SpatialHadoop MapReduce layer extension and performance analysis

Xiang Ji[1] Siyuan Guo[1] Zhuo Chen[1]
Chengjun Yuan[1] Fabrice Mizero[1]

[1]University of Virginia, VA, USA
[1]{xj4hm, sg5jn, zc6yn, cy3yb, fm9ab}@virginia.edu

ABSTRACT

1. INTRODUCTION

As the web grew in the in the late 1900s and early 2000s search engines and indexes were created to help locate relevant information amid the text-based content. In the early years, search results really were returned by humans. But as the web grew from dozens to millions of pages, automation was needed. Web crawlers were created, many as university-led research projects, and search engine start-ups took off (Yahoo, AltaVista, etc.). One of the open source project called NUTCH. The original team want to invent a way to return web search results faster by distributing data and calculations across different computers so multiple tasks could be accomplished simultaneously. And after several years the Nutch project divided by yahoo to two separate parts, the web crawler part remained as Nutch, and the distributed storage part became Hadoop.

Hadoop is an open-source software framework for storing data and running, applications on clusters of commodity hardware. And Hadoop can provide any kind of data. Open source give Hadoop the ability can be created and maintained by a network of developers from around the globe. Hadoop currently processes large amounts of data using multiple low-cost computers for fast results. One of the top reason why Hadoop widely used is since the ability to store and process huge amount of data. And since Hadoop can handle varieties constant increasing data, which means it can handle social media and internet thing really well. On the other hand, user don't have to processing data before store it. You can store as much data as you want and decide how to use it later. which includes unstructured data like text, images and videos. Another big benefits of using Hadoop is the ability of fault tolerance, if a node goes down, jobs are automatically redirected to other nodes to make sure the distributed computing does not fail. Last but not least, Hadoop has a great scalability, user can easily grow Hadoop system by adding more more nodes.

Basic component of Hadoop including: Hadoop common - the libraries and utilities used by other Hadoop modules. Hadoop Distributed File System (HDFS) - the Java-based scalable system that stores data across multiple machines without prior organization. MapReduce - a software programming model for processing large sets of data in parallel. YARN - resource management framework for scheduling and handling resource requests from distributed applications. (YARN is an acronym for Yet Another Resource Negotiator.)

Software component that run on top of Hadoop: Pig- A platform for manipulating data stored in HDFS that includes a compiler for MapReduce programs and a high-level language called Pig Latin. It provides a way to perform data extractions, transformations and loading, and basic analysis without having to write MapReduce programs. Hive - a data warehousing and SQL-like query language that presents data in the form of tables. Hive programming is similar to database programming. (It was initially developed by Facebook.)

As the developing of mobile technology and enhancing of internet speed. The spatial data is increasing explosively. Spatial data also known as geospatial data or geographic information it is the data or information that identifies the geographic location of features and boundaries on Earth, such as natural or constructed features, oceans, and more. Spatial data is usually stored as coordinates and topology, and is data that can be mapped. Currently, Geographic information system is the most popular geographical information system, the system designed to capture, store, manipulate, analyze, manage, and present all types of spatial or geographical data. Spatial data is the basis of GIS and other geology applications. With the advancements of data acquisition techniques, large amounts of geospatial data have been collected from multiple data sources, such as satellite observations, remotely sensed imagery, aerial photography, and model simulations. The geospatial data are growing exponentially to PB (Petabyte) scale even EB (Exabyte) scale . As this presents a great challenge to the traditional database storage, especially in terms of vector spatial data storage due to its complex structure, the traditional spatial database storage is facing a series of questions such as poor scalability and low efficiency of data storage. Consequently, the challenging of spatial data has cause several vital shortcomings. First, only spatial data consist of one of four types of graphic primitive, namely, points, lines, polygons, or pixels can suit in the GIS, other data type can not be used in GIS. Secondly the data themselves can also cause some problems. Much historical data will be taken from historical maps which may not be accurate, and the representation of features from these maps in the GIS at best will only be as accurate as the original source. In reality they are likely to be worse, as new errors are added when the data are captured. Many of the clues about the accuracy of the original source will be lost when the data are captured. Thirdly, Being on-top of Hadoop, MapReduce programs defined through map and reduce cannot access the constructed spatial index. Hence, users

cannot define new spatial operations beyond the already supported ones, range query and self-join.

