HBaseSpatial:a Scalable Spatial Data Storage Based on Hbase

Ningyu Zhang
Department of Computer Science
Zhejiang University
Hangzhou, China
zxlzr@zju.edu.cn

Huajun Chen
Department of Computer Science
Zhejiang University
Hangzhou, China
huajunsir@zju.edu.cn

Xi Chen
Department of Computer Science
Zhejiang University
Hangzhou, China
xichen@zju.edu.cn

***Abstract*—Recent years, the scale of spatial data is developing more and more huge, and its storage has encountered a lot of problems. Traditional DBMS can efficiently handle some big spatial data. However, popular open source relational database systems are overwhelmed by the high insertion rates, querying requirements and terabytes of data that these systems can handle. On the other hand, key-value storage can efficiently support large-scale operations. To resolve the problems of big vector spatial data's storage and query, we bring forward HBaseSpatial:a scalable spatial dada storage based on HBase. At first, we analyze the distributed storage model of HBase. Then, we design a storage and index model. Finally, the advantages of our storage model and index algorithm are proven by experiments with both large sample sets and typical benchmarks on the cluster compared with MongoDB and Mysql, which shows that HBaseSpatial can efficiently enhance the query speed of big spatial data and provide a good solution for storage.**

***Keywords*-Big Spatial Data;Distributed;HBase**

I. Introduction

With the advancements of data-acquisition techniques, large amounts of spatial data have been collected from multiple data sources, such as satellite observations, remotely sensed imagery, aerial photography, and model simulations. The spatial data is growing exponentially to PB (Petabyte)scale even EB (Exabyte) scale [32].

As an important spatial data type in the field, vector data has been widely used in spatial information area. However, because it has a large amount of data, variable-length data records, etc., how to store and manage large expansion of vector data become a topic of research. Geographic data is the Earth's surface as a basic reference framework for spatial data and attribute information. In order to facilitate the acquisition, storage, analysis and management of geographic data, we establish a Geographic Information System, or GIS systems (geographic information system). GIS system represents each target of the real world, such as roads, land, elevation and more. Vector data and raster data are commonly used data structures in GIS. In vector data structure, an area is divided among a plurality of polygons, each polygon by a number of line segments or arc's components. Vector data structure stores a small amount of data and graphics with high precision, which is easy to define a single spatial object. However, when dealing with spatial relationships it compares time-consuming, so it is often used to describe the graphical data. Raster data structure, to the entities within the grid cell within the row and column logo, it is simple and easy to handle spatial relationship. However, its data storage capacity and graphics accuracy is low, which is commonly used to describe images and image data.

Currently, we mostly use the file system to manage raster data and databases to manage vector data. However, vector data not only has the structured data which is suitable for traditional relational database to manage, but includes some unstructured data, which is unsuitable. At present, present the major database vendors finally continue to optimize the management of vector data. Many manufacturers in their products support and extend on the realization of the vector data. Most solutions are to store the attribute data associated with vector data in a traditional relational database, and the really spatial data is stored in other databases or using middleware. However, most of their functional is not perfect, which has some drawbacks.

Moreover, the spatial data, particularly vector data, which not only has variable length in the real part from the vector data and non-uniform of all kinds, but also has the number and type of inconsistencies within the attribute data related to the vector data. For example, vector data for a typical shape file document, the distinct attributes for the shape file's data field type, the number of fields and the fields are different meaning. If we use the conventional relational database to store, we may need to create a lot of tables. These data are not structured data, which means the use of traditional relational database storage is not very appropriate.

SQL database provides a rich semantic support for complex queries, but for a relatively simple spatial data query, these semantic features are not used in spatial data processing, which cause an additional system resources waste in storage and query. For the current GIS systems, spatial data is the huge amount of data. With the increasing scale of vector data, the processing power of a singular node will progressively become a bottleneck. Single point of failure problem has imperceptibly become severe. The more traditional relational database running on a unique node, the higher the cost of supporting distributed database and

configuration costs will be. And most of the relational database cluster performance and scalability are not very good.

Thus, with the characteristics of the unstructured or semi-structured data volume of the vector data, its storage, query and management should be taken full account of these characteristics, the use of traditional relational database to store and on the vector data, management has been not very appropriate. HBase is a high-reliability, high performance, column-oriented, scalable, distributed storage system [6] [21] [3] [4]. HBase by adding low-cost commodity servers to increase the computing power and storage capacity, is primarily used for storing unstructured and semi-structured data lose. These features are just HBase suitable for large-scale vector data's storage, analysis and management [33] [20].

