J. Cent. South Univ. (2018) 25: 2423−2437
DOI: https://doi.org/10.1007/s11771-018-3926-1

Springer

Design and achievement of cloud geodatabase for a sponge city

HOU Jing-wei(侯景伟)[1, 2], SUN Shi-qin(孙诗琴)[1, 2], LIU Ren-tao(刘仁涛)[3],
LI Jian-hua(李建华)[1, 2], ZHANG Ming-xin(张明鑫)[1, 2]

1. School of Resources and Environment, Ningxia University, Yinchuan 750021, China;
2. Ningxia (China-Arab) Key Laboratory of Resource Assessment and Environment Regulation in Arid Region, Yinchuan 750021, China;
3. Key Laboratory for Restoration and Reconstruction of Degraded Ecosystem in Northwestern China of Ministry of Education, Ningxia University, Yinchuan 750021, China

Abstract: Building a cloud geodatabase for a sponge city is crucial to integrate the geospatial information dispersed in various departments for multi-user high concurrent access and retrieval, high scalability and availability, efficient storage and management. In this study, Hadoop distributed computing framework, including Hadoop distributed file system and MapReduce (mapper and reducer), is firstly designed with a parallel computing framework to process massive spatial data. Then, access control with a series of standard application programming interfaces for different functions is designed, including spatial data storage layer, cloud geodatabase access layer, spatial data access layer and spatial data analysis layer. Subsequently, a retrieval model is designed, including direct addressing via file name, three-level concurrent retrieval and block data retrieval strategies. Main functions are realised, including real-time concurrent access, high-performance computing, communication, massive data storage, efficient retrieval and scheduling decisions on the multi-scale, multi-source and massive spatial data. Finally, the performance of Hadoop cloud geodatabases is validated and compared with that of the Oracle database. The cloud geodatabase for the sponge city can avoid redundant configuration of personnel, hardware and software, support the data transfer, model debugging and application development, and provide accurate, real-time, virtual, intelligent, reliable, elastically scalable, dynamic and on-demand cloud services of the basic and thematic geographic information for the construction and management of the sponge city.

Key words: cloud geodatabase; sponge city; Hadoop; concurrent retrieval; access

Cite this article as: HOU Jing-wei, SUN Shi-qin, LIU Ren-tao, LI Jian-hua, ZHANG Ming-xin. Design and achievement of cloud geodatabase for a sponge city [J]. Journal of Central South University, 2018, 25(10): 2423–2437. DOI: https://doi.org/10.1007/s11771-018-3926-1.

1 Introduction

Presently, the rapid urbanisation has changed the urban land use pattern and hydrological mechanism, increased the impervious area of the city. These changes deteriorate the microclimate environment, shorten the flood peak time, increase surface runoff, peak discharge, runoff pollution and waterlogging risk, decrease the amount of groundwater recharge and surface interception. China has vigorously constructed sponge cities to solve the abovementioned ‘urban diseases’. A sponge city refers to a city like a sponge that has certain flexibility in response to natural disasters and environmental changes. Sponge city is a construction model of low-impact development (LID) for natural infiltration, natural accumulation and natural purification. Rainwater in sponge cities is infiltrated, stored, absorbed and purified in the rainy season, and then the stored rainwater is released and utilised in the dry season.

Foundation item: Project(NZ1628) supported by the Natural Science Foundation of Ningxia, China
Received date: 2017−07−04; **Accepted date:** 2017−12−02
Corresponding author: HOU Jing-wei, PhD, Associate Professor; Tel: +86−15121888067; E-mail: hjwei2005@163.com; ORCID: 0000-0002-5308-3204

The spatial layout of spongy bodies, such as bio-swale, green roof, rainwater garden, is the most important factor that determines the efficiency of sponge city. The optimal spatial layout of sponge bodies can improve the overall performance, capital utilization and operation efficiency of sponge city. However, massive spatial data and attribute data are involved during the optimisation of spatial layout of the sponge bodies [1, 2]. Vector data include administrative boundaries, topography, roads and buildings. Raster data include digital elevation model, remote sensing images and land use. Drainage data include municipal pipe network, drainage node and water inlet node. Land use data include soil, vegetation, rivers and lakes, watershed and waterlogging area. Hydrologic data include permeability rate, groundwater level, impervious area, Manning coefficient, storage volume, erosion coefficient and erosion index. Pollution data include maximum pollutant accumulation and distribution of major pollutants. Meteorological data include temperature and precipitation. Attribute data include hydrological and hydraulic parameters, LID facility parameters, water supply, water demand and construction costs.

All of these data will be stored from the terabyte (TB) level to petabyte (PB) level. Meanwhile, these data will be distributed among various departments, such as the construction headquarter for sponge city, construction bureau, planning bureau, water conservancy bureau, meteorological bureau, institute of remote sensing surveying and mapping and land resource bureau. The data are massive, multi-scale, multi-type, multi-source and spatiotemporal so that they are difficult to concurrently access, process and compute [3–5]. Building a cloud geodatabase for a sponge city can solve these problems. Cloud geodatabase can integrate the geospatial information dispersed in various departments for massive data processing and multi-user high concurrent access.

Many cloud geodatabases have been developed, such as Database as a Service [6], geoprocessing-tools-for-hadoop [7], MongoDB [8], Amazon Relational Database Service [9], Building information models (BIMs) [10], GeoCouch [11], SpatialHadoop [12], GRIDDBLite [13], TerraFly GeoCloud [14], RDB-KV CloudDB [15], Geopot [16], Cloud-based geographic information system (GIS) [17], Hadoop-GIS [18, 19] and GeoMPP [20]. These cloud database products are to ensure the high efficiency and high availability of data. However, their architectures are not the same, and their functions, compared to that of traditional databases, have not been completely realized. These cloud geodatabases provide methods and techniques to achieve our cloud geodatabase for sponge city.

Cloud geodatabases have developed with the increase in demands of cloud computing technology, such as massive data storage space, fast data retrieval and processing and high data security. Cloud geodatabases have been applied to various fields as shown in Table 1.

According to the application fields and software types of cloud geodatabases abovementioned, cloud geodatabase has not been applied to the sponge city for spatial data storage, MapReduce, retrieval and access to optimize the spatial layout of the sponge bodies. Therefore, the purpose of this paper is to design and achieve a cloud geodatabase to provide accurate, real-time, virtual, intelligent, reliable, elastically scalable, dynamic and on-demand cloud services of the basic

Table 1 Application of cloud geodatabases

Application of cloud geodatabase	Usage of techniques	Reference
Evaluation of the dynamic groundwater quality using a spatial database, a property database and mathematical models	GIS and cloud-based simulation technologies	[21]
An efficient retrieval method for skewed spatial data	An R^+-tree	[22]
Online spatial analytical processing that is three-layer architecture: data layer, service layer and application layer	Cloud spatial data warehouses	[23]
High-performance cloud computing system for remote sensing image analysis, storage, data processing and on-demand services	Cloud Hadoop MapReduce	[24–27]
A middleware and a vector spatial data storage schema to express spatial topological relations and store large-scale vector spatial data	Key/value mapping, GeoTools toolkit and column-oriented storage structures	[28]
A cloud-based service for analysing, storing and viewing massive building information models	MapReduce and Bigtable as the data processing and storage paradigms	[29]
Data indexing of large-scale spatial and non-spatial datasets	Combination of MapReduce programming model with cloud computing	[30]

and thematic geographic information for the optimal spatial layout of the sponge bodies.

In this study, Hadoop distributed computing framework including a master and multiple slaves is first designed using virtualisation technology, geohash fragmentation, parallel computing and cross-platform service-oriented architecture of WebService. Cloud geodatabase access for the sponge city is then achieved including spatial data storage layer, cloud geodatabase access layer, spatial data access layer and spatial data analysis layer. Retrieval model is also constructed using direct addressing via a file name, three-level concurrent retrieval and block data retrieval techniques. Finally, the cloud geodatabase for sponge city is implemented and validated. It is shown that the cloud geodatabase for sponge city can support data transfer, model debugging and application development. It can also store and retrieve massive data with high-performance computing and communication.

