An Efficient Organization Method for Large-Scale and Long Time-Series Remote Sensing Data in a Cloud Computing Environment

Jining Yan, Yuanxing Liu, Lizhe Wang, *Fellow, IEEE*, Zhipeng Wang, Xiaohui Huang, and Hong Liu

***Abstract*—Historical earth observation (EO) data have played an important role in long-term scientific and environmental monitoring. The effective organization of large-scale and long-term remote-sensing data to achieve efficient retrieval and access has become one of the important issues. However, inherent big data characteristics, such as a large scale, and asymmetrical temporal and spatial distributions, have caused problems with the efficiency of data retrieval and access. Therefore, this study proposes an efficient data organization method for use in a cloud-computing environment that has two aims. First, it addresses the problem of low retrieval efficiency. An asymmetrical index model for the image metadata is constructed that is based on a unified spatio-temporal grid coding; a prepartitioning mechanism under the HBase architecture is established to realize the uniform storage of the metadata with an asymmetrical spatiotemporal distribution and to avoid retrieval efficiency bottlenecks caused by a load imbalance. Second, it addresses low access efficiency. By dividing the remote-sensing image into tiles, a unified spatio-temporal code is established for each tile, and a consistent hash operation is performed; tiles with similar hash values are stored in the same or adjacent Hadoop Distributed File System nodes. In this way, tiles with temporal or spatial correlations can be gathered and stored, and lots of disk seeks can be avoided during retrieval, thereby greatly improving the data access efficiency. Comparative experiments showed that the data organization method can effectively improve the retrieval and access efficiencies of large-scale and long time-series remote-sensing data in a cloud-computing environment.**

***Index Terms*—Cloud computing, data management, data organization, remote sensing, time series.**

Nomenclature

EO	Earth observation.
HDFS	Hadoop Distributed File System.
GFS	Google File System.
TB	TeraByte.
NetCDF	Network Common Data Form.
API	Application Programming Interface.
MB	Million Byte.
GB	Giga Byte.
ISO	International Organization for Standardization.
JSON	JavaScript Object Notation.

Manuscript received July 17, 2021; revised August 23, 2021; accepted September 5, 2021. Date of publication September 8, 2021; date of current version September 27, 2021. This work was supported by the National Natural Science Foundation of China under Grant 41801363 and Grant 41925007. *(Corresponding author: Lizhe Wang.)*

The authors are with the School of Computer Science, China University of Geosciences, Wuhan 430074, China, and also with the Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430074, China (e-mail: yanjn@cug.edu.cn; liuyx@cug.edu.cn; lizheng.wang@gmail.com; wangzp@cug.edu.cn; xhhuang@cug.edu.cn; liuhong@cug.edu.cn).

Digital Object Identifier 10.1109/JSTARS.2021.3110900

I. Introduction

Large-scale and long time-series remote-sensing data will inevitably produce repeated observations of the same area, providing sufficient and usable data sources for the detection of land cover changes [1], urban expansion analysis [2], and environmental pollution prevention and control [3]. However, the massive, multisource, heterogeneous, and long time-series characteristics of remote-sensing data also bring great challenges to data organization and management, such as making retrieval and access efficiency low, and not convenient to comprehensively and in-depth analyze the hidden information from multiple dimensions and multiple angles.

In terms of the abovementioned problems, some scholars have proposed carrying out geometric and radiation normalization on long-term remote-sensing data, splitting and organizing the segmented tiles into a three-dimension datacube [4] in chronological order, and finally, performing information extraction and time sequence analysis using the constructed datacube. However, this spatio-temporal sequence organization of large-scale and long time-series remote-sensing data still has the following problems.

1) The DataCube stores the remote-sensing tile metadata and catalog data directly in the PostgreSQL database in the JavaScript object notation (JSON) document format by default [5]. However, due to the performance limitations of the PostgreSQL database in terms of expansion and concurrent access, such as the maximum relation size of 32 TeraByte (TB) and the maximum number of columns per table 1600 [6], this mode cannot meet the unlimited growth of remote-sensing tile data storage and management requirements.
2) The remote-sensing tile storage does not fully consider the temporal and spatial logical relationships, and the retrieval efficiency of massive remote-sensing tiles is not high [7]. The DataCube model stores the segmented tiles in file directories and does not consider the spatial proximity and time-series relationships between the segmented tiles, which makes tiles from the same or similar areas lack logical relevance, and the retrieval efficiency is not high.