In this paper we introduce spatial Hadoop, SpatialHadoop is an open source MapReduce extension designed specifically to handle huge datasets of spatial data on Apache Hadoop. SpatialHadoop is shipped with built-in spatial high level language, spatial data types, spatial indexes and efficient spatial operations. Spatial Hadoop has overcome the shortcoming of GIS. (1) [9] SpatialHadoop is build-in Hadoop consequently, Spatial Hadoop inherit nearly all feature of Hadoop (2) SpatialHadoop is different from Hadoop, it support different indexing methods, e.g grid, R tree, and R+ tree, and (3) SpatialHadoop users can interact with Hadoop directly to develop the spatial function that they want to develop. This is in contrast to Hadoop-GIS and other systems that cannot support such kind of flexibility, on the other hand, traditional Hadoop is very restricted in the function they support, however SpatialHadoop has a really well scalability and user friendly interface. All of the spatial index and operation can be edit in the source file, relatively stable than the traditional Hadoop, since the SpatialHadoop is existing as an patch in the Hadoop, any change in the SpatialHadoop will not affect the ground of whole file.

In this paper, we mainly use two real dataset, OpenStreetMap and Tigger. OpenStreetMap (OSM) [3], lunched in 2004, is a collaborative community project to create a free editable map of the world. It is considered as the Wikipedia project for maps, where the community can help in building the maps around the world OpenStreetMap has over 1.6 million registered users, where around thirty percent of them have made actual contributions to the maps . As it stands now, Open-StreetMap has a very high accuracy that is comparable to proprietary datasources . OpenStreetMap whole world dataset is free and accessible as an XML 500 GB file called Planet.osm, updated on weekly basis. OpenStreetMap consist of following datatype (1) Node, which is defined as a point in the space associated with a node identifier, latitude and longitude coordinates, (2) Way, which represents a line between two nodes, and associated with the way identifier and the two nodes identifiers of the two end points of the line. The line could be simply a road segment, part of a boundary of a building, city/country boundary, or part of a lake contour (3) Relation, which represents the relation between nodes, ways, or even other relation, and is used to express polygons. For example, to express the boundaries of a certain lake, the nodes need to be defined, then the ways that connect nodes to each other, then a relation that connects the ways together to express the lake boundary.

TIGER/Line files are a digital database of geographic features, such as roads, railroads, rivers, lakes, political boundaries, and census statistical boundaries, covering the entire United States. The Tiger database contains information about these features, such as their location in latitude and longitude, their names, the types of features, address ranges for most streets, the geographic relationship to other features, and other related information.

2. RELATED WORK

A paper [6] written by Ahmed Eldawy describes and presents CG Hadoop, which has two version, one is for the Apache Hadoop System, one is for Spatial Hadoop System. The performance of CG Hadoop is improved greatly. It is a kind of new algorithms and the improvement is brought by the algorithms. What we design is to add spatial index to improve the ability that hadoop processes spatial data.

A paper named CloST: A Hadoop-based Storage System for Big Spatio-Temporal Data Analytics [15] presented a design and implementation of CloST, which is a scalable big spatio-temporal data storage system to support data analytics using Hadoop. And author presented it to avoid scan the whole dataset when a spatio-temporal range is given. Additionally, the author presented a noval data model which has special treatments on three core attributes including an object id, a location and a time. Based on the model this paper presented, CloST hierarchically partitions data using all core attributes which enables efficient parallel processing of spatio-temporal range scans. Besides, the author presented a new data storage structure which can reduce the size, and propose a new algorithms to add new data to the system. However, the performance can still improve by adjust map reduce layer. Our work in map reduce layer can improve the performance of SpatialHadoop farther.

Another paper named GeoSpark: A Cluster Computing Framework for Processing Large-Scale Spatial Data [17] , which introduces GeoSpark an in-memory cluster computing framework for processing large-scale spatial data. GeoSpark consists three layers, they are Apache Spark Layer, Spatial RDD Layer and Spatial Query Processing Layer. Apache Spark Layer provides basic Spark functionalities that include loading and storing data to disk as well as regular RDD operations, which is like SpatialHadoop's operation layer. Spatial RDD Layer consists of three novel Spatial Resilient Distributed Datasets (SRDDs) and it extends regular Apache Spark RDDs to support geometrical and spatial objects. Besides, GeoSpark provides a geometrical operations library which can accesses Spatial RDDs to perform basic geometrical operations, which makes system users can use the newly defined SRDDs to effectively develop spatial data processing programs in Spark.The Spatial Query Processing Layer this paper presented efficiently executes spatial query processing algorithms, such as range query, knn and join. Additionally, Users can also GeoSpark to create a spatial index which boosts spatial data processing performance in each SRDD partition. And the experiments show that GeoSpark achieves better run time performance than its Hadoop-based counterparts, such as SpatialHadoop.