2 Distributed computing framework of Hadoop

Cloud geodatabase for the sponge city is built using Hadoop distributed computing framework. MongoDB is used to keep the details of geodata set in the hadoop distributed file system (HDFS). The servers of the construction headquarter and other departments with relation to sponge city are abstracted as physical resources, such as central processing unit (CPU), memory, disk. These physical resources can be managed dynamically into a logical resource pool by using virtualisation technology. Cloud geodatabase for the sponge city adopts a master and multiple slaves' architecture. Servers of construction headquarter for the sponge city are the master centre. They mainly take charge of receiving and parsing the production order, results feedback, scheduling of task and allocation of data and service portals. The portals can provide cloud storage service, production service and data service. The master centre also accepts cloud register, preserves cloud information list, collects query results and consumes thumbnails and metadata from each slave centres. Slave clouds are the servers of the construction bureau, planning bureau, water conservancy bureau, meteorological bureau, land resource bureau and institute of remote sensing surveying and mapping. They are mainly responsible for task execution and massive data storage.

The parallel import function of spatial data is realised by using the geohash fragmentation method to improve the spatial query and concurrent access performances. Geohash is a geocoding method that transforms the two-dimensional coordinates of longitude and latitude into a string in accordance with the accuracy requirement.

A communication framework is built for data communication and sharing by using the cross-platform service-oriented architecture of WebService between the master and slave centres. It provides the external service interfaces, such as query, download and upload, for the cloud geodatabase. The framework can directly achieve interoperability between data and information, and also integrate and share data among different operating system users.

Hadoop is an open source programming of the Google FileSystem. Hadoop can process huge datasets across clusters of computers and is widely applied in many fields, such as graph processing, machine learning and behavioral simulations. It can scale up from single servers to a great many machines. Hadoop provides a cloud platform software with a parallel computing framework, including HDFS and MapReduce, to process massive spatial data [25]. Hadoop utilizes the high operation speed and mass data storage of the cluster. Meanwhile, it hides the underlying distributed details of parallel computing, data distribution and failure processing to users. HDFS locates the underlying part of Hadoop framework and accesses the file system data in binary stream form with high throughput and high fault tolerance. HDFS is deployed on a low-cost hardware. It can divide the data in the HDFS file into tile data and then store the tile data on each DataNode of Hadoop. MapReduce is a parallel computing framework that generates and processes large datasets for batch workloads of data-intensive applications implemented by HDFS. It can divide the computational work into small segments and distribute the segments among hundreds of different machines to ensure data reliability.

MapReduce computing consists of the mapping phase (Mapper) and the Reduce specification stage (Reducer). In the mapping phase, a set of key-value pairs are inputted by utilising the Map function to generate a set of intermediate results <*Key*, *Value*>. Each DataNode of MapReduce has a Mapper under the default configuration. Each Mapper reads the split file data from the DataNode as the key-value pair of the Map function, where *key* is a row number by default, and *value* is a string. The *value* contains the hierarchical structure and computing metrics for all dimensions. The Map function splits the fields based on dimension and hierarchical structure descriptions. A new string forms a *key* of the output key-value pair, whereas the string calculating metric is the *value* of the output key-value pair. At the same time, the MapReduce computing model sorts and groups the intermediate key-value pair output from the mapping phase according to the *keys*. The model then generates the intermediate data <*key*, *list* (*value*)> as the input of the Reduce stage.

In the Reduce specification stage, the Reduce function is used to aggregate and simplify the intermediate results from Mapper. All of the intermediate data <*key*, *list* (*value*)> are inputted into the Reduce function and then iterated until a set of *values* and fields corresponding to each *key* are accumulated. A new string is reconstructed as a *value* of key-value pair which is outputted from the Reducer.

The key-value pair of Hadoop MapReduce model can convert relational data in a relational database into structural data [31] to enhance the interoperability of Hadoop cloud geodatabase with Oracle relational database. Firstly, all dimension and metric relational datasets are calculated by using Cartesian product according to association key. Each dimension table is connected with fact table to generate a new dataset. Secondly, data in each row of the new dataset in HDFS forms a string as the *value* of the key-value pair which is inputted from the Map function of the MapReduce model. The row number is the *key* of the key-value pair. Finally, the dimensions are extracted from the *value* via Map operation to form a new string, which is the *key* of the Reducer. The information of all dimensions contained in the *value* is removed. Calculating metric is used as the *value* of the Reducer.

3 Design of cloud geodatabase access

Cloud geodatabase access is an important part of the cloud geodatabase application for the sponge city. It mainly includes spatial data storage layer, cloud geodatabase access layer, spatial data access layer and spatial data analysis layer. Each layer provides a series of standard application programming interfaces (APIs) for different functions. The functions of cloud geodatabase are abstracted and packaged into the engines to shield the differences among multi-source heterogeneous databases and provide a unified data access interface for the application layer. Figure 1 shows the architectural design of the cloud geodatabase access.

3.1 Spatial data storage layer

The spatial data storage layer is located at the bottom layer of the cloud geodatabase access. It is composed of many storage engines, such as stored procedures, triggers, and views, of an Oracle relational database and a Hadoop cloud geodatabase for sponge city. The Oracle relational database stores small amount of real-time data. It can also stores data that can be directly involved in model calculation, such as attribute data, metadata, water level and water quality. Cloud geodatabase for sponge city based on Hadoop stores large-scale raster and vector datasets. Then, the massive data in the cloud geodatabase are segmented, aggregated, cleaned and optimised using the MapReduce calculation model. It generates the relatively small ordered datasets to meet the high concurrent access demand of a large number of users [32].

The storage space of a storage node in the study is divided into several visual disk spaces (VDSs) [33]. The VDSs are adjacently encoded with Arabic numerals. It improves the parallel access and bandwidth utilisation of the cloud geodatabase for sponge city and shields the computer hardware details of the distributed database. The hardware configuration information is transparent within the cloud.

The mapping relation between tile file and VDS is established according to some vector or raster tile data attributes as follows:

$$E=(I_\mathrm{r}+I_\mathrm{c})\mathrm{Mod}\ N_\mathrm{max}$$

where E is a VDS-encoded tile datum. I_r and I_c are the row and column numbers of a tile data. Mod is modulo operator. N_max is the total number of VDSs.

According to the above equation, tile data with the same row number and column number are stored in the adjacent VDSs, whereas tile data with the same summation of row number and column

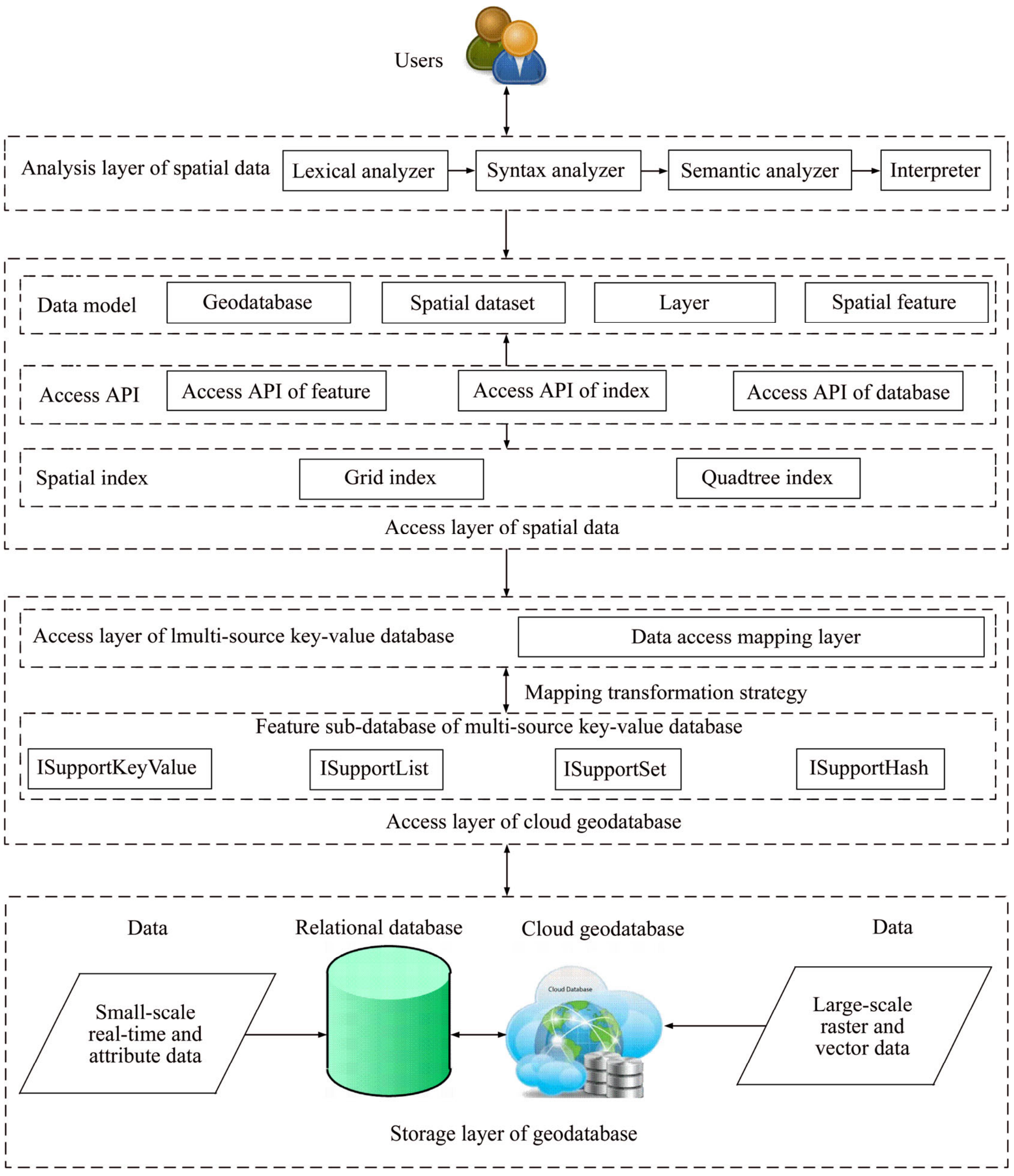
Users
Analysis layer of spatial data
Lexical analyzer
Syntax analyzer
Semantic analyzer
Interpreter
Data model
Geodatabase
Spatial dataset
Layer
Spatial feature
Access API
Access API of feature
Access API of index
Access API of database
Spatial index
Grid index
Quadtree index
Access layer of spatial data
Access layer of lmulti-source key-value database
Data access mapping layer
Mapping transformation strategy
Feature sub-database of multi-source key-value database
ISupportKeyValue
ISupportList
ISupportSet
ISupportHash
Access layer of cloud geodatabase
Data
Relational database
Cloud geodatabase
Data
Small-scale real-time and attribute data
Large-scale raster and vector data
Storage layer of geodatabase