3) The DataCube model is based on the Python or Dask parallel computing libraries for multiprocess parallel processing, which are not conducive to a large data volume, multitasking, and multinode parallel computing and cannot meet the rapid processing needs of massive remote-sensing tiles [8].

Dask is a Python parallel-computing tool based on external memory. By dividing a dataset into blocks and allocating the amount of calculation according to the number of cores it has, Dask can improve the parallel-computing efficiency of a single machine to a certain extent [9]. However, Dask should manually intervene in data scheduling between clusters. When the number of parallel tasks increases, the task scheduling of Dask often deadlocks, which is very unfavorable for large data volumes, multitasking, and multinode parallel computing [10]. Massive remote-sensing tiles after segmentation will inevitably be stored in multiple data nodes in a distributed manner, and Dask will not be able to meet their rapid-processing needs.

The abovementioned DataCube model's shortcomings in terms of index management and the rapid processing of massive tiles have limited its application in information extraction and mining. Therefore, how to effectively manage large-scale and long time-series remote-sensing data to achieve efficient indexing and retrieval and how to quickly access and obtain large-scale remote sensing data to conduct multidimensional and multiangle information extraction and mining, are urgent needs in the field of EO.

This study aims at the shortcomings of the current remote-sensing big data spatiotemporal sequence organizational model, takes advantage of cloud computing's advantages in massive data storage management and processing, and establishes a cloud computing-based remote-sensing big data spatiotemporal asymmetrical index and distributed storage optimization model. Finally, this study performs large-scale and long-term data retrieval in Hadoop ecosystem, which is a mature and popular used distributed system, to verify the effectiveness of the model.

In particular, the study accomplishes the following.

1) It addresses the problem of low retrieval efficiency caused by the characteristics of a massive amount of remote-sensing data with an asymmetrical temporal and spatial distribution. An asymmetrical index model for the image metadata under the HBase [11] storage architecture is constructed that is based on a unified spatio-temporal grid coding of the remote-sensing image metadata; a prepartitioning mechanism under the HBase architecture is established to realize uniform storage of remote-sensing image metadata with an uneven temporal and spatial distribution and to avoid retrieval efficiency bottlenecks caused by a load imbalance.
2) It addresses low access efficiency caused by the massive number of remote-sensing image files. A distributed storage optimization strategy for remote-sensing tiles based on consistent hashing is constructed. That is, by dividing the remote-sensing image file into tiles (each with a size of about 128 MB), a unified spatio-temporal grid code is established for each tile, and a consistent hash operation is performed; tiles with similar hash values are stored in the same or adjacent Hadoop Distributed File System (HDFS) [12] nodes. In this way, remote-sensing tiles with temporal or spatial correlations can be gathered and stored, and a large number of disk seeks can be avoided during spatio-temporal retrieval, thereby greatly improving the efficiency of data access.

II. Related Work

The efficient organization and management of long time-series remote sensing data can provide guarantee for in-depth data analysis and understanding [13], [14], and effectively improve the convenience of data mining. At present, the storage and management of remote-sensing data mostly adopts a hybrid method of file system and database; that is, remote-sensing image files are stored on a file system other than the database system, and the database system stores and manages attribute data and catalog information [15]. Therefore, the related work research focuses on the following two aspects: 1) remote-sensing data storage and management and 2) the spatio-temporal organization and indexing of metadata.

A. Remote-Sensing Data Storage and Management

In general, distributed file systems commonly used for remote-sensing data storage include the Google File System (GFS) [16], HDFS [17], Lustre [18], FastDFS [19], GridFS [20], GlusterFS [21], MooseFS [22], etc. Luo *et al.* [23] proposed RAMCloud, a cloud storage system, to achieve efficient random reading in a cloud environment. The storage system is based on the distributed file system HDFS, which improves the throughput and I/O performance of the system by reading and writing data in the memory and has good availability and scalability. Qin *et al.* [24] proposed a remote-sensing data storage model, which used GridFS [20] to store the remote-sensing images and a MongoDB database to store their relevant attribute information. Experimental results showed that this method was superior to relational databases in all aspects.