Another paper named PAIRS: A scalable geo-spatial data analytics platform [11]. This paper presented a new platform called Physical Analytics Integrated Repository and Services (PAIRS), which enables rapid data discovery. And PAIRS also provides a foundation for developing custom analytics.

A paper named Enabling Spatial Big Data via CyberGIS: Challenges and Opportunities [10] , which presented that CyberGIS seeking to synthesize spatial analysis.

A paper named Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce [1] , this paper presents Hadoop-GIS a scalable and high performance spatial data warehousing system for running large scale spatial queries on Hadoop. Through spatial, partitioningHadoop-GIS supports multiple types of spatial queries on MapReduce , customizable spatial query engine RESQUE, implicit parallel spatial query execution on MapReduce, and effective methods for amending query results through handling boundary objects. It also utilizes global partition indexing as well as customizable on demand local spatial indexing to achieve efficient query processing. And it is also integrated into Hive to support declarative spatial queries with an integrated architecture. In this paper, the authors make some experiments which have demonstrated the high efficiency of Hadoop-GIS on query response and high scalability to run on commodity clusters.

A paper named Pigeon: A Spatial MapReduce Language [8] is motivated by the huge amounts of spatial data collected everyday, and it focus more on MapReduce frameworks. The example it uses is Hadoop, which have become a common choice to analyze big spatial data for scientists and people from industry. And user prefer to use high level languages to deal Hadoop for simplicity. Additionally these languages are designed for primitive non-spatial data and have no support for spatial data types or functions. Pigeon is implemented through user defined functions (UDFs) making it easy to use and compatible with all recent versions of Pig. Our work still used Pigeon as our language layer to operate Spatial Hadoop. This

language also allows it to integrate smoothly with existing non-spatial functions and operations such as Filter, Join and Group By.

A paper named Spatial Queries Evaluation with MapReduce [19] introduces Spatial queries include spatial selection query, spatial join query, nearest neighbor query. Most of spatial queries are computing intensive and individual query evaluation may take even hours. Parallelization seems a good solution for such problems. However, parallel programs must communicate efficiently, balance work across all nodes, and address problems such as failed nodes. Additionally the author presented performance evaluation for the several spatial queries and prove that MapReduce is also appropriate for small scale clusters and other computing intensive applications, which is to provide a efficient data computing and management capabilities.

A paper named Implementing WebGIS on Hadoop: A Case Study of Improving Small File I/O Performance on HDFS [13] . Under the situation that Hadoop framework has been widely implemented in various clusters to build large scale, high performance systems. However, it has some drawbacks such as Hadoop distributed file system (HDFS) is designed to manage large files and suffers performance penalty while managing a large amount of small files. As a result, may application like web application, for example, WebGIS, may not get benefits from Hadoop. In this paper, the author presents an approach to optimize I/O performance of small files on HDFS. The main idea is to combine smaller file into larger file to reduce file number and build index for each file. Additionally, this paper still propose some features like grouping neighboring files and reserving several latest version of data is like to meet the characteristics of WebGIS access patterns. And the experiments of this paper show the improvment of performance.

A paper named Research on Vector Spatial Data Storage Schema Based on Hadoop Platform [20] is related with the spatial data storage. Under the situation that cloud computing technology is changing the mode of the spatial information industry. Since Hadoop platform provides easy expansion, high performance, high fault tolerance and other advantages,cloud computing technology is applied and provided new ideas for it. In this paper, the author presented that novel vector spatial data storage schema based on it. This method can solve the problems that how to use cloud computing to manage spatial data. Firstly, this vector spatial data storage schema is designed based on column-oriented storage structures and key/value mapping to express spatial topological relations. Besides, this paper designs middleware and merge with vector spatial data storage schema in order to directly store spatial data and present geospatial data access refinement schemes based on GeoTools toolkit. Last but not least, the author verify these storage scheme to through Hadoop experiments.

A paper named sksOpen: Efficient Indexing, Querying, and Visualization of Geo-spatial Big Data [14], this paper provides a indexing, querying, and visualization of Geo-spatial Big Data. Since the performance of indexing and querying of location-based data is a critical quality of service aspect. Spatial indexing is not available for end-users and it is typically time-consuming. To solve this problem, this paper presents a open source online Indexing and Querying System for Big Geospatial Data, sksOpen. This paper provides ergonomic visualization of query results on interactive maps to facilitate the user's data analysis.