In a view of this, in this paper we propose a HBase based big vector spatial data-storage system. The framework can support very large scale distributed vector spatial data storage and management and vector data for the spatial extent of secondly indexes. The range can support fast queries based on geographic location with higher access con currency. Based on HBaseSpatial, we can also easily extend the data size. Our proposal has several significant characteristics.

1)We propose the design of distributed vector spatial data storage and index based on the HBase.

2)We demonstrate how this design can be used to improve the efficiency of storage and management of spatial data.

The remaining of the paper is organized as follows. In Sect. 2, we describe related work of distributed storage of big vector spatial data. . Section 3 gives the details of Vector spatial data storage and indexing. Section 4 shows our experiments and result evaluation. Section 5 is the conclusion.

II. Related Work

A. Conventional SQL Based Spatial Data Storage

Conventional SQL based spatial data-storage system provides a straightforward formal structure for storing and managing information in tables. Data storage and retrieval are implemented with simple tables. The multiuser geodatabase utilizes the power of the RDBMS. Certain characteristics of geographic data management, such as disk-based storage, definition of attribute types, query processing, and multiuser transaction processing, are delegated to the RDBMS.

MySQL spatial provides a basic implementation of OGCs SFS for SQL standard, but query and analysis operations utilize bounding rectangles instead of true geometries [22]. MySQL Spatial has some substantial disadvantages. On the plus side, it does have spatial types, functions and an index. And it follows the OGC specification for geometry representations. However, the number of functions MySQL supports is very small, and as a result it is difficult to use the database for anything more complex than simple storage and retrieval-by-bounding-box use cases. Additionally, because the spatial option is implemented in the (non-transactional) MyISAM table type, it is not possible to use spatial objects within transactions.

PostgreSQL was designed from the very start with type extension in mind C the ability to add new data types, functions and access methods at run-time [23] [25] [8]. Because of this, the PostGIS extension was developed by a separate development team, yet still integrate very tightly into the core PostgreSQL database. PostGIS began following the OGC SFSQL [15] document early in the development process, and it achieved full specification coverage.

Conventional SQL based spatial data-storage system do have some advantages, which significantly lower the development time of client applications. It offsets complicated tasks to the DB server. Organization and index have done for you, and you do not have to re-implement operators's functions.Moreover,you can use simple SQL expressions to determine spatial relationships and perform spatial operations.However,disadvantages of conventional SQL based spatial data-storage system are clear. Cost of implemention of the data base system can be high and when the scale of spatial data is developing more and more huge.Neverthless,conventional SQL based spatial data-storage system incompatibilities with some GIS software and are slower than local, specialized data structures [7].

B. NoSQL Baesd Spatial Data Storage

Spatial data can be stored by traditional SQL databases and also by NoSQL databases [10] [9] [34]. NoSQL databae systems are designed to scale to thousands or millions of users doing updates as well as reads in contrast to traditional DBMSs and data warehouses. NoSQL database systems are often highly optimized for retrieval and appending operations and often offer little functionality beyond record storage (e.g. keyCvalue stores). In short, NoSQL database management systems are useful when working with a huge quantity of data (especially big data) when the data's nature does not require a relational model [27].

MongoDB is a scalable, high-performance, open source, schema-free, document-oriented database. MongoDB has direct support for spatial data and index along with features like finding nearby locations creating a geospatial index on legacy coordinate pairs [10].MongoDB computes geohash values for the coordinate pairs within the specified range and indexes the geohash values. Geohash is a latitude/longitude geocode system. In spite of spite of MongoDB, other NoSQL databaese do not natively support spatial index, but this can be extended via geohashing [5] [1].

MD-HBase is a scalable data-management system for LBSs that bridge this gap between scale and functionality, which approaches leverages a multi-dimensional index structure layered over a Key-value store [24]. The underlying

Key-value store allows the system to sustain high insert throughput and large data volumes, while ensuring fault-tolerance, and high availability. On the other hand, the index layer allows efficient multi-dimensional query processing.

SpatialHadoop is an open-source MapReduce framework designed specifically to handle huge datasets of spatial data, which is shipped with built-in spatial high level language, spatial data types, spatial indexes and efficient spatial operations. SpatialHadoop can interact within the system easily with a simple high level language and load all datasets in SpatialHadoop with the built-in spatial data types.Moreover,Spatialhadoop can also store your data efficiently in a spatial index of your choice and analyze your data on large clusters with built-in spatial operations.