Remote-sensing data storage and management based on distributed storage-related technologies has very important research value and practical significance. However, for specific business scenarios, native distributed file systems still need to be expanded and optimized. Kuang *et al.* [25] proposed a copy placement strategy based on consistent hashing. This strategy is based on the HDFS copy placement strategy, which optimizes the server load and improves the access efficiency and performance of the system. Yang [26] proposed a dynamic copy storage scheme. The solution was based on a distributed file system, which improved its network access rate and storage efficiency. Zhou [27] proposed a small file-merging strategy based on an eigenvalue classification algorithm. This strategy is based on the problems caused by the large number of small files in the distributed file system; it improves the memory consumption of the system and the reading efficiency of the small files. Zhao *et al.* [28] proposed an improved block placement strategy to address some of the defects of the default HDFS.

However, none of the abovementioned studies has optimized the placement of replicas based on the temporal and spatial logical correlation characteristics of remote-sensing data; therefore, their proposed solutions are not directly suitable for the storage and management of large-scale and long-term remote-sensing data. Therefore, in this research, a spatiotemporal correlation strategy for remote-sensing data based on a consistent hashing algorithm to optimize the storage and placement of data and its copies in the distributed file system is constructed to improve the access efficiency of remote-sensing image files.

B. Spatiotemporal Organization and Indexing of Metadata

The spatiotemporal organization and indexing of remote-sensing metadata further introduces time information to form a complex data structure that contains spatiotemporal information. This mainly includes two types: a tree structure-based spatiotemporal index and a spatial grid-based spatiotemporal index.

A tree structure-based spatiotemporal index refers to a data structure that is arranged in the order of rectangle (R)-trees, quad-trees, K-dimensional trees, etc., according to the position and shape of spatial objects or a certain spatial relationship between them, and superimposes temporal attributes. Xu *et al.* [29] proposed an ST-OpenGIS spatio-temporal data model for geographic information system (GIS) data. The model maintains the spatiotemporal data structure and correlations; abstracts the spatiotemporal data out of space, time, attributes, and other information; uses the spatial Z-curve to reduce the dimensionality of the spatiotemporal data; and constructs a B+ tree to build a global distributed index. It has been verified by experiments that this method has high efficiency for querying spatio-temporal data, but the model cannot process spatial topology and temporal information at the same time. In summary, for the spatio-temporal index methods based on tree structures, although the spatial range query is efficient, the construction of a balanced tree structure requires a great deal of resources.

Spatial grid-based spatiotemporal indexes can use excellent spatial grid models, such as GeoHash and GeoSOT, to traverse spatial objects; they can effectively avoid the resource consumption problem caused by the construction of a balanced tree index. Li *et al.* [30] proposed a spatiotemporal indexing method based on improved GeoHash coding that used the ElasticSearch engine to store data to improve query processing efficiency. This method expands the spatial grid GeoHash code, combines time and the GeoHash, can support spatial and temporal queries of spatiotemporal data, and has better query performance than the traditional databases PostgreSQL and MySQL. Li *et al.* [31] proposed a spatiotemporal block index framework STB-HBase. Two HBase tables are constructed to achieve secondary indexes. Experiments showed that this method can effectively avoid storage hotspots of traffic data and support spatio-temporal range queries. However, it needs to build two tables, which are difficult to maintain, and the efficiency of multiple queries is low. Zhao *et al.* [32] built a two-layer spatiotemporal index GRIST, based on GeoHash and an R-tree. This method uses a two-layer index structure; the first layer uses spatial grid GeoHash coding, and the second layer uses an R-tree to process time information. The index construction of this method was greatly improved compared to the GeoMesa [33] and PostGIS [34] systems. However, it did not effectively combine time information and space, and the filtering of spatiotemporal data required a secondary index, which weakens the effect of the index to a certain extent. Jiang *et al.* [35] proposed a spatio-temporal hybrid index, based on the combination of Hilbert coding [36] and an R-Tree. Zhang *et al.* [37] proposed an HBase spatio-temporal index, which is implemented through a secondary index.

The spatiotemporal indexes constructed by the abovementioned methods can improve the spatiotemporal retrieval efficiency of remote-sensing data to a certain extent, but most of the algorithms use a simple spatiotemporal index superimposed on a secondary index to process the spatiotemporal data, and it needs to be filtered twice during a query, which reduces the indexing efficiency. Therefore, this study proposes a GeoSOT-ST-based data organization method, which converts high-dimensional time and space information into a one-dimensional GeoSOT-ST [38], code, thereby effectively reducing the amount of conditional filtering in the query process. For remote-sensing metadata, an HBase database was chosen as the storage medium. In view of the asymmetrical temporal and spatial distribution of remote-sensing data, the HBase region prepartition is constructed based on GeoSOT-ST coding in the process of metadata storage to achieve balanced storage, thereby improving the data retrieval efficiency. For image files, HDFS was selected as the storage medium. Spatio-temporal information hash coding was used to optimize the replica placement strategy and store the space-time related data in the same or adjacent data nodes, thereby improving the efficiency of the data access.