Figure 1 Access structure of cloud geodatabase

number are stored in a VDS to sort them based on row major order.

The purpose of spatial data partitioning is to horizontally fragment spatial dimension tables via a spatial hierarchy and replicate the remaining tables of the cloud geodatabase for parallel processing according to spatial proximity. The start-point of data partitioning is the spatial data with the large level of granularity to avoid spatial data redundancy. A cloud node must store spatial data instances with large and fine levels of granularity. Let dataset $D=\{\cup_{I=1\cdots I}, s_i\}$, where I defines the number of partitions, s_i is the ith non-overlapping subset. Cloud node set $N=\{n_1, n_2, \cdots, n_j \cdots\}$, where n_j is the jth cloud node. $H_1 \le H_2 \le \cdots \le H_n$ is a hierarchy of spatial data which instances are $\{h_1^1, h_1^2 \cdots\}$, $\{h_2^1, h_2^2 \cdots\}$, $\cdots$, $\{h_n^1, h_n^2 \cdots\}$, respectively, where H_1 is the large level of granularity and H_n is the fine level of granularity. One or more instances $\{h_1^1, h_1^2 \cdots\}$ of H_1 are stored in the cloud node n_j ($n\ge1$). Each instance h_1^k of H_1 ($k\ge1$) and all instances h_l^k ($l>1$ and $k\ge1$) related to h_1^k are stored in the same cloud node n_j.

The abovementioned data block method can realise rapid location and flexible scheduling of spatial data. The spatial data are divided into tiles with unique identifiers. The target level index of geography is established for geographical target contained in a data block to realise multilevel index of spatial data. Each spatial data block is compressed, serialised and then stored in HDFS in binary stream form as a table cell. The tile data are stored and organised based on the vector or raster target record, and the block is used as a unit to compress. Redundant attribute data are stored in the Oracle relational database, including layer description table, spatial data table, information table of map sheet and its version and attribute information table. Spatial data stored in the Hadoop cloud geodatabase are used for load display and spatial query, whereas attribute data stored in the Oracle relational database are used to support attribute query. This method improves data storage granularity and loading speed and map display and spatial query performances.

3.2 Cloud geodatabase access layer

The cloud geodatabase access layer is located at the top of the spatial data storage layer. Spatial data access model is mapped into the spatial data of the cloud geodatabase for intelligent invocation, efficient and transparent access and communication with other databases. The cloud geodatabase access layer is divided into two sub-layers with small granularity. The lower sub-layer is the feature interface layer of the database. The features of each database are abstracted into a series of interfaces, and different databases implement different interface sets. Transparent access of the cloud geodatabase is realised by using a mapping conversion strategy according to the feature interface provided by the database. The corresponding features of the database are extended in the feature library if a new database is added in the cloud geodatabase.

The upper sub-layer is the data access mapping layer. To realise transparent access to data, the access commands of spatial data and spatial index are passed down from the spatial data access layer. They are transformed into a series of invocations of interface functions according to the mapping conversion strategy. The transformation can directly communicate with the feature interface library and shield the differences among the slave nodes. The mapping conversion strategy is described as follows: Firstly, judging whether the requested database directly supports a *set* data structure. If the *set* structure is supported, it is directly called. Otherwise, examing whether a *list* structure is supported. If supported, the *list* data structure is used to simulate the *set* data structure. Otherwise, the *KeyValue* data structure is simulated. The data access mapping feature of the database isolated the data model from the spatial database. Thus, the realisation of physical model is not dependent anymore on the specific database, which significantly improved the scalability of spatial data engine.

3.3 Spatial data access layer

The spatial data access layer is located at the top of the cloud geodatabase access layer, which is encapsulated into three modules: spatial access, data model and spatial index. The spatial access module, that is, access API module, provides all the interfaces that operate on the geospatial data. The interfaces are packaged into small modules, such as feature access, layer access, database access, spatial analysis and index access. The interfaces can access spatial geographical data with different levels and granularities for operation, implementation and query.

Spatial data access begins with cloud geodatabase. Spatial dataset information is obtained from the cloud geodatabase, and then layer information is acquired from the spatial dataset. Finally, the layer features are found via spatial index. Spatial data access includes the following steps (Figure 2):

1) The metadata of the cloud geodatabase is read and the spatial dataset information is obtained.

2) The metadata of the spatial dataset is read and the layer name is obtained.

3) The layer metadata is read through layer name, and the name and type of the spatial index are obtained.

4) Whether the index name is empty is determined. If index name is empty, the primary key index is read through the '*key*=Index: Primary Key-layer name' because of only one primary key index. The filtering results are obtained according to the minimum bounding rectangle (MBR) filtering and primary key index information.

5) If the index name is not empty, whether the index type is a quadtree index is determined. If it is a quadtree index, quadtree metadata is read by considering the index name. The node location of the quadtree in the spatial query window is calculated.

6) All node records on the path between the

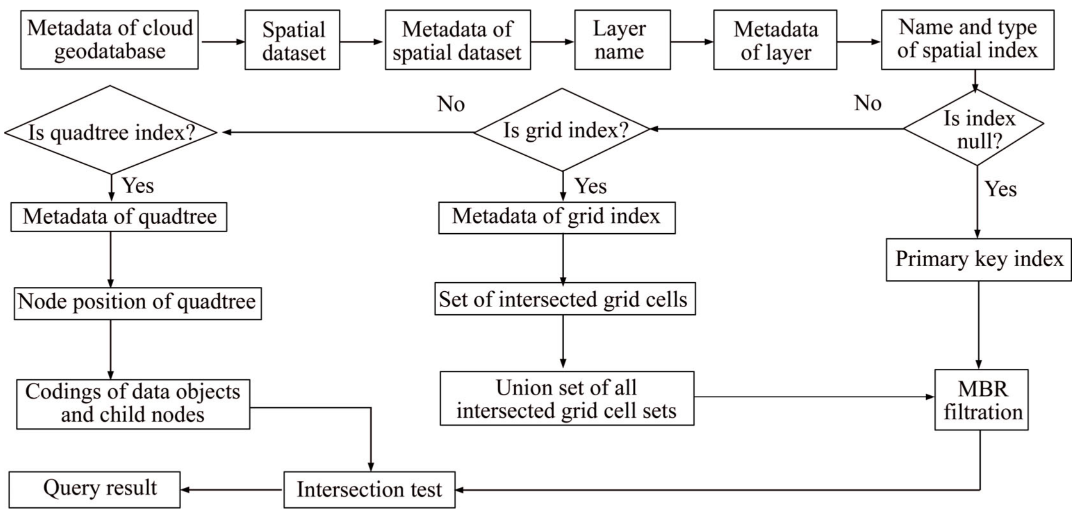
Metadata of cloud geodatabase
Spatial dataset
Metadata of spatial dataset
Layer name
Metadata of layer
Name and type of spatial index
Is index null?
No
Is grid index?
No
Is quadtree index?
Yes
Metadata of quadtree
Node position of quadtree
Codings of data objects and child nodes
Yes
Metadata of grid index
Set of intersected grid cells
Union set of all intersected grid cell sets
Yes
Primary key index
MBR filtration
Intersection test
Query result

Figure 2 Algorithm of spatial data access

root node and slave node of the quadtree are obtained. Then, the encoding of data object and preorder traversal encodings of all the slave nodes of the node are obtained.