A paper named An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics [16] tries to solve a problem related to Hadoop. Bioinformatics researchers have to face analysis of ultra large-scale data sets, a problem that will only increase at an alarming rate in coming years. Recent developments in open source software, that is, the Hadoop project and associated software, provide a foundation for scaling to petabyte scale data warehouses on Linux clusters, providing fault-tolerant parallelized analysis on such data using a programming style named MapReduce. This paper is concise and clear and describe what the author try to do efficiently and effectively, we are more familiar with MapReduce of Hadoop. The problem we try to solve is to increase the performance of Hadoop when it deal with spatial data instead of bioinformatics data.

A paper named Efficient big data processing in Hadoop MapReduce [5] focus on different data management techniques, going from job optimization to physical data organization like data layouts and indexes, and highlight the similarities and differences between Hadoop MapReduce and Parallel DBMS. And this paper also show and explain the static physical execution plan of Hadoop MapReduce and how it affects job performance. We learn some details about Hadoop's MapReduce and we combine some characteristics about spatial data to improve the ability of MapReduece layer to process spatial big data.

A paper named Voronoi-based Geospatial Query Processing with MapReduc [2] introduces Geospatial queries, which have been used in a wide variety of applications such as decision support systems, profile-based marketing, bioinformatics and GIS. However, some existing query-answering approaches only apply to the systems with limited parallel processing capability, far from that of the cloud-based platforms. In this paper, the authors study the problem of parallel geospatial query processing with the MapReduce programming model and propose a method that creates a spatial index, Voronoi diagram, for given data points in 2D space and enables efficient processing of a wide range of GQs.

A paper named Efficient Spatial Query Processing for Big Data [12] focus on spatial query. Spatial queries are widely used in many data mining and analytics applications. However, a huge and growing size of spatial data makes it challenging to process the spatial queries efficiently. In this paper, authors presents a lightweight and scalable spatial index for big data stored in distributed storage systems. This paper's work is highly related with what we plan to do. SpatialHadoop we work on in this paper is an open source MapReduce extension designed specifically to handle huge datasets of spatial data on Apache Hadoop. SpatialHadoop is shipped with built-in spatial high level language, spatial data types, spatial indexes and efficient spatial operations. Spatial Hadoop has overcome the shortcoming of GIS. (1) [9] SpatialHadoop is build-in Hadoop consequently, Spatial Hadoop inherit nearly all feature of Hadoop (2) SpatialHadoop is different from Hadoop, it support different indexing methods, e.g grid, R tree, and R+ tree, and (3) SpatialHadoop users can interact with Hadoop directly to develop the spatial function that they want to develop. This is in contrast to Hadoop-GIS and other systems that cannot support such kind of flexibility, on the other hand, traditional Hadoop is very restricted in the function they support, however SpatialHadoop has a really well scalability and user friendly interface. All of the spatial index and operation can be edit in the source file, relatively stable than the traditional Hadoop, since the SpatialHadoop is existing as an patch in the Hadoop, any change in the SpatialHadoop will not affect the ground of whole file. We consider that we can improve the performance of Hadoop processing spatial big data further.

A paper named A Demonstration of SpatialHadoop: An Efficient MapReduce Framework for Spatial Data [7] presents SpatialHadoop as the first full-fledged MapRe- duce framework with native support for spatial data. It is closely related to our work. it focus on the MapReduce layer, and we try to improvement it MapReduce layer further in our work by dmodify some code of the program.

A paper named SksOpen: Efficient indexing, querying, and visualization of geo-spatial big data [14] introduce a method to deal with geo-spatial data. Spatial indexing is typically time-consuming and is not available to end-users. To address this challenge, the au-

thors present open-sourced an Online Indexing and Querying System for Big Geospatial Data, sksOpen. This paper can bring us some inspiration about spatial index, however, this paper dose not focus on MapReduce layer of Hadoop, which means what we plan to do can improve the performance of processing spatial big data further.

A paper named TAREEG: A MapReduce-Based System for Extracting Spatial Data from OpenStreetMap [3] presents TAREEG; a web-service that makes real spatial data, from anywhere in the world, available at the fingertips of every researcher or individual. TAREEG gets all its data by leveraging the richness of OpenStreetMap data set; the most comprehensive available spatial data of the world. Yet, it is still challenging to obtain OpenStreetMap data due to the size limitations, special data format, and the noisy nature of spatial data. Since real spatial data are not easily available for most of the world. This hinders the practicality of many research ideas that need a real spatial data for testing and experiments, however, such data is prohibitively expensive to build or buy for academia or individual researchers. That is the motivation of this paper. Due to the size limitations, the most comprehensive available spatial data of the world is still challenging to obtain OpenStreetMap data , special data format, and the noisy nature of spatial data. TAREEG employs MapReduce-based techniques to make it efficient and easy to extract OpenStreetMap data in a standard form with minimal effort. Experimental results show that TAREEG is highly accurate and efficient.