III. Method

The efficient organization model of large-scale and long time-series remote-sensing data in a cloud-computing environment mainly includes the following content.

1) *Deheterogeneity*. This addresses the problem of inconvenient access to multisource remote-sensing images caused by different storage formats. The data ingest application programming interface (API) of the DataCube is used to realize a unified storage format, as well as splitting large volume images into small tiles for convenient access. In view of the heterogeneity of the multisource remote-sensing data, the International Organization for Standardization (ISO) geographic information metadata standard is adopted to realize the unification of the metadata, and the GeoSOT-ST spatio-temporal grid is established to realize the unification of the spatial and temporal reference of the multisource remote-sensing tiles.
2) *GeoSOT-ST-based unified tile spatio-temporal identification construction in HBase*. This uses unified spatial identification, time identification, and sensor information to establish an asymmetrical spatial-temporal grid index under the HBase architecture to achieve the rapid retrieval of massive remote-sensing tiles.

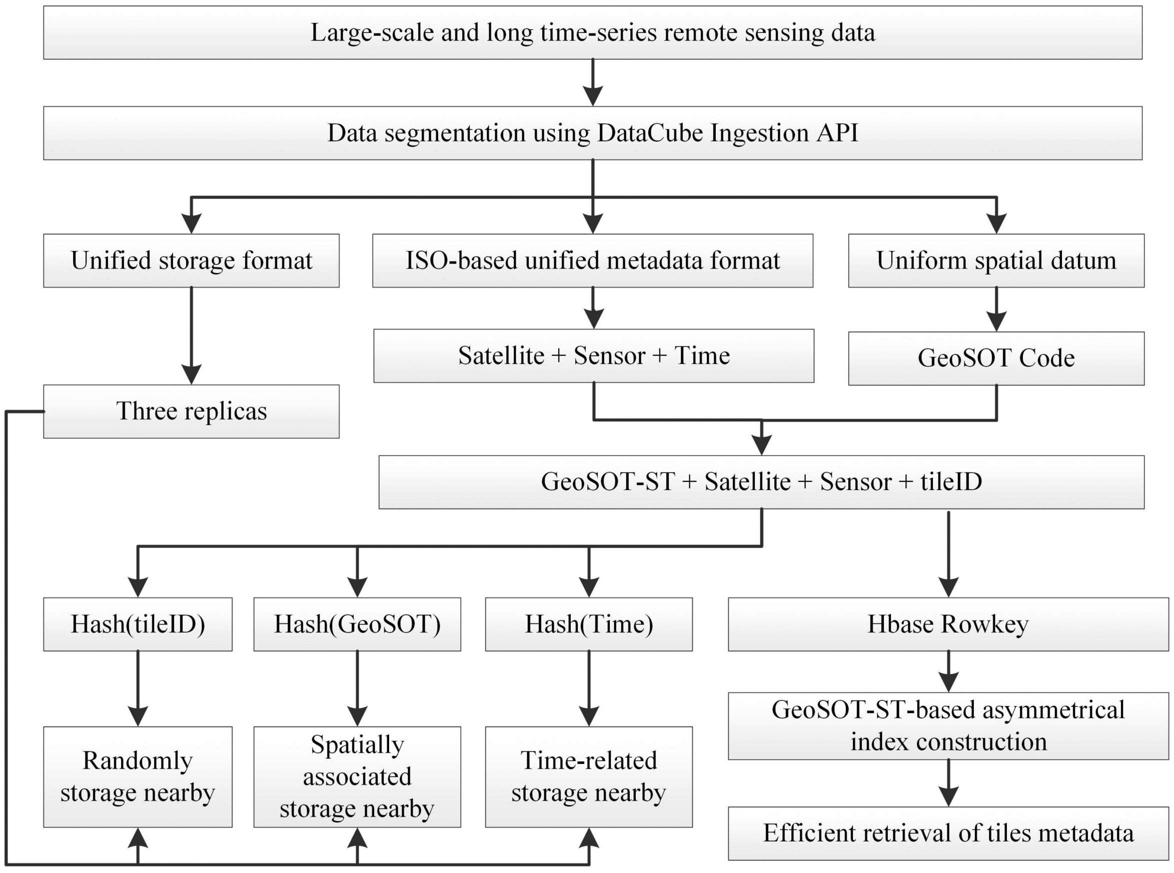

Large-scale and long time-series remote sensing data
Data segmentation using DataCube Ingestion API
Unified storage format
ISO-based unified metadata format
Uniform spatial datum
Satellite + Sensor + Time
GeoSOT Code
Three replicas
GeoSOT-ST + Satellite + Sensor + tileID
Hash(tileID)
Hash(GeoSOT)
Hash(Time)
Hbase Rowkey
GeoSOT-ST-based asymmetrical index construction
Randomly storage nearby
Spatially associated storage nearby
Time-related storage nearby
Efficient retrieval of tiles metadata

Fig. 1. Overall technical route.

3) *Distributed storage optimization of remote-sensing tiles based on consistent hash coding of spatio-temporal information*. This uses the asymmetrical spatio-temporal grid index of the tiles to construct a consistent hash code and establish a mapping relationship with distributed storage nodes based on it. In this way, the tiles that are logically related in time and space are stored in the same or nearby cloud storage nodes to achieve distributed and optimized storage of massive tiles, thereby improving the efficiency of the distributed data access.