CloST is a scalable big spatio-temporal data-storage system which can support data analytics using Hadoop [28].The main objective of CloST is to avoid scanning the whole dataset when a spatio-temporal range is given. CloST uses a novel data model which has special treatments on three core attributes, including an object id, a location and a time. Based on this data model, its hierarchically partition's data using all core attributes which enable efficient parallel processing of spatio-temporal range scans. According to the data characteristics, CloST devises a compact storage structure which reduces the storage size by an order of magnitude.

In short, because SQL databases are not that good at some tasks like "indexing a large number of documents, serving pages on high-traffic websites, and delivering streaming media" [2].NoSQL solutions are particularly good at dealing with lots of reading/writing tasks coming in at once, something that tends to slow down SQL/relational databases. Some of NoSQL databases currently being used to manage geospatial data include: MongoDB (open source), BigTable (developed by Google, proprietary, used in Google Earth), Cassandra (developed by Facebook, now open source and maintained by Apache), CouchDB (open source, Apache) and so on. With the growth of scalability of spatial data, NoSQL database may be a good choice.

III. Big Spatial Data Storage And Indexing

A. Architecture

Figure 1 illustrates the architecture of our distributed vector spatial storage system. The systemis divided into two parts:storage model and index model. The storage model receives the vector data from original shapefiles [13]. Then it does two jobs:directly puts these data into the index model;converts these data to WKB type and stores in the HBase table. When the vector data comes from the index model, according to to our index algorithm, we will calculate the id of each vector data and puts them into the index table. To search for a range of vector data, the index model will calculate the IDs in the index table which is just in the input range and find these IDs from the data table. Through this secondary index method, we can efficiently make a range search of vector data.

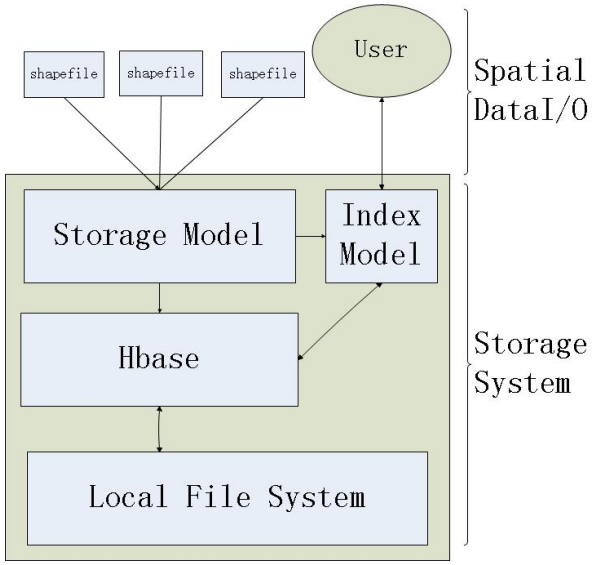

Figure 1. Architecture of Vector Spatial Data Storage

B. Physical Storage Of Big Spatial Data

Non-spatial attribute of spatial data objects can be accessed by common data types such as string or number. But spatial attribute of spatial data cannot be accessed easily because it contains a large number of geometric coordinate information.

We try to use a method which compresses spatial attribute into a binary large object to store [16] [31].We adopt the persistent coding rules with the object which have been regulated by the OGC and described in their Access and Coordinate. Transformation Service Specifications. It is Well Known Text (WKT) and. Well Known Binary (WKB). WKT is a text markup language for representing vector geometry objects on a map, spatial reference systems of geographic objects and transformations between spatial reference systems. WKB is used to transfer and store the same information in binary bytes. For example, if we have a point object with the x coordinate and y coordinate value is 1, then the equivalent WKT fotmat is Point(1,1).

C. HBase Storage Model

Apache HBase is the Hadoop database, a distributed, scalable, big data store [14].HBase store data in form of a table. Each table consists of rows and columns. Each column belongs to a particular column family (Column Family) The storage unit in rows and columns as an element, each of the elements saves multiple versions of the same data by timestamp marked. Since HBase store data sparsely, so some columns can be empty, which make it possible to store different kind of vector spatial data [19] [30].