A paper named Large-Scale Spatial Data Processing on GPUs and GPU-Accelerated Clusters [18] reports works on data parallel designs of spatial indexing, spatial joins and several other spatial operations, including polygon rasterization, polygon decomposition and point interpolation. The data parallel designs are further scaled out to distributed computing nodes by integrating single-node GPU implementations with High-Performance Computing (HPC) toolset and the new generation in-memory Big Data systems such as Cloudera Impala.

3. PRELIMINARIES

3.1 Background

Spatial data is as known as geospatial data or geographic information it is the data or information that identifies the geographic location of features and boundaries on Earth, such as natural or constructed features, oceans, and more. Spatial data is usually stored as coordinates and topology, and is data that can be mapped. Generally speaking, spatial data represents the location, size and shape of an object on planet Earth such as a building, lake, mountain or township. Spatial data may also include attributes that provide more information about the entity that is being represented. Geographic Information Systems (GIS) or other specialized software applications can be used to access, visualize, manipulate and analyze geospatial data.

Support of high performance queries on large volumes of spatial data becomes increasingly important in many application domains, including geospatial problems in numerous fields, location based services, and emerging scientific applications that are increasingly data-intensive and compute-intensive. There are two major challenges for managing and querying massive spatial data to support spatial queries: the explosion of spatial data, and the high computational complexity of spatial queries.

Apache Hadoop is an open-source software framework for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common and should be automatically handled by the framework.

The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part called MapReduce. Hadoop splits files into large blocks and distributes them across nodes in a cluster. To process data, Hadoop transfers packaged code for nodes to process in parallel based on the data that needs to be processed. This approach takes advantage of data locality to allow the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking.

The base Apache Hadoop framework is composed of the following modules: (i) Hadoop Common contains libraries and utilities needed by other Hadoop modules; (ii) Hadoop Distributed File System (HDFS) is a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster; (iii)Hadoop YARN is a resource-management platform responsible for managing computing resources in clusters and using them for scheduling of users' applications; and (iv)Hadoop MapReduce is an implementation of the MapReduce programming model for large scale data processing.

SpatialHadoop [9], a MapReduce Framework for Spatial Data, is designed and implemented by Dr.Eldawy in 2015. SpatialHadoop is an open source MapReduce extension designed specifically to handle huge datasets of spatial data on Apache Hadoop. It is shipped with built-in spatial high level language, spatial data types, spatial indexes and efficient spatial operations. It provides spatial data types to be used in MapReduce jobs including point, rectangle and polygon. It also adds low level spatial indexes in HDFS such as Grid file, R-tree and R+-tree. Some new InputFormats and RecordReaders are also provided to allow reading and processing spatial indexes efficiently in MapReduce jobs. SpatialHadoop also ships with a bunch of spatial operations that are implemented as efficient MapReduce jobs which access spatial indexes using the new components. Developers can implement myriad spatial operations that run efficiently using spatial indexes.

3.2 Problem Definition

SpatialHadoop can handle large amount of spatial dataset processing since it leverages the distributed big data computation framework Hadoop and MapReduce [4] model. Based on users' own needs, they can specify and develop the corresponding operations such as range query, k-nearest neighbor and spatial join. After the query operation in SpatialHadoop system, a result report will be generated under the HDFS. However, when spatial data scales to large dataset, this spatial data operations is still time consuming. To improve the query speed performance of spatial data processing, the under-layer functionalities can still be optimized.

The object of this work is to extend the MapReduce layer of SpatialHadoop framework for improving the time efficiency of spatial data processing.

3.3 Solution Architecture

The architecture of the SpatialHadoop system is given in Figure 1.