7) Data object details are queried according to sequential coding. Whether the data object intersects or contains the query frame is judged.

8) If the index type is a grid index, the metadata information of the grid index is read through the index name.

9) The grid cell intersected by the grid index and the spatial query window are calculated. All the intersected code sets are obtained.

10) All the index information of the intersected grid cell is combined, and then the union sets are filtered according to MBR.

11) The filtered features are read, and whether they are a compulsory primary key index or an optional spatial index is judged. An intersection test is performed for the results filtered by MBR to obtain the final query results.

3.4 Spatial data analysis layer

The spatial data analysis layer is located at the top of the cloud geodatabase access. It supports the user-oriented declarative Map Query Language (MapQL), including lexical analysis, syntax analysis, semantic analysis, code generation and interpreter execution.

MapQL is a query language for semantically-translatable extracting and mapping from inclusion relation to classification of indivisuals according to description logic [34]. MapQL can convert from one representation to query mapping by using an operator, and to mapping expression by using a translator. For example, a MapQL model ‘*map_implies* (*C*, *D*)’, if its MapQL operator is ‘isa <*C*, *D*>’, then its response result is ‘{True, False}’. For another example, a MapQL model ‘*map_instance* (*C*(*a*), *D*)’, if its MapQL opterator is ‘translate <*C*, *a*, *D*>’, then its response result is ‘*D*(*a*) or None’. A MapQL request of the operator ‘*isa*’ is translated using JSON (JavaScript Object Notation) format, for example, as follows: {‘domain-class’: ‘LID’, ‘domain-instance’: ‘Roof Garden’, ‘range-class’: ‘Sponge city’}. The response will be returned ‘True’ if there is a mapping inclusion relation among ‘LID’, ‘Roof Garden’ and ‘Sponge city’. Otherwise, the response will be returned ‘False’ if the mapping does not exist or there is an inference error.

The specific processes of the layer are as follows: 1) The spatial data analysis layer accepts the MapQL input by a user and transmits it to a lexical analyser. 2) The lexical analyser performs lexical analysis of the MapQL, then generates a token stream and transmits the token stream to a syntax analyser. 3) The syntax analyser analyses the syntax of the token stream and obtains the structurised abstract syntax tree objects. 4) The spatial geographic data engine system analyses the abstract syntax tree semantics and constructs a corresponding execution plan. 5) Obtain the query results and return them to the user interface in the format of pictures, tables, vactors or images.

Implementation of lexical analyser. Lexical analysis reads source program characters one by one and identifies relatively independent tokens, also known as marks or words. The tokens include keywords such as *select*, *from*, *and* and *as*, identifier and special symbols. Unnecessary content such as annotation symbols is filtered down during the compilation process. The remaining content forms a sequential token stream. The general design process of the lexical analyser is as follows (Figure 3): 1) Aggregate the statement form and expression limitation supported by the descriptive language according to spatial query demand. 2) The descriptive language is decomposed into lexical granularities and expressed in terms of compiler syntax. 3) Query the symbol table of keywords. If the string is queried and found, it is a keyword. Otherwise, the string is an identifier. Token type is described by a regular expression to form a syntax regular expression. 4) Lexical analyser with a specified program language such as C source code is generated using an ANother Tool for Language Recognition compiler.

Implementation of syntax analyser. The syntax analyser accepts a token stream text file that conforms to the MapQL specification and then generates a syntax tree. Take the *SELECT* query statement of spatial data as an example, its general form is '*SELECT* < query object set > *FROM* < data source > *WHERE* < query condition >', where *SELECT* statement is an object property, which can provide an attribute name directly, or point out an attribute source of an object with the format of 'object.', or find all the attribute information of an object with the symbol of '*'. The *FROM* sub-clause is an object set that has already been found in a database, or acquired from the nested queries. The object result set must define an alias using the *AS* keyword to facilitate references to other sub-clauses. The *WHERE* sub-clause is optional, which represents the filter condition of the query, such as a logical expression, a function expression and a comparison among object attributes, or between object attributes and their values.

The semantic analyser checks static semantics, such as type checking and transformation of the syntactic unit in a syntax tree based on semantic rules. Semantic analysis does not need code generation. It iterates and analyses a syntax tree, and calls the corresponding interface to obtain the information of a string. The syntax analysis tree is a geometric arborescence representation of the abstract syntax structure of source code in the course of sentence pattern derivation. Meaningless nodes, such as separator ',' left parenthesis '(' and right parenthesis ')', are hidden when a syntax tree is built. Meanwhile, the syntax tree adds virtual nodes with abstract meaning and keywords with a unified form. Translation process is separated from the process of syntax analysis to reflect the natural and distinct hierarchical structure of language elements and explicit node meaning of syntax tree.

4 Design of cloud geodatabase retrieval

4.1 Direct addressing via file name

The relative path of tile data is adopted to store tile data for efficient data migration, backup, recovery and parallel access. It is determined by computation and query of global information. The front part of a relative path is composed of IP and VDS encoding of a storage node. The mapping table between a storage node and a VDS encoding determines a one-to-many relationship between IP and VDS encoding. A field in the mapping table exists to identify whether a storage node is an operation node. The latter part of the relative path is composed of the determinant attributes of tile data. Many tile data can exist in a VDS, thus its retrieval rate is far less than that of data records in a database. Therefore, stratifying the storage space is necessary in each VDS and directory structure in the virtual space according to the determinant attributes of tile data. For example, raster tile data are stratified as follows: '\Number of tile levels\Satellite name\Sensor name\Year of shooting image\Month of shooting image\Day of shooting image\Tile type (band file or thumbnail file)\Tile name'. The stratification can significantly reduce the number of files in a subdirectory and facilitate direct addressing via a file name in the case of the known path for high-efficiency concurrent retrieval.

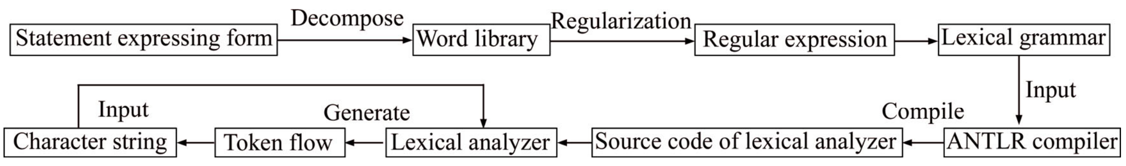

Figure 3 Implementation of lexical analyzer

Direct addressing via a file name utilises the file name of tile data and the mapping relationship between a storage node and VDS encoding. It has been initialised when the cloud geodatabase starts to provide services. Finding the file's storage path through a massive database is not required. The mapping relationship table between VDS encoding and storage node is only modified if data need to be transfered and recovered due to hardware failure. And continual update of the storage path of tile data is unnecessary. Direct addressing via file name and mapping relationships between the cloud centre and various clouds and between storage nodes and VDS encoding constitute the mapping part of MapReduce model. They can map file names of tile data into a storage path of tile file. Direct addressing via file name based on virtual distribution enhances the storage security and efficient retrieval of data.

4.2 Three-level concurrent retrieval

The cloud geodatabase for sponge city is composed of clouds with unpredictable number and no theoretical upper limit. Therefore, a three-level concurrent retrieval strategy is constructed to improve data storage efficiency, retrieval, backup, recovery and the overall system of concurrent execution mechanism (Figure 4).