A table in HBase consists of a collection of splits, called regions, where each region stores a range partition of the

key space. Its architecture comprises a three layered B+ tree structure; the regions storing data constitute the lowest level. The two upper levels are special regions referred to as the ROOT and META regions. A HBase installation consists of a collection of servers, called region servers, responsible for serving a subset of regions. Regions are dynamically assigned to the region servers; the META table maintains a mapping of the region to region servers. If a regions size exceeds a configurable limit, HBase splits it into two sub-regions. This allows the system to grow dynamically as data is inserted, while dealing with data skew by creating finer grained partitions for hot regions.

D. Vector Data Storage Model

HBase has rich data types and is different from RDBMSstorage mode, which has only a simple string type and all the types must be dealt with [29]. In this paper. We use OGC recommended for space as a model for processing. Spatial data follows OGC WKB /WKT standard. WKT through text description geometry object and space reference and WKB through serialized bytes object of description geometry have a higher literacy and storage efficiency. So we use format of spatial data to storage. Mainly, the WKB contains two kinds of numeric types, including uint32 nodes, which can be used to store number, geometric information such as object type. Double node can be used to store the sit values and other information.

Vector spatial data generally includes attribute data, spatial coordinates and the topology data [31] [14]. According to the characteristics of the vector spatial data design based on HBase storage table structure as shown in figure 3, column in turn in the column family, geodata is spatial coordinates and some other attribute data, each type of data type is string type and will be parsed into the corresponding data type. For example, when we store a point information in the HBase, storage model will find out whether there exists a type of point in column. If so, all point data will be put into the corresponding columns. On the other hand, storage the other hand,storage model will add a new column to store the data. ”Row Key” is a vector layer, which is the only ID elements. Attribute data can have multiple rows. Each row represents the vector. The coordinate data uses WKB format to store in HBase. The whole table is sparse because different types of spatial data have the different kind of attributes. The total algorithm of storing spatial data in HBase is shown below.

Algorithm 1 Algorithm of Insert Spatial Data in HBase

Require: Input:Several shapefiles ;Output:insertion state

Step 1) Retrieve the vector data from shapefiles and put them into a special array.

Step 2) Convert the spatial information into WKB format.

Step 3) Record the type of vector data and store them into different column family in HBase.

Step 4) Output the HBase return information.

RowKey	Timestamp	Column Family: geodata			
		Col1: WKB	Col2: Attr1	Col3: Attr2	Col^n: Attrn
$fida_1$	t_n	001100011		$attr_2a_1$	
$fida_2$	t_{n-1}	001100010		$attr_2a_2$	
$fida_3$	...	011100011		$attr_2a_3$	
...	...	...		...	...
$fida_{4021}$	...	101111011		$attr_2a_{4021}$	
$fidb_1$	...	110010010	$attr_1b_1$		$attr_nb_1$
$fidb_2$	...	110010011	$attr_1b_2$		$attr_nb_2$
$fidb_3$	...	110010110	$attr_1b_3$		$attr_nb_3$
...	...	...		...	
$fidb_{378}$	...	111110010	$attr_1b_{378}$		$attr_nb_{378}$
...	t_1	...	...	...	...

Figure 2. Table structure of vector data

E. Vector Data Index

In order to improve the efficiency of spatial query, we need to construct a spatial index. At present, there are many spatial index algorithms [17] [18]. This paper mainly adopts the grid spatial index method.

1)Algorithm of Index

We divide the global scope of longitude and latitude into different levels of the grid, as shown in figure 4 and 5 are two different levels of the grid partition method. They respectively show the mesing length of 1 degree and meshing length of 0.1 degree. Each of the partition granularities corresponds to a kind of level. So we can get each grid number by dividing. The rules of grid numbers are as follows: the grid center coordinates + grid level. For example, 11.50_23.00_2 can uniquely identify a grid, the grid center coordinates (121.50, 121.50), the level of grid of 2. Coordinate values in the serial number use the same precision and digits.

