Querying and Visualizing Spatio-temporal Satellite Data" by Ahmed Eldawy et al. In *IEEE 31st International Conference on Data Engineering*, pp. 1585-1596, 2015.
17. "Demonstration of Taghreed: A System for Querying, Analyzing, and Visualizing Geotagged Microblogs" by Amr Magdy et al. In *IEEE 31st International Conference on Data Engineering*, pp. 1416-1419, 2015.
18. "TAREEG: A MapReduce-Based Web Service for Extracting Spatial Data from OpenStreetMap" by Louai Alarabi et al. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pp. 897-900, 2014.

Appendices have not been included for brevity. Please email on abdul.zummerwala@yahoo.co.in if needed.