The overall technical route is shown in Fig. 1.

A. Deheterogeneity

1) Standardization of Spatial Reference and Storage Format: The storage formats of multisource remote-sensing images are diverse, typically including GeoTIFF [39], hierarchical data format (HDF) [40], etc.; there are also spatial reference and projection differences, such as moderate-resolution imaging spectroradiometer (MODIS) series data that use sinusoidal projection [41], Landsat-series data based on the Worldwide Reference System-2, and universal horizontal-axis Mercator projections [42]. Remote-sensing images with different storage formats and spatial references are inconvenient to establish a unified data-access interface, which reduces the availability and scalability of the storage system. In addition, the volume of a single remote-sensing image is often very large. For example, the image size of a single scene Landsat _8 OLI_ TIRS image is about 900 MB. When a distributed file system is used for image storage, a single scene image will be automatically divided into several small blocks. The original spatial neighbor relationship between blocks is not considered, as they are arbitarily distributed to a data node. This will greatly reduce the efficiency of data retrieval related to the original image.

Therefore, this study uses the data ingest API of the DataCube [43] to physically partition the entire image. It then organizes each partitioned slice into the same file format and uses a unified metadata standard model to extract the slices' metadata (see Fig. 2). In this article, the actual volume of the divided tiles is less than 128 MB to meet the block storage requirements of the HDFS, and each divided tile is uniformly converted into the network common data form format for unified data access. In addition, each segmented tile is uniformly converted to the EPSG:4326 geographic coordinate system to facilitate a unified spatial query.

2) Unified Remote-Sensing Image Metadata Format: Multisource remote-sensing data are generated in different data centers, using independent metadata standards, such as the HDF-EOS metadata standard, which is not convenient for unified data organization, management, and retrieval. Therefore, this research refers to the ISO geographic information metadata standard in constructing a unified metadata model (see Table I); each