The storage node in the bottom level of the architecture has multiple VDSs, each of which stores an index file of tile data. The number of VDSs determines the number of threads, which deploys request information from a storage node to VDSs and then process them separately [35]. Multiple thread processing results from multiple VDSs are aggregated through the interprocess communication channels provided by .NET Remoting technology. Microsoft.NET remote processing is a kind of network communication and transmission technology between the registered client and server using Hypertext Transfer Protocol and Transmission Control Protocol (TCP) / Internet Protocol. It speeds up the internal data exchange within a storage node and realises command issue, result collection, data communication and transmission among the storage nodes.

The cloud in the middle level has multiple storage nodes, which can apply multiple threads after the external service interface issued uniformly receives data processing request [36]. Each thread releases data request at the same time inside each node through the TCP channel of .NET Remoting technology. The results from parallel processing within the node are also aggregated to the operation node via the TCP channel. If the processing request comes from the outside of a cloud, the result returns to the requester via the web service. If the processing request comes from the inside of a cloud, it returns directly to the requester for quick data transmission.

Clouds on the top level constitute cloud geodatabase. Data processing requests at the clouds level are generally received from the external service interface of the cloud centre. The server of cloud centre can also apply multiple threads, each of which accesses the external service interface of the clouds or cloud centre via reflection calls. Data processing results are transmitted through the web service when performing global retrieval in the clouds [37]. A large amount of data will take a long period of time for retrieval.

4.3 Block data retrieval

Although the three-level concurrent processing architecture speeds up parallel data processing, long period of time may be consumed when a mass of query results are returned from a massive database. Furthermore, the user cannot quickly find one or several data records from the returned massive query results. Meanwhile, the user is difficultly utilising quadratic search from the returned massive query results [38]. Therefore, a block retrieval algorithm (Figure 5) is designed to improve data retrieval efficiency within the cloud geodatabase.

The block retrieval algorithm is described in detail as follows:

1) Compute the centroid of a space object according to the MBR.

2) Encode the longitude and latitude of the centroid with base32 and then combine them according to the Geohash algorithm.

3) Obtain and sort the Geohash value of the target object.

4) Divide the Geohash value into *n* blocks according to the sorting code.

5) Establish the spatial index for the geographic object of each block according to Hilbert curve. The index information from the query condition and query result is transmitted based on cache mechanism for retrieval efficiency.

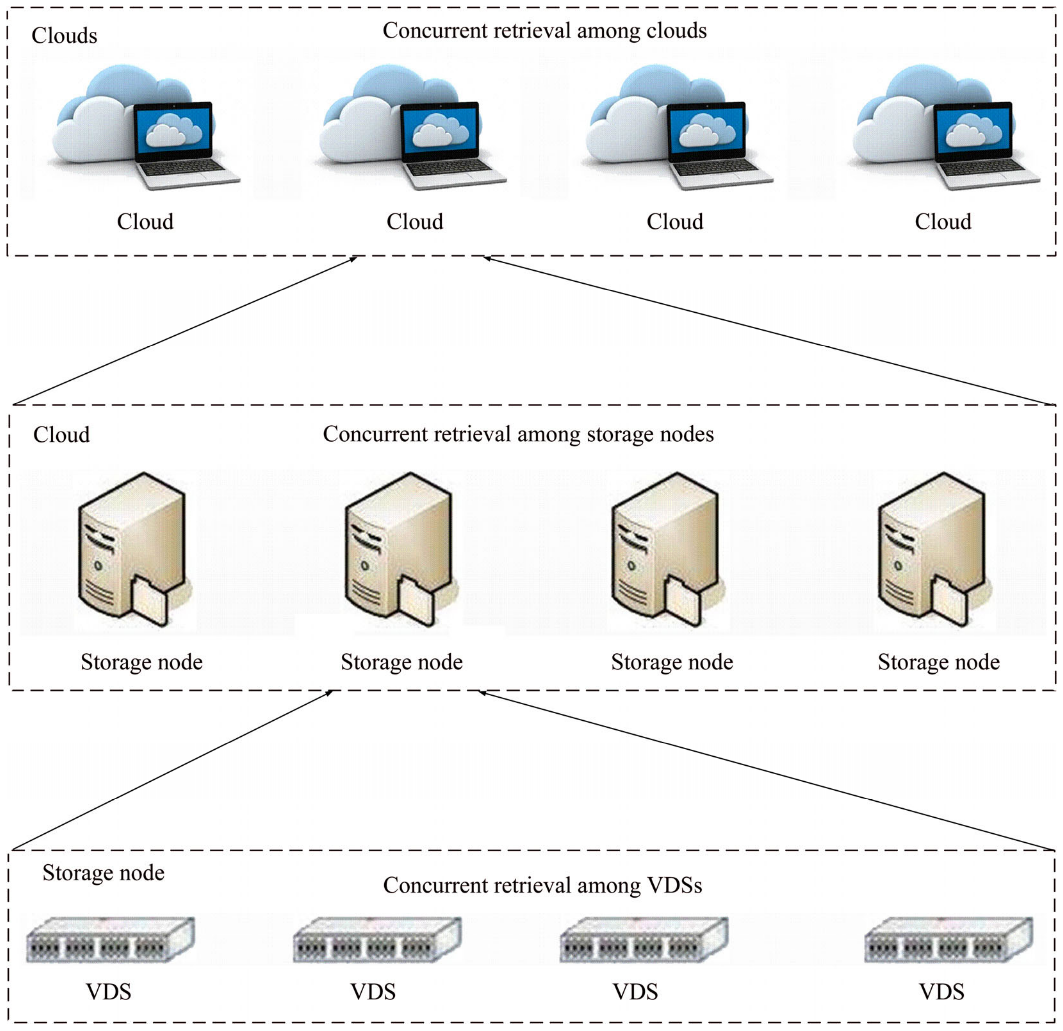
Clouds
Concurrent retrieval among clouds
Cloud
Cloud
Cloud
Cloud
Cloud
Concurrent retrieval among storage nodes
Storage node
Storage node
Storage node
Storage node
Storage node
Concurrent retrieval among VDSs
VDS
VDS
VDS
VDS

Figure 4 Architecture of three-level concurrent retrieval

Each query fragment can independently run in the table fragments of the slave node. Each block generally contains the same number of records.

6) Deploy data in parallel among different physical nodes. Each slave node supports local index.

7) Receive data query request, judge whether the request is new or the same as the previous request.

8) If the request is a new query combination, hand out the request to each cloud, and then to each node. The algorithm returns to step (10). If the request is consistent with the last request, the algorithm judges whether the last request result is timeout or not.

9) If the request has exceeded the time threshold, the query results in the database are updated, thus they are not used again for the last request result. The algorithm returns to step (10). If the request is not a timeout, the data in the database are not updated, thus the block index can be intercepted directly from the global retrieval list. The algorithm returns to step (12).

10) Process the query request in each node and cloud. Return and aggregate a large amount of query results from each cloud, storage node and VDS.

11) The aggregated results are encoded, and the index of each data record is generated. For example, given 10 query records in the VDS1, the indices of each record are designed as follows: 1_1, 1_2, ⋯ and 1_10. Then, one index is taken out from an index list. And it added to the global index list of the query request. The index is deleted from the index list until all indices of the index list are added to the global index list.