As Fig. 1 presents, SpatialHadoop cluster use the similar computation model as Hadoop. The cluster consists of a master node that breaks a MapReduce job into smaller tasks, then a bunch of slave nodes process the smaller tasks individually. Based on the original SpatialHadoop design [9], there are mainly two type of users can use this system. (i) Casual users who do not know much about the computing and programming can access this system through a spatial language to process their datasets. (ii) Developer or data scientists who have a deeper understanding of the system and can implement their new spatial operation based on what they need, or they can just simply use the SpatialHadoop command line started

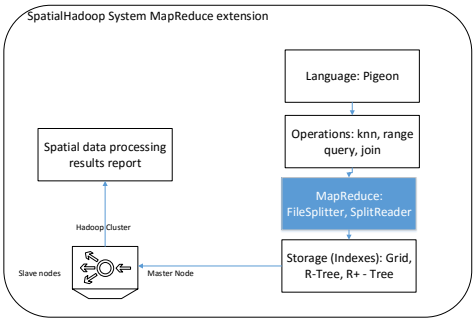

Figure 1: SpatialHadoop system with extension architecture

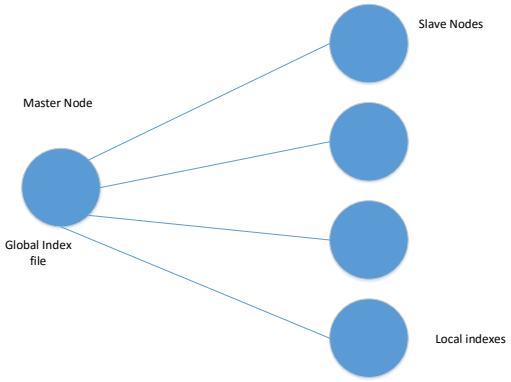

Figure 2: SpatialHadoop two layer indexing

with `shadoop`. Basically, it is a compile binary executable file give users access to the spatial operations by commands.

The language layer provides a simple high level SQL-like language Pigeon [8], which is complant spatial data types like point and Polygon, and spatial operations like overlap and touches for simplifying spatial data processing.

The operations layer encapsulates the implementation of various spatial operations that take advantage of spatial indexes and the new components in the MapReduce layer. The most common three spatial data operations provided are range query, kNN, and spatial join. In this paper, the performance measurements were mainly evaluated through the kNN query.

The MapReduce mainly has two components to allow MapReduce program to access the spatial index structures. The SpatialFileSplitter exploits the global index to prune file blocks that is not the possible query results, and SpatialRecorderReader exploits local indexex to efficiently retrieve a partial answer from each block. We improved the SpatialFileSplitter with a new class called `FileSplitUtil` which provides a more flexible way to split the big spatial data file block and combine them based on knn query accuracy (With different k values).

Since inputs files in Hadoop are non-indexed heap files, the performance is limited as the input scales. The SpatialHadoop employs spatial index structures within Hadoop Distributed File System (HDFS) as a means of efficient retrieval of spatial data. Indexing in SpatialHadoop is the main reason in its better time efficiency over normal Hadoop. To overcome the challenges of building index structures in Hadoop, the SpatialHadoop system uses a two-layers indexing approach of global and local index as shown in Fig. 2. Each index contains one global index stored in the Master node, that partitions data across a set of partitions, stored in slaves nodes. This indexing organization lends itself to MapReduce programming, where the local indexes can be processed in a MapReduce job. And the small size of local indexes allows each one to be bulk loaded in memory and written to a file in an append-only manner. Also, indexing in SpatialHadoop adapts HDFS to accommodate general purpose standard spatial index structures, namely, Grid file, R-tree, R+-Tree, and use them to support to support many spatial many spatial queries written in MapReduce. In this paper, we mainly use the Grid as the default indexing method for evaluation purpose.

3.4 Data Structures

In terms of data structure used in this paper, three indexing data structure are introduced as the previous section. While the main three phases of indexing, which is partitioning, local indexing, and global indexing, several steps are done by the indexing system. Certain algorithm are used to decide the number of partitions and the partitions boundaries. For the purpose of building requested local index on the data contents of each physical partition. This is implemented as a `reduce` function that takes the small records and stores the in a spatial index written in a local index file. Each local index fits in one HDFS block about 64 MB. Mainly, two type of data structure are used. The first is Grid indexing structure, the second is R tree.

Grid indexing is a regular tessellation of a manifold or 2-D surface that divides it into a series of contiguous cells, which can then be assigned unique identifiers and used for spatial indexing purposes. A wide variety of such grids have been proposed or are currently in use, including grids based on "square" or "rectangular" cells, triangular grids or meshes, hexagonal grids and grids based on diamond-shaped cells. In practice, construction of grid-based spatial indices entails allocation of relevant objects to their position or positions in the grid, then creating an index of object identifiers vs. grid cell identifiers for rapid access. This is an example of a "space-driven" or data independent method, as opposed to "data-driven" or data dependent method. A grid-based spatial index has the advantage that the structure of the index can be created first, and data added on an ongoing basis without requiring any change to the index structure; indeed, if a common grid is used by disparate data collecting and indexing activities, such indices can easily be merged from a variety of sources. On the other hand, data driven structures such as R-trees can be more efficient for data storage and speed at search execution time, though they are generally tied to the internal structure of a given data storage system. Fig. 3 shows how to use grid indexing to partition the spatial data and build the indexes.