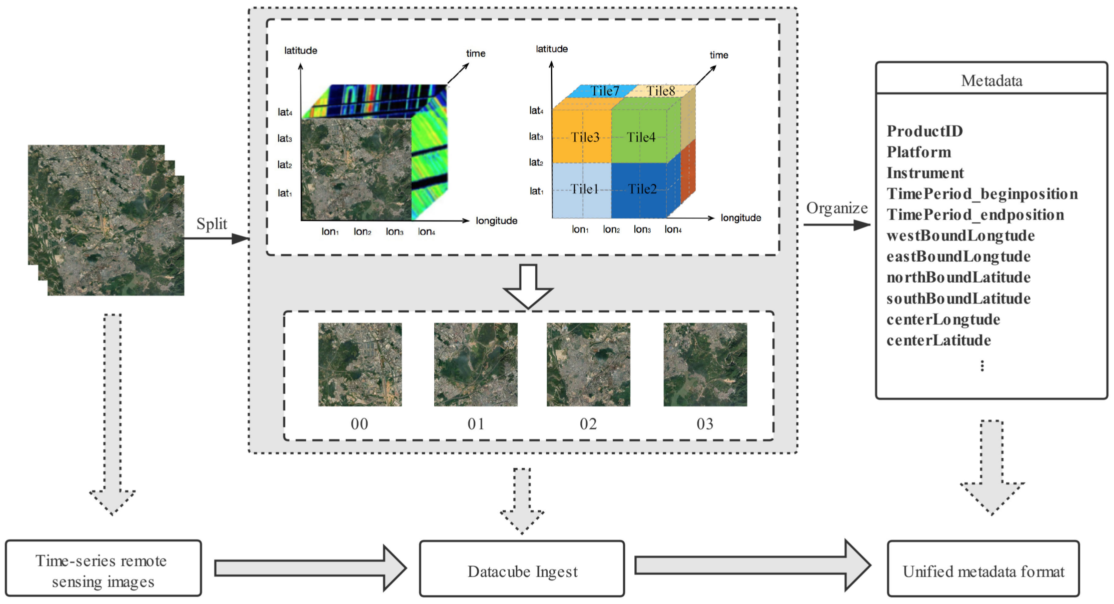


Fig. 2. Unified spatial reference and storage format.

TABLE I
UNIFIED REMOTE-SENSING IMAGE METADATA

No.	Category	Metadata Item	Metadata Description
1	Product	ProductID	imageID
2	Platform	Platform	Satellite
3	Sensor	Instrument	Sensor
4	TemporalExtent	TimePeriod_beginposition	Start Time
5	TemporalExtent	TimePeriod_endposition	End Time
6	SpatioExtent	westBoundLongtude	West Bound Longitude
7	SpatioExtent	eastBoundLongtude	East Bound Longitude
8	SpatioExtent	northBoundLatitude	North Bound Latitude
9	SpatioExtent	southBoundLatitude	South Bound Latitude
10	SpatioExtend	centerLongtude	Longitude of Center Point
11	SpatioExtend	centerLatitude	Latitude of Center Point
12	Product	ProductName	Product Name
13	Product	ProductLevel	ProductLevel
14	SpatialReference	Reference	Spatial Reference
15	Path	Path	Data Storage Path
16	Format	Format	Data Format
17	Cloud	CloudCoverPercentage	Cloud Cover Percentage
18	Band	Band	Band Information

segmented tile needs to perform metadata production according to the standard metadata model.

B. GeoSOT-ST-Based Unified Tile Spatio-Temporal Identification Construction

As typical spatio-temporal data, collecting the time, space range, satellite, and sensor information of remote-sensing data is necessary for its identification for later query and access. However, whether for data query in traditional structured query language (SQL) or NoSQL, multicondition joint retrieval is a major bottleneck that restricts query efficiency. Therefore, how to establish spatial identification based on the necessary satellite, sensor, time and space information and reduce the multicondition joint search to a single-condition search are the keys to improving the efficiency of massive data query.

The GeoSOT spatial-grid-coding model is a spatial index model based on space division. It takes the intersection of the prime meridian and the equator as the center point and transforms the earth into a plane through simple projection. The size of the plane is expanded from $180^\circ \times 360^\circ$ in the earth space to $512^\circ \times 512^\circ$, which will be the 0th layer of the spatial division. Through the quad-tree recursive subdivision of the earth's surface, different subsection levels are formed, and finally, the entire earth space is divided into multilevel grids of whole degrees, whole minutes, and whole seconds.

Each level grid unit of GeoSOT is coded according to the inverse Z-space-filling curve. The code is unique, that is, each grid unit has a unique code. This unique code corresponds to a rectangle that describes a geographic space of remote-sensing images. Moreover, the code uses a dimensionality reduction method to identify the two-dimensional spatial latitude and longitude area with a one-dimensional spatial code. Because the next-level grid is expanded on the basis of the last-level grid, their binary codes have the same prefix, which improves the spatial correlation of the remote-sensing data. When the image is segmented into small tiles, the spatial area of each segmented tile can be represented by the latitude and longitude of its center