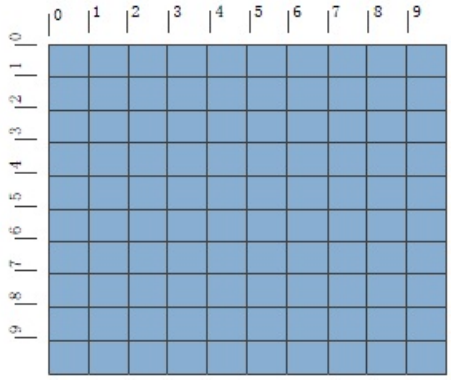

Figure 3. Mesing Length of 1 Degree

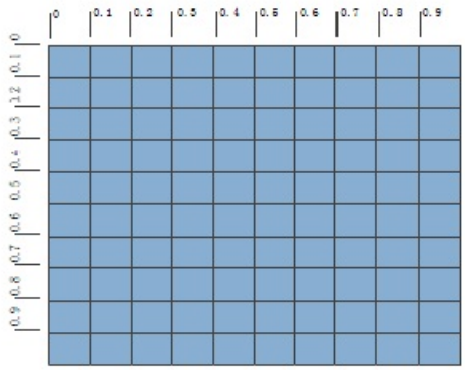

Figure 4. Mesing Length of 0.1 Degree

For vector data records, we should find out the minimum grid which can fully include the vector object, which means that we should find out the most accurate description for the vector object to coordinate grid. The vector data records described by vector object scope of label is the serial number of the grid. Vector objects such as POINT (47.6245, 47.6245), the most accurate description of vector objects coordinates range of grid side length is 0.1 degrees(if the side length of 0.1 degrees is the smallest partition granularity) respectively the diagonal coordinates is (47.60, 47.60) and (47.70, 47.70) of the grid, then we mark the vector object POINT (47.6245, 47.6245) in the range of numbers 47.55_123.05_4. For other vector objects, we can use the same method to calculate space range number.So, we can build an index table to fully correspond to the spatial data model.

The total algorithm of calculating accurate description for the vector object to coordinate grid is shown below.

Algorithm 2 Algorithm of Caculate the Grid Level of Spatial Object

Require: Input: One kind of vector spatial data s with the type of WKT;Output:The id of vector spatial data (point,level)

Step 1) Process the data s and find out if it is a Point.If it is a Point make it level of 4 and output (point,level)

Step 2)Traverse all coordinates to identify the smallest longitude dot note it x_min,as the same way find out the smallest latitude y_min,biggest longitude x_max and latitude y_max.

Step 3)Find a point which is to 0.5 or 0.0 at the end of decimal and is nearest to the point of $(\frac{x\_min+x\_max}{2}, \frac{y\_min+y\_max}{2})$.

Step 4)Caculate the maximum of y_max-y_min,x_max-x_min to identify the level of the grid.

Step 5)Output (point,level).

2)Index Table Architecture

We build another table in HBase database as shown in figure 6, which has a column cluster and develops the index number within the vector object range. We use the vector data record ID as HBase database rows of keys (RowKey).The column has a value of 0 to a plurality of vector data records of the id, individual id separated by semicolons. The table level identifier for each vector data lines keys (RowKey). So you can quickly check out the latitude and longitude coordinate in a certain range all the vector data record id. Then we can get the vector data recorded by ID, according to the ID in the First Data table quickly after the vector data record. In the table fida1, fida2 are all vector data record id. Other property fields as columns for the vector number according to the records stored in the column below, the field name is the column name. Because HBase uses B + tree sort storage method on the line key (RowKey) [14],which is so convenient that you can query all vector objects that belong to a grid of the vector, and it can greatly accelerate the speed of geographical location range query.

RowKey	Timestamp	Column Family: fid
		Col1: fids
11.50_47.00_0	t_n	$fida_1$; $fida_{17}$; $fida_{309}$; $fidc_{78}$; $fidc_{581}$; $fidk_{99}$; $fidz_1$
11.50_47.00_1	t_{n-1}	$fidm_{165}$; $fidn_{187}$; $fidq_{409}$; $fidq_{718}$
11.50_47.00_2	...	$fidr_{559}$
11.50_47.00_3	...	$fidx_{334}$; $fidx_{798}$; $fidx_{6112}$; $fidz_3$
11.50_47.50_0	...	$fida_{73}$; $fida_{766}$; $fida_{901}$; $fidc_{21}$; $fidc_{151}$; $fidk_4$; $fidz_9$
11.50_47.50_1	...	$fidt_{212}$
11.50_47.50_2	...	
11.50_47.50_3	...	$fidb_{73}$; $fidd_{766}$; $fidf_{901}$; $fidk_{21}$; $fidr_{151}$; $fids_4$; $fids_9$
...	t_1	...

Figure 5. Table Structure of Vector Index

3)Range Query Method

In order to make a range query of spatial data, we need to conform the input data from an array of grid IDs so the HBase can identify. In fact, searching fact,searching spatial data in a range of rectangles is quite easy. The total algorithm is shown below.