12) User sets the block number of the block index list from each VDS, storage node and clouds, in which stores the initial index and offset information of the block data. Intercept and process

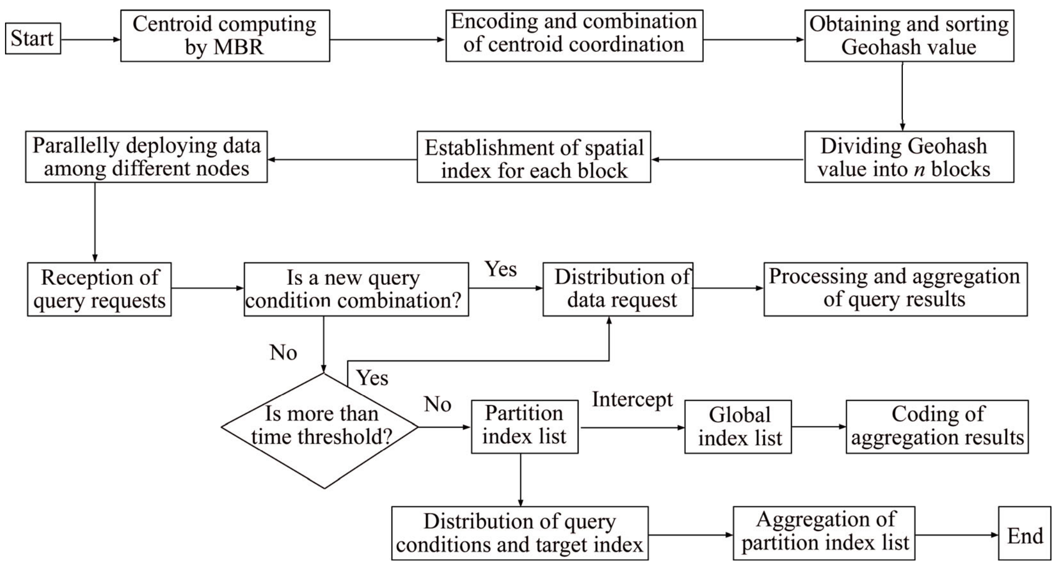
Start
Centroid computing by MBR
Encoding and combination of centroid coordination
Obtaining and sorting Geohash value
Dividing Geohash value into n blocks
Establishment of spatial index for each block
Parallelly deploying data among different nodes
Reception of query requests
Is a new query condition combination?
Yes
Distribution of data request
Processing and aggregation of query results
No
Yes
Is more than time threshold?
No
Partition index list
Intercept
Global index list
Coding of aggregation results
Distribution of query conditions and target index
Aggregation of partition index list
End

Figure 5 Chunked retrieval algorithm

the block index list from the global index list according to the block number.

13) Query condition and target index information are redeployed. Query and block index are re-encoded into each cloud and storage node to obtain data records corresponding to each encoding.

14) The returned record lists from each VDS, storage node and clouds submit to the master node, and then jointly presented to the user. The algorithm is finished.

5 Implementation and validation of cloud geodatabase

5.1 Implementation of cloud geodatabase

The experiments on the Hadoop cloud geodatabase and Oracle relational database are simulated on Windows machines to validate the performance of the Hadoop cloud geodatabase. The Hadoop server cluster consists of a cloud centre in the construction headquarter for sponge city. Six slave nodes are distributed in the departments, such as meteorological bureau, water conservancy bureau, planning bureau, housing and urban construction department, land resource bureau and institute of remote sensing surveying and mapping in Yinchuan City, Ningxia, China. HDFS is deployed to the servers of the seven departments. Oracle 10g is used as a relational database and deployed in seven Windows computers, which had the same machine configuration as the Hadoop 0.20.2 cluster machines. Four client machines are tested, which are configured with 2 GB memory and 2.83 GHz quad core CPU. Various client browser interfaces are designed by using Active Server Pages technology. Figure 6 shows one of the interfaces to illustrate the zones of the sponge city. The top of the browser interface provides with some basic tools, such as map operation, query and find, layer control and thematic mapping. The middle of the browser interface is the map view area. The bottom of the browser interface displays the work log, including real-time operation time, map tool, coordinates and other information.

5.2 Validation of concurrent access to cloud geodatabase

By considering the layer of ‘Housing in Yinchuan City’ as an example, the layer has the maximum number of features with 715942 housings. A new file is specially created to define 500 rectangles, which covered different spatial ranges. The contained or intersected geographical objects are queried and loaded by using the rectangles as a spatial query condition. Four clients are used to simulate multiple users in a multi-threaded manner. Spatial queries are performed on the Oracle database and Hadoop cloud geodatabase. Each computer simulated 20 concurrent users at the outset. The query condition is executed iteratively for 5 min. Afterwards, each computer increased 20 concurrent users every 5 min. The average time spent in the spatial query of concurrent users for the Oracle and Hadoop databases is recorded as shown in Figure 7.

As shown in Figure 7, with the increase of the

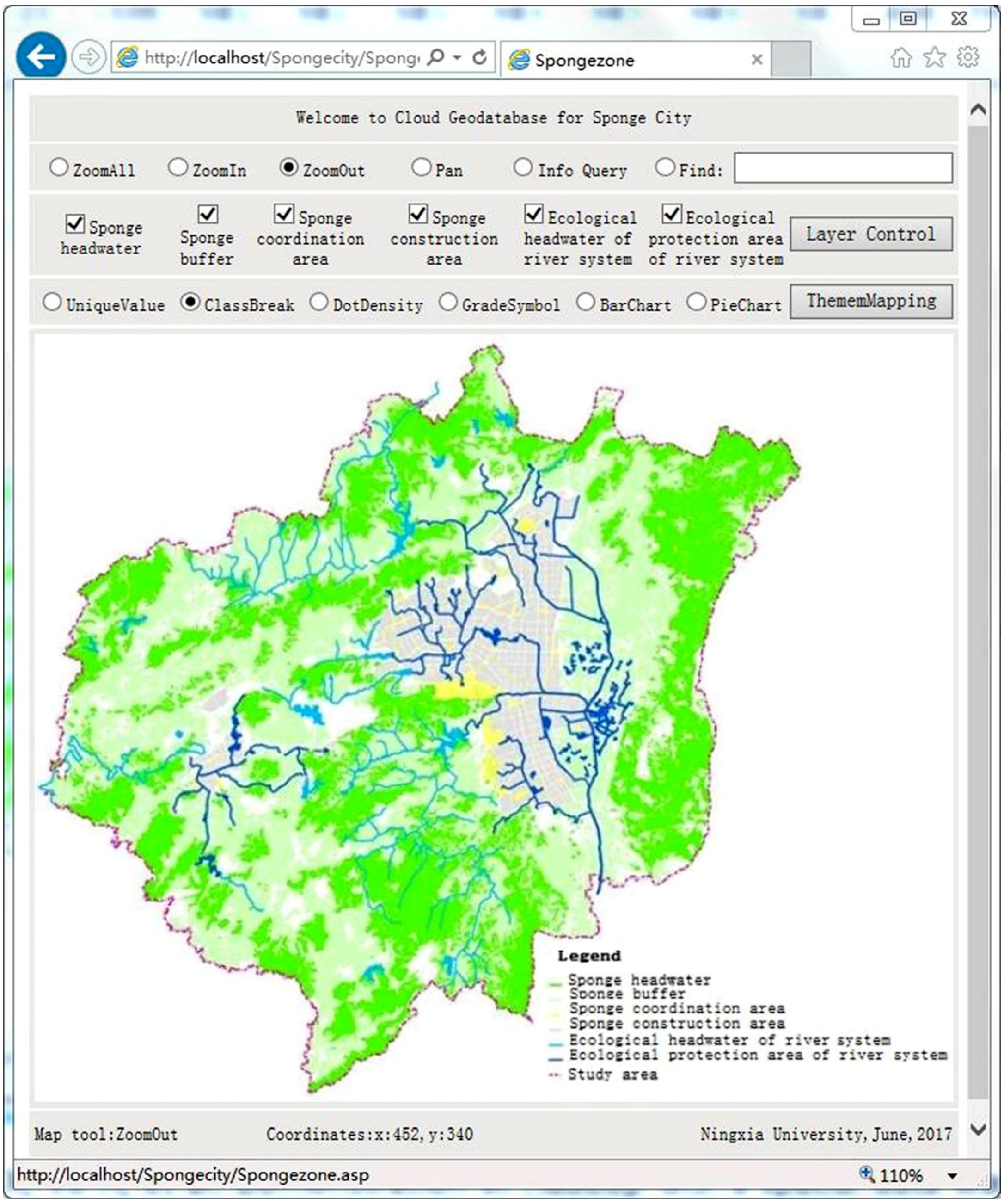

Figure 6 Browser interface for zones of sponge city

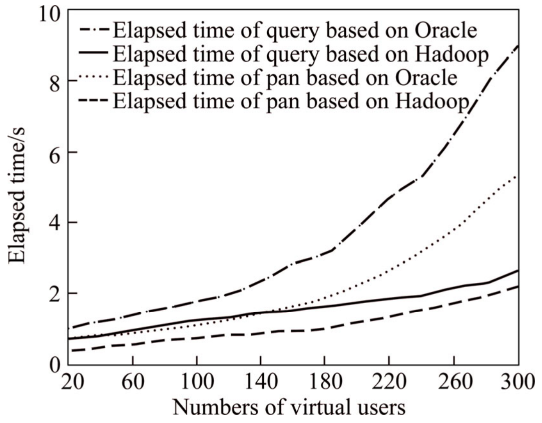

Figure 7 Experimental result of elapsed time for spatial query and pan based on Oracle and Hadoop

number of concurrent users, the response time of spatial queries based on the Oracle and Hadoop database schemes have all increased. However, the query time of the Oracle geodatabase is constantly longer than that of the Hadoop geodatabase. The time spent on request and response of the Oracle geodatabase is relatively stable in the case of a small number of concurrent accesses. However, the request and response time will increase intensely when the number of concurrent access exceeds a certain threshold. This condition is possibly because that the long period of communication time among the nodes is occupied. On the contrary, time spent on request and response of the Hadoop geodatabase increases slowly and maintain relatively stable although a large number of users for concurrent access existed. Thus, the Hadoop geodatabase can simply handle highly concurrent spatial query requests, whereas the Oracle geodatabase cannot meet the requirements of high concurrent access.