R-trees are tree data structures used for spatial access methods, i.e., for indexing multi-dimensional information such as geographical coordinates, rectangles or polygons. A common real-world usage for an R-tree might be to store spatial objects such as restaurant locations or the polygons that typical maps are made of: streets, buildings, outlines of lakes, coastlines, etc. The R-tree can also accelerate nearest neighbour search for various distance metrics, including great-circle distance. Fig. 4

The key idea of the data structure is to group nearby objects and represent them with their minimum bounding rectangle in the next higher level of the tree; the “R” in R-tree is for rectangle. Since all objects lie within this bounding rectangle, a query that does not intersect the bounding rectangle also cannot intersect any of the contained objects. At the leaf level, each rectangle describes a sin-

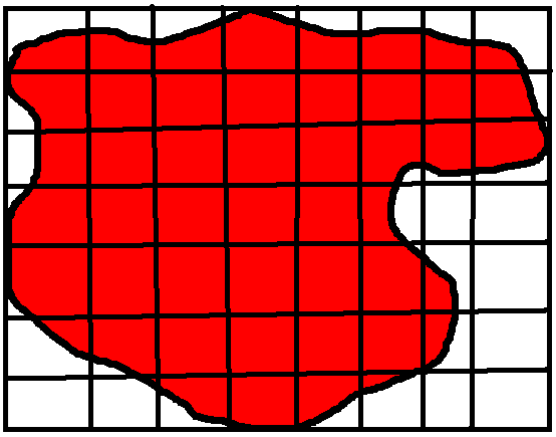

Figure 3: Grid indexing partitioning

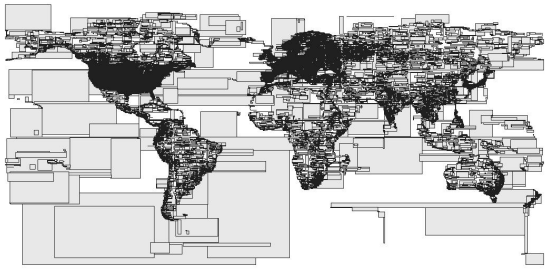

Figure 4: Rtree indexing partitioning

gle object; at higher levels the aggregation of an increasing number of objects. This can also be seen as an increasingly coarse approximation of the data set.

4. PROPOSED SOLUTION

4.1 Module 1

Main Idea FileSplitUtil : combine ....
Algorithm
Example

4.2 Module 2

Main Idea
Algorithm
Example

4.3 Cost And Correctness Analysis

Lemma 1: The time asymptotic time complexity of the *W* algorithm is $O(KlogN)$ where K is ..., and N is the....

Proof:
Lemma 2:

5. EXPERIMENTAL EVALUATION

This section provides an extensive experimental study for the performance of SpatialHadoop with MapReduce layer extension and compares it with the original SpatialHadoop framework results. For this paper, we mainly compare it with the original SpatialHadoop instead of other parallel spatial DBMSs or HadoopGIS for two reasons. First, SpatialHadoop [9] paper already presented the superior performance over other parallel spatial DBMSs. Second, the HadoopGIS is not a open source software, thus it is not appropriate to evaluate it for research purpose. SpatialHadoop was implemented inside Hadoop 2.7.2 on Java 1.8, our MapReduce extension version was developed based on the previous SpatialHadoop and compiled with the same steps.

All experiments are conducted on Amazon Elastic MapReduce Service, which is a abstract service combines Amazon EC2 service and Amazon S3 service. Amazon Elastic MapReduce (Amazon EMR) is a web service that makes it easy to quickly and cost-effectively process vast amounts of data. Amazon EMR simplifies big data processing, providing a managed Hadoop framework that makes it easy, fast, and cost-effective for you to distribute and process vast amounts of your data across dynamically scalable Amazon EC2 instances. Amazon EMR securely and reliably handles your big data use cases, including log analysis, web indexing, data warehousing, machine learning, financial analysis, scientific simulation, and bioinformatics.

In these experiments, we used mainly two categories of real datasets, TIGER and OpenStreetMap (OSM) to test various performance aspects. We focused on the time efficiency. TIGER datasets are real datasets which represents spatial features in the US, such as streets and rivers. Two specific datasets were picked for our evaluation experiments. The first one is Counties information of US, with size 149MB, which contains of 3,233 Polygons spatial data records. The second one is 5-Digit ZIP Code Tabulation Area (ZCTA5), with size 1GB, which contains of 33,144 Polygons. The steps for conducting time measurements are explained in later sections.