Algorithm 3 Algorithm of Query a Range of Vector Spatial Data

Require: Input: Rectangular angular coordinate value x_1,y_1,x_2,y_2; Output: Array of vector spatial data
Step 1) Calculate the rectangular's side and find out whether it is in a meshing grid.If so,caculate the grid id and go to step 3
Step 2) Calculating: let $s = x_1$,$t = y_1$
while $s \leq x_2$ **do**
a)Traverse all longitude with the length of 0.5
while $t \leq y_2$ **do**
a1)Traverse all latitude with the length of 0.5
a2)Put the point into a array
end while
Step 3)Find the grid id in the index table and get the corresponding spatial data from data table.
Step 4)Output the spatial data.
end while

IV. Experiment Evolution

A. Test Environment And Data

We use several computers as hosts, each installs VirtualBox virtual machine and have the environment of HDFS and HBase. The configuration is shown below. Host: CPU Duo T7700 Memory 4GB operation System Windows 7; virtual machine: the operating systems are UbuntuLinux10.1, 1GB of memory, Hadoop-0.20.203, HBase -0.90.3. The development environment Eclipse3.6. The test data is from jackpine [26],which is 1:50000 vector data (about 1. 00GB).

B. Build Vector Spatial Data Index

In this article, we use GeoTools-2.7.4 to store the vector data [12],which is a open-source project to read the client's local shapefile data, import the data into the table. Total time includes the read and writes time of index procedure.

C. Vector Spatial Data Query Test

The steps of querying vector data based on the HBaseSpatial are as follows:

(1) Calculate the IDs of the spatial object according to the computational grid;

(2) From the different layers of the index table, read the contained grid ID;

(3) Read different layers of spatial data and attribute data based on the grid ID in the data table;

(4) Write to the local client. Query time includes the computation if grid ID, reading the HBase table, the amount of reading and writing spatial data to the local file system.

D. Performance Of Horizontal HBase Cluster Expansion

As the table shown below which uses the unit of second, with the increase of a number of HBase nodes, the time of creating an index of vector spatial data storage is lowing. How ever, after six nodes the efficiency is not that clear. Because HBase divide Region according to the different amount of data (default 256M) [14].A Region is managed by a Region Server. When the amount of data expending, the Region portion uses different numbers.Only under the condition of special nodes that can accelerate the most efficient of making index.

Table I
Performance Of Horizontal HBase Cluster Expansion

Inserting Test(M)	Node 1(s)	Node 5(s)	Node 6(s)	Node 7(s)
6M	3.112	2.932	2.891	2.883
12M	20.251	19.231	18.911	18.882
50M	131.223	109.231	93.224	92.121
100M	214.324	208.421	181.212	180.993
500M	1231.231	1107.122	883.231	851.823
1G	2632.234	2214.324	1753.234	1702.214

E. Comparison Of MySQL and MongoDB

We use the same computers above to build the MySQL Cluster. MySQL Cluster is a write-scalable, real-time, ACID-compliant transactional database. We use shp2mySQL tool to make shape file become SQL. Then we use these SQL to store them into the MySQL Cluster. We calculate the whole time of inserting data and make server query test just the same as above.

MongoDB is established in the same environment as above.We use a python script to deal with shape file and store the data to the MongoDB. From then MongoDB can store different kind of spatial data but can only make the index of point data.

The figure below shows the three different kind of spatial data small range query test, which is point data, LingString data and MultiLingString data [11].The horizontal axis means the data size and uses the unit of megabytes.The vertical axis means the time of searching based on the different kind of storage platform and use the unit of second. Because MongoDB only supports the point data type of the index, so when searching the point, MongoDB performs mostly the same as HBase. With the scale growth of data, MySQL performs not that fifteenth,when searching the other kind of spatial data such as LineString data and MultiLingString data HBase performs better.