The performances of the geodatabases based on Oracle and Hadoop are then tested in the case of highly concurrent pan scenarios using the same method as the highly concurrent query. The testing process is described as follows. A display area in the layer of 'Yinchuan City Housing' is considered as the starting point. One-half map sheet within the scope of two map sheets close to the display area is loaded at a time to simulate the pan operation. To avoid the possibility of selecting from the area of the starting point on the experimental results, 10 starting regions are randomly selected within the layer before the experiment started. After completing a pan of starting ranges, the next area is navigated until the 10 regions are all navigated. The total time spent in the 10 pan operations is recorded and then averaged. Then, the number of concurrent users for pan operation increases gradually to validate the elapsed time of pan operation spent on the Oracle and Hadoop geodatabases (Figure 7).

As shown in Figure 7, when the number of concurrent users is small in the experiment of map pan operation, the performance of the Oracle scheme is quite similar to that of the Hadoop scheme. However, when the number of concurrent users exceeds a certain threshold, the performance of the Oracle relational database decreases intensely and the pan response time is significantly increased. However, the number of concurrent users slightly affected the performance of the Hadoop distributed cluster. When 300 virtual users exist, the map pan time for loading adjacent data to the Oracle scheme is 2.5 times as fast as that of the Hadoop scheme.

A contrast experiment between the Oracle database and the Hadoop cloud geodatabase is carried out to verify the efficiency of MapReduce for construction of parallel spatial index. The numbers of Mapper and Reducer on a single node are set to 2. The block size in HDFS is set to 128M. Construction time of parallel spatial index based on different data quantity by using Oracle and Hadoop, respectively, is shown in Table 2.

As shown in Table 2, construction times of parallel spatial index from both Oracle database and Hadoop cloud geodatabase gradually increase with the increase of data volume. The advantage of cloud geodatabase in the parallel performance of spatial index is not very obvious compared with that of single machine environment when the amount of processed spatial data is relatively small. The parallel generation of spatial index of Oracle is faster than that of the Hadoop cloud geodatabase because there is an excellent index mechanism at the bottom of the Oracle database. However, the parallel generation time of spatial index spent from the Hadoop cloud geodatabase is much shorter than that from the Oracle database when the volume of spatial data is greater than 200TB.

6 Conclusions

In the study, a cloud geodatabase for sponge city is designed and achieved. virtualisation, geohash fragmentation, parallel computing and WebService architecture are first used to design Hadoop distributed computing framework with a master and multiple slaves. Spatial data storage layer, cloud geodatabase access layer, spatial data access layer and spatial data analysis layer are built and integrated into cloud geodatabase access for the sponge city. Direct addressing via a file name, three-level concurrent retrieval and block data retrieval are then designed for retrieval model. The cloud geodatabase for sponge city is finally implemented and validated by using the comparisons of the Hadoop cloud geodatabase with Oracle database. It is shown that the cloud geodatabase for sponge city can store and retrieve massive data with high-performance computing and communication.

The cloud geodatabase for the sponge city has high scalability, high availability, large-scale parallel processing and effective resource allocation. It can flexibly store and manage heterogeneous data and massive data to integrate and organise the

Table 2 Construction time of parallel spatial index based on different data quantity by using Oracle and Hadoop

Data/TB	1.7	19.2	56.8	91.7	118.6	225.5	375.8	569.4
Oracle/s	3.65	8.02	15.48	23.68	30.92	57.26	84.04	130.58
Hadoop/s	5.26	12.52	20.58	31.94	38.32	46.08	56.83	79.15

multi-source, multi-scale and massive basic spatial data of the sponge city. It can virtualise many backend functions, and avoid redundant configuration of personnel, hardware and software. Meanwhile, the cloud geodatabase can significantly enhance the database storage capacities, retrieval, access, sharing and security for the construction and management of the sponge city.

The future study is mainly 1) to adjust joining strategy of multiple tables to reduce communication and unnecessary repetitive operation; 2) to propose a locking scheme based on attribute predicates with multiple granularities to realize fast lookup of conflict predicates; 3) to dynamically partition concepts according to the existing data in a cloud database; 4) to carry out the real-time monitoring, defect diagnosis and data mining by using cloud computing technology.

References

[1] HOU Jing-wei, LI Long-tang, HE Jie. Detection of grapevine leafroll disease based on 11-index imagery and ant colony clustering algorithm [J]. Precision Agriculture, 2016, 17(4): 488–505. DOI: 10.1007/s11119-016-9432-2.

[2] HOU Jing-wei, FAN Xin-gang, LIU Ren-tao. Optimal spatial allocation of irrigation water under uncertainty using the bilayer nested optimisation algorithm and geospatial technology [J]. International Journal of Geographical Information Science, 2016, 30(12): 2462–2485. DOI: 10.1080/13658816.2016.1181264.

[3] ZHOU Xiao-guang, CHEN Jun, JIANG Jie, ZHU Jian-jun, LI Zhi-lin. Event-based incremental updating of spatio-temporal database [J]. Journal of Central South University of Technology, 2004, 11(2): 192–198.

[4] LEI Xiang-dong, ZHAO Yue-long, CHEN Song-qiao, YUAN Xiao-li. Scheduling transactions in mobile distributed real-time database systems [J]. Journal of Central South University of Technology, 2008, 15: 545−551. DOI: 10.1007/s11771−008−0103−y.

[5] HOU Jing-wei, MI Wen-bao, LI Long-tang. Spatial quality evaluation for drinking water based on GIS and ant colony clustering algorithm [J]. Journal of Central South University of Technology, 2014, 21(2): 1051–1057. DOI: 10.1007/s11771-014-2036-y.

[6] JANAKIRAMAN K K, ORGUN M A, NAYAK A. Geospatial editing over a federated cloud geodatabase for the state of NSW [C]// Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems. San Jose: ACM, 2010: 144–151. DOI: 10.1145/1869790.1869813.

[7] VORA A, JAIN D, MAVANI G, SHAH S. Incorporating database management system in cloud [J]. International Journal of Scientific Research and Education, 2016, 5(11): 61–64. DOI: 10.17148/IJARCCE.2016.51112.

[8] LI Wei-yue, LIU Chun, HONG Yang, ZHANG Xin-hua, WAN Zhan-ming, SAHARIA M, SUN Wei-wei, YAO Dong-jing, CHEN Wen, CHEN Sheng, YANG Xiu-qin, YUE Jing. A public cloud-based China's landslide inventory database (CsLID): Development, zone, and spatiotemporal analysis for significant historical events, 1949–2011 [J]. Journal of Mountain Science, 2016, 13(7): 1275–1285. DOI: 10.1007/s11629-015-3659-7.

[9] DŐRNEMANN T, JUHNKE E, FREISLEBEN B. On-demand resource provisioning for BPEL workflows using Amazon's elastic computer cloud [C]// Processing of the 9th IEEE/ACM Int Symposium on Cluster Computing and the Grid. Shanghai, China: TEEE, 2009: 140–147.

[10] KANG C. Cloud computing and its Applications in GIS [D]. Ann Arbor: Clark University, 2011.

[11] BALKIĆ Z，ŠOŠTARIĆ D，HORVAT G. GeoHash and UUID identifier for multi-agent systems [M]// Agent and Multi-Agent Systems Technologies and Applications. Berlin, Springer, 2012, 7327: 290–298. DOI: 10.1007/978-3-642-30947-2_33.

[12] ELDAWY A. MOKBEL M F. SpatialHadoop: A MapReduce framework for spatial data [C]// IEEE International Conference on Data Engineering, Seoul: IEEE, 2016: 1352–1363.