5.1 Competitive Approach

5.2 Experimental Setup

5.3 Experiment 1

5.4 Experiment 2

5.5 Experiment N

5.6 User Study

5.7 Experiments Summary

6. CONCLUSION

7. REFERENCES

[1] A. Aji, F. Wang, H. Vo, R. Lee, Q. Liu, X. Zhang, and J. Saltz. Hadoop gis: a high performance spatial data warehousing system over mapreduce. *Proceedings of the VLDB Endowment*, 6(11):1009–1020, 2013.

[2] A. Akdogan, U. Demiryurek, F. Banaei-Kashani, and C. Shahabi. Voronoi-based geospatial query processing with mapreduce. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 9–16. IEEE, 2010.

[3] L. Alarabi, A. Eldawy, R. Alghamdi, and M. F. Mokbel. Tareeg: a mapreduce-based system for extracting spatial data from openstreetmap. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 83–92. ACM, 2014.

[4] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[5] J. Dittrich and J.-A. Quiané-Ruiz. Efficient big data processing in hadoop mapreduce. *Proceedings of the VLDB Endowment*, 5(12):2014–2015, 2012.

[6] A. Eldawy, Y. Li, M. F. Mokbel, and R. Janardan. Cg_hadoop: computational geometry in mapreduce. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 294–303. ACM, 2013.

[7] A. Eldawy and M. F. Mokbel. A demonstration of spatialhadoop: an efficient mapreduce framework for spatial data. *Proceedings of the VLDB Endowment*, 6(12):1230–1233, 2013.

[8] A. Eldawy and M. F. Mokbel. Pigeon: A spatial mapreduce language. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 1242–1245. IEEE, 2014.

[9] A. Eldawy and M. F. Mokbel. Spatialhadoop: A mapreduce framework for spatial data. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 1352–1363. IEEE, 2015.

[10] M. R. Evans, D. Oliver, K. Yang, X. Zhou, and S. Shekhar. Enabling spatial big data via cybergis: Challenges and opportunities. *a book chapter in CyberGIS: Fostering a New Wave of Geospatial Innovation and Discovery (Ed. S. Wang and M. Goodchild)*, 2014.

[11] L. J. Klein, F. J. Marianno, C. M. Albrecht, M. Freitag, S. Lu, N. Hinds, X. Shao, S. Bermudez Rodriguez, and H. F. Hamann. Pairs: A scalable geo-spatial data analytics platform. In *Big Data (Big Data), 2015 IEEE International Conference on*, pages 1290–1298. IEEE, 2015.

[12] K. Lee, R. K. Ganti, M. Srivatsa, and L. Liu. Efficient spatial query processing for big data. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 469–472. ACM, 2014.

[13] X. Liu, J. Han, Y. Zhong, C. Han, and X. He. Implementing webgis on hadoop: A case study of improving small file i/o performance on hdfs. In *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pages 1–8. IEEE, 2009.

[14] Y. Lu, M. Zhang, S. Witherspoon, Y. Yesha, and N. Rishe. Sksopen: Efficient indexing, querying, and visualization of geo-spatial big data. In *Machine Learning and Applications (ICMLA), 2013 12th International Conference on*, volume 2, pages 495–500. IEEE, 2013.

[15] H. Tan, W. Luo, and L. M. Ni. Clost: a hadoop-based storage system for big spatio-temporal data analytics. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2139–2143. ACM, 2012.

[16] R. C. Taylor. An overview of the hadoop/mapreduce/hbase framework and its current applications in bioinformatics. *BMC bioinformatics*, 11(Suppl 12):S1, 2010.

[17] J. Yu, J. Wu, and M. Sarwat. Geospark: A cluster computing framework for processing large-scale spatial data. 2015.

[18] J. Zhang, S. You, and L. Gruenwald. Large-scale spatial data processing on gpus and gpu-accelerated clusters. *SIGSPATIAL Special*, 6(3):27–34, 2015.

[19] S. Zhang, J. Han, Z. Liu, K. Wang, and S. Feng. Spatial queries evaluation with mapreduce. In *Grid and Cooperative Computing, 2009. GCC'09. Eighth International Conference on*, pages 287–292. IEEE, 2009.

[20] K. Zheng and Y. Fu. Research on vector spatial data storage schema based on hadoop platform. *International Journal of Database Theory and Application*, 6(5):85–94, 2013.