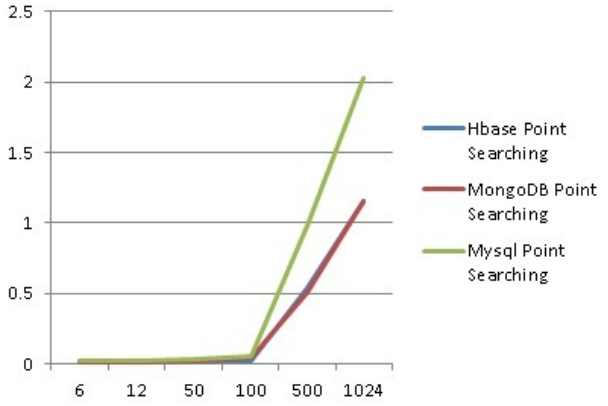

Figure 6. Point Data Query Efficiency

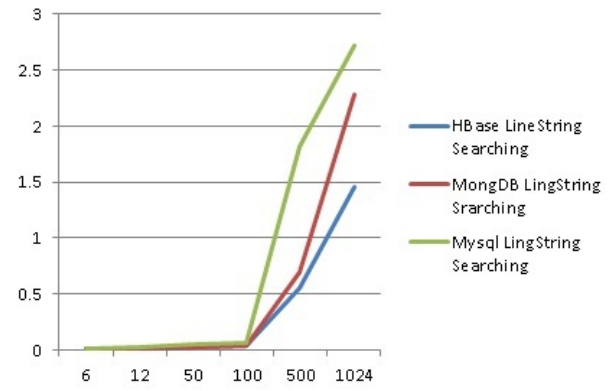

Figure 7. LingString Data Query Efficiency

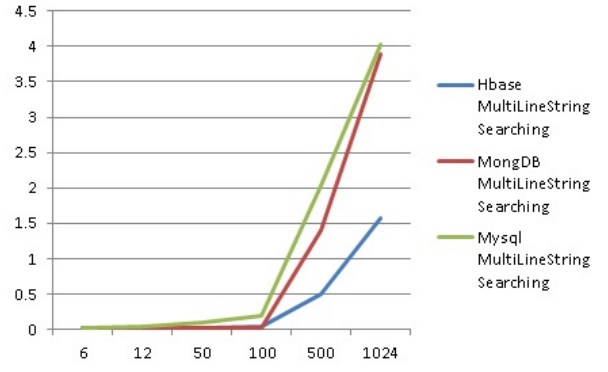

Figure 8. MultiLingString Data Query Efficiency

V. Conclusion

In this paper, we bring forward HBaseSpatial:a scalable spatial dada storage based on HBase. We design and establish the index table and index storage format. Finally, the effectiveness and practicability of this framework are verified comparing with MongDB and MySQL. When we search complex spatial data, especially MultiLingString and LingString types, our framework performs better than MongnDB and MySQL. This framework provides effective and efficient distributed storage and parallel processing paradigm for big spatial data.

References

[1] Kashif Ali, Dina Al-Yaseen, Ali Ejaz, Tayyab Javed, and Hossam S Hassanein. Crowdits: Crowdsourcing in intelligent transportation systems. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pages 3307–3311. IEEE, 2012.

[2] Amin Anjomshoaa and A Min Tjoa. How the cloud computing paradigm could shape the future of enterprise information processing. In *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*, pages 7–10. ACM, 2011.

[3] Wan D Bae, Sada Narayanappa, Shayma Alkobaisi, and Kye Y Bae. Mobis: a distributed paradigm of mobile sensor data analytics for evaluating environmental exposures. In *Proceedings of the First ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*, pages 93–96. ACM, 2012.

[4] Jiun-Wen Bai, Jun-Zhe Wang, and Jiun-Long Huang. Spatial query processing on distributed databases. In *Advances in Intelligent Systems and Applications-Volume 1*, pages 251–260. Springer, 2013.

[5] Zoran Balkić, Damir Šoštarić, and Goran Horvat. Geohash and uuid identifier for multi-agent systems. In *Agent and Multi-Agent Systems. Technologies and Applications*, pages 290–298. Springer, 2012.

[6] Yuan Bao, Lei Ren, Lin Zhang, Xuesong Zhang, and Yongliang Luo. Massive sensor data management framework in cloud manufacturing based on hadoop. In *Industrial Informatics (INDIN), 2012 10th IEEE International Conference on*, pages 397–401. IEEE, 2012.

[7] Daniel Bartholomew. Sql vs. nosql. *Linux Journal*, 2010(195):4, 2010.

[8] David Blasby. Building a spatial database in postgresql. *Open Source Database Summit*, 2001.

[9] Rick Cattell. Scalable sql and nosql data stores. *ACM SIGMOD Record*, 39(4):12–27, 2011.

[10] Kristina Chodorow and Michael Dirolf. *MongoDB: the definitive guide*. O'Reilly Media, 2010.