[13] SRIVASTAVA P, BINH N T, KHARE A. Content-based image retrieval using moments [C]// Context-Aware Systems and Applications. ICCASA 2013. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Berlin: Springer, 2014, 128: 228–237. DOI: 10.1007/978-3-319-05939-6_23.

[14] ZHANG Ming-jin, WANG Hui-bo, LU Yun, LI Tao, GUANG Yu-dong, LIU Chang, EDROSA E, LI Hong-tai, RISHE N. TerraFly GeoCloud: An online spatial data analysis and visualization system [J]. ACM Transactions on Embedded Computing Systems, 2010, 9(4): 1–24. DOI: 10.1145/2700494.

[15] DING Zhi-ming, GUO Li-min, YANG Qi. RDB-KV: A cloud database framework for managing massive heterogeneous sensor stream data [C]// Second International Conference on Intelligent System Design and Engineering Application. Sanya: IEEE, 2011: 653–656. DOI: 10.1109/ISdea.2011.33.

[16] LEE D W, LIANG S H L. Geopot: A Cloud-based geolocation data service for mobile applications [J]. International Journal of Geographical Information Science, 2011, 25(8): 1283–1301. DOI: 10.1080/13658816.2011.558017.

[17] BHAT M A, SHAH R M, AHMAD B. Cloud computing: a solution to geographical information systems (GIS) [J]. International Journal on Computer Science and Engineering, 2011, 3(2): 594–600.

[18] YANG Chao-wei, GOODCHILD M, HUANG Qun-ying, NEBERT D, RASKIN R, XU Yan, BAMBACUS M, FAY D. Spatial cloud computing: How can the geospatial sciences use and help shape cloud computing [J]. International Journal of Digital Earth, 2011, 4(4): 305–329. DOI: 10.1080/17538947. 2011.587547.

[19] AJI A, SUN Xi-ling, VO H, LIU Qiao-ling, LEE R, ZHANG Xiao-dong, SALTZ J, WANG Fu-sheng. Demonstration of Hadoop-GIS: A spatial data warehousing system over MapReduce [J]. Advances in Geographic Information Systems, 2013, 6(11): 528–531. DOI: 10.1145/2525314. 2525320.

[20] CHEN Da-lun, CHEN Rong-guo, XIE Jiong. Research of the parallel spatial database proto system based on MPP architecture [J]. Journal of Geo-information Science, 2016,

18(2): 151–159. DOI:10.3724/SP.J.1047.2016.00151.

[21] LIU Bao-ling, LI Gang, YOU Hong, SUI Ming-rui, WANG Shu-tao. Evaluation of dynamic groundwater quality simulation based on Cloud-GIS: A case study in Harbin urban area, China [J]. Water Science & Technology: Water Supply, 2014, 14(6): 1095–1103. DOI: 10.2166/ws.2014.070.

[22] WEI Ling-yin, HSUA Ya-ting, PENGA Wen-chih, LEE Wang-chien. Indexing spatial data in cloud data managements [J]. Pervasive and Mobile Computing, 2014, 15(12): 48–61. DOI: 10.1016/j.pmcj.2013.07.001.

[23] MATEUS R C，SIQUEIRA T L L，TIMES V C， CIFERRI R R， CIFERRI C D D A. Spatial data warehouses and spatial OLAP come towards the cloud: design and performance [J]. Distributed and Parallel Databases, 2015, 4: 1–37. DOI: 10.1007/s10619-015-7176-z.

[24] KUNE R, KONUGURTHI P K, AGARWAL A, CHILLARIGE R R, BUYYA R. XHAMI – extended HDFS and MapReduce interface for big data image processing applications in cloud computing environments [J]. Software: Practice and Experience, 2017, 47: 455–472. DOI: 10.1002/ spe.2425.

[25] WANG Yong, LIU Zhen-ling, LIAO Hong-yan. Improving the performance of GIS polygon overlay computation with MapReduce for spatial big data processing [J]. Cluster Computing, 2015, 18(2): 507–516. DOI: 10.1007/s10586-015-0428-x.

[26] YAN Ji-ning, MA Yan, WANG Li-zhe. A cloud-based remote sensing data production system [J]. Future Generation Computer Systems, 2018, 86(9): 1154–1166. DOI: 10.1016/ j.future.2017.02.044.

[27] LIN Feng-cheng, CHUNG Lan-kun, WANG Chun-ju. Storage and processing of massive remote sensing images using a novel cloud computing platform [J]. GIScience & Remote Sensing, 2013, 50(3): 322–336. DOI: 10.1080/ 15481603.2013.810976.

[28] WANG Peng-yao, WANG Jian-qin, CHEN Ying. Rapid processing of remote sensing images based on cloud computing [J]. Future Generation Computer Systems, 2013, 29(8): 1963–1968. DOI: 10.1016/j.future.2013.05.002.

[29] CHEN Hung-ming, CHANG Kai-chuan, LIN Tsung-hsi. A cloud-based system framework for performing online viewing, storage, and analysis on big data of massive BIMs [J]. Automation in Construction, 2016, 71: 34–48. DOI: 10.1016/j.autcon.2016.03.002.

[30] CARY A, YESHA Y, ADJOUADI M, RISHE N. Leveraging cloud computing in geodatabase management [C]// IEEE International Conference on Granular Computing. San Jose: IEEE, 2010: 73–78.

[31] CAFARELLA M, CHANG E, FIKES A, HALEVY A, HSIEH W, LERNER A, MADHAVAN J, MUTHUKRISHNAN S. Data management projects at Google [J]. ACM Signod Record, 2008, 37(1): 34–38. DOI: 10.1145/1374780.1374789.

[32] ZHENG Kun, FU Yan-li. Research on vector spatial data storage schema based on Hadoop platform [J]. International Journal of Database Theory and Application, 2013, 6(5): 85–94. DOI: 10.14257/ijdta.2013.6.5.08.

[33] XU Xiao-long, ZHANG Qi-tong, ZHOU Jing-lan. NC-MACPABE: Non-centered multi-authority proxy re-encryption based on CP-ABE for cloud storage systems [J]. Journal of Central South University, 2017, 24(4): 807−818. DOI: 10.1007/s11771-017-3483-z.

[34] RYOTA H, KIMIO K. MapQL: Map-based Ontology Query Language [C]// DEWS2008, C3-2. 2008: 1–8. (in Japanese)

[35] DUAN L F. The research and application of cloud database technology for remote sensing data [D]. Kaifeng: Henan University, 2014. (in Chinese)

[36] ZHANG Xiu-ling, GAO Wu-yang, LAI Yong-jin, CHENG Yan-tao. Flatness predictive model based on T-S cloud reasoning network implemented by DSP [J]. Journal of Central South University, 2017, 24(10): 2222−2230. DOI: 10.1007/s11771-017-3631-5.

[37] YAO Guang-shun, DING Yong-sheng, HAO Kuang-rong. Multi-objective workflow scheduling in cloud system based on cooperative multi-swarm optimization algorithm [J]. Journal of Central South University, 2017, 24(5): 1050−1062. DOI: 10.1007/s11771-017-3508-7.

[38] VIJAYA P, RAJU G, RAY S K. Artificial neural network-based merging score for Meta search engine [J]. Journal of Central South University, 2016, 23(10): 2604−2615. DOI: 10.1007/s11771-016-3322-7.

(Edited by HE Yun-bin)

中文导读

海绵城市云空间数据库设计与实现

摘要：构建海绵城市云空间数据库可以整合不同管理部门分散的地理空间信息，实现海量数据的多用户高并发访问和检索、高可扩展性和可用性、高效存储和管理。为了处理海量空间数据，本研究首先利用并行计算技术设计了包括 Hadoop 分布式文件系统和 MapReduce 的 Hadoop 分布式计算框架。其次，用一系列标准 API 设计了访问控制模块，包括空间数据存储层、云空间数据库访问层、空间数据访问层和空间数据分析层。然后，设计了一个检索模型，包括利用文件名直接寻址、3 层并行检索和数据块检索策略。实现了多尺度、多源和海量空间数据的实时并行访问，高性能计算、通讯、存储、高效检索和时序安排等功能。最后，通过与 Oracle 数据库的比较，验证了 Hadoop 云空间数据库的性能。海绵城市云空间数据库能避免人员、硬件和软件资源的冗余配置，支持数据传输、模型调试和应用开发，为海绵城市建设和管理提供基础和专题地理信息的精确、实时、虚拟、智能、可靠、动态、按需和弹性可扩展的云服务。

关键词：云空间数据库；海绵城市；分布式计算；并发检索；访问