[11] Simon Cox, Adrian Cuthbert, Paul Daisey, John Davidson, Sandra Johnson, Edric Keighan, Ron Lake, Marwa Mabrouk, Serge Margoulies, Richard Martell, et al. Opengis® geography markup language (gml) implementation specification, version. 2002.

[12] John M Coyne and Ivan Henson. Geotool sourcebook: User's manual. Technical report, DTIC Document, 1995.

[13] ESRI ESRI. Shapefile technical description. esri. *INC, http://www. esri. com*, 1998.

[14] Lars George. *HBase: the definitive guide*. O'Reilly Media, Incorporated, 2011.

[15] Michael Gould and Louis Hecht Jr. Ogc: A framework for geospatial and statistical information integration. *Joint UNECE/Eurostat Work Session on Methodological Issues. Tallinn, Estonia*, 2001.

[16] Ralf Hartmut Güting. An introduction to spatial database systems. *The VLDB JournalłThe International Journal on Very Large Data Bases*, 3(4):357–399, 1994.

[17] Antonin Guttman. *R-trees: a dynamic index structure for spatial searching*, volume 14. ACM, 1984.

[18] Dan Han and Eleni Stroulia. A three-dimensional data model in hbase for large time-series dataset analysis. In *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2012 IEEE 6th International Workshop on the*, pages 47–56. IEEE, 2012.

[19] Vamshi Krishna Konishetty, K Arun Kumar, Kaladhar Voruganti, and GV Rao. Implementation and evaluation of scalable data structure over hbase. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, pages 1010–1018. ACM, 2012.

[20] Yan Li, GyoungBae Kim, LongRi Wen, and HaeYoung Bae. Mhb-tree: A distributed spatial index method for document based nosql database system. In *Ubiquitous Information Technologies and Applications*, pages 489–497. Springer, 2013.

[21] Youzhong Ma, Jia Rao, Weisong Hu, Xiaofeng Meng, Xu Han, Yu Zhang, Yunpeng Chai, and Chunqiu Liu. An efficient index for massive iot data in cloud environment. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2129–2133. ACM, 2012.

[22] Zhang Mingbo, Shen Paiwei, Lu Feng, and Cheng Changxiu. Analysis and discussion on spatial data engine technologies. *Geo-information Science*, 6:80–84, 2004.

[23] Bruce Momjian. *PostgreSQL: introduction and concepts*, volume 192. Addison-Wesley, 2001.

[24] Shoji Nishimura, Sudipto Das, Divyakant Agrawal, and Amr El Abbadi. Md-hbase: a scalable multi-dimensional data infrastructure for location aware services. In *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, volume 1, pages 7–16. IEEE, 2011.

[25] Regina Obe and Leo Hsu. *PostGIS in action*. Manning Publications Co., 2011.

[26] Suprio Ray, Bogdan Simion, and Angela Demke Brown. Jackpine: A benchmark to evaluate spatial database performance. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 1139–1150. IEEE, 2011.

[27] Christof Strauch, Ultra-Large Scale Sites, and Walter Kriha. Nosql databases. *URL: http://www. christof-strauch. de/nosqldbs. pdf (07.11. 2012)*, 2011.

[28] Haoyu Tan, Wuman Luo, and Lionel 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.

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

[30] Mehul Nalin Vora. Hadoop-hbase for large-scale data. In *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, volume 1, pages 601–605. IEEE, 2011.

[31] Yonggang Wang and Sheng Wang. Research and implementation on spatial data storage and operation based on hadoop platform. In *Geoscience and Remote Sensing (IITA-GRS), 2010 Second IITA International Conference on*, volume 2, pages 275–278. IEEE, 2010.

[32] Yunqin Zhong, Jizhong Han, Tieying Zhang, and Jinyun Fang. A distributed geospatial data storage and processing framework for large-scale webgis. In *Geoinformatics (GEOINFORMATICS), 2012 20th International Conference on*, pages 1–7. IEEE, 2012.

[33] Yunqin Zhong, Jizhong Han, Tieying Zhang, Zhenhua Li, Jinyun Fang, and Guihai Chen. Towards parallel spatial query processing for big spatial data. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 2085–2094. IEEE, 2012.

[34] Yunqin Zhong, Xiaomin Zhu, and Jinyun Fang. Elastic and effective spatio-temporal query processing scheme on hadoop. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, pages 33–42. ACM, 2012.