TABLE II
SIZE OF A GEOSOT-ST SEGMENTATION BLOCK AT DIFFERENT LEVELS

Level	Grid Size	Time	Level	Grid Size	Time
1	256°	64 years	12	8'	16 days
2	128°	32 years	13	4'	8 days
3	64°	16 years	14	2'	4 days
4	32°	8 years	15	1'	2 days
5	16°	4 years	16	32"	1 day
6	8°	2 years	17	16"	16 h
7	4°	1 year	18	8"	8 h
8	2°	8 months	19	4"	4 h
9	1°	4 months	20	2"	2 h
10	32'	2 months	21	1"	1 h
11	16'	1 month			

point. Therefore, when calculating a tile's GeoSOT code, it is only necessary to calculate the GeoSOT code corresponding to the latitude and longitude of its center point to represent its spatial identification. Converting the two-dimensional spatial information of remote-sensing data into a one-dimensional GeoSOT code not only effectively improves the efficiency of the data retrieval but also can greatly improve the spatial correlation degree of the remote-sensing tiles.

GeoSOT-ST coding is based on spatial GeoSOT segmentation coding [44] but expands the time dimension. Its start time is January 1, 1970, and the time span is 128 years. A time between 0 and 64 years is coded as 0, and a time between 64 and 128 years is coded as 1; recursive subdivision is performed in turn to form different hierarchical levels. In the continuous recursive division process, when the division reaches one year, one year will be expanded into 16 months; when the division continues to month, each month will be expanded into 32 days. In this way, the time division code converts the time range from 1970 to 2098 into whole years, whole months, and whole days. See Table II for the size of a GeoSOT-ST segmentation block at different levels.

The calculation method of GeoSOT-ST code is as follows.

1) Determine the slice level. The default level is 21; that is, the space is divided into seconds, and the time is divided into hours. If the level of spatial calculation is less than 21, directly fill it with 1.
2) Because the area of each tile is very small, they can be treated as points during retrieval. Therefore, the GeoSOT grid code of the center point of each segmented tile is used as its spatial identification.
Assuming that the latitude and longitude of the center point of a certain tile is D ° M'S" and the collecting time is YMDH, S_n represents the spatial grid range at level *n*, and T_n represents the time range at level *n*. The coding calculation formulas for longitude, latitude, and time are provided by the following equations:

$$\text{Code}_{\text{Lon}} = \begin{cases} \frac{\text{Lon}_D}{S_n} & 1 \le n \le 9 \\ \frac{\text{Lon}_D * 64 + \text{Lon}_M}{S_n} & 9 < n \le 15 \\ \frac{\text{Lon}_D * 64 * 64 + \text{Lon}_M * 64 + \text{Lon}_S}{S_n} & 15 < n \le 21 \end{cases} \quad (1)$$

$$\text{Code}_{\text{Lat}} = \begin{cases} \frac{\text{Lat}_D}{S_n} & 1 \le n \le 9 \\ \frac{\text{Lat}_D * 64 + \text{Lat}_M}{S_n} & 9 < n \le 15 \\ \frac{\text{Lat}_D * 64 * 64 + \text{Lat}_M * 64 + \text{Lat}_S}{S_n} & 15 < n \le 21 \end{cases} \quad (2)$$

$$\text{Code}_{\text{Time}} = \begin{cases} \frac{Y}{T_n} & 1 \le n \le 6 \\ \frac{Y * 16 + M}{T_n} & 6 < n \le 10 \\ \frac{Y * 16 * 32 + M * 32 + D}{T_n} & 10 < n \le 15 . \\ \frac{Y * 16 * 32 * 32 + M * 32 * 32 + D * 32 + H}{T_n} & 15 < n \le 21 \end{cases} \quad (3)$$

In (1), Code_{Lon} represents the code of the longitude of the center point of the tile, Lon_D represents the integer degrees part of the longitude value, Lon_M represents the integer minutes part of the longitude value, and Lon_S represents the integer seconds part of the longitude value. Similarly, in (2), Code_{Lat} represents the code of the latitude of the center point of the tile, Lat_D represents the integer degrees part of the latitude value, Lon_M represents the integer minute part of the latitude value, and Lon_S represents the integer seconds part of the latitude value. In (3), $\text{Code}_{\text{Time}}$ represents the code of the collection time of the tile, Y represents the year value, M represents the month value, D represents the day value, and H represents the integer hours value.
3) Combine the latitude, longitude, and time codes into Morton code and convert every three digits into an octal code. The GeoSOT-ST code calculated in this way can then represent a range of time and space, which will be seen as the unified spatio-temporal identification of each segmented tile.

C. GeoSOT-ST-Based Asymmetrical Spatiotemporal Index Construction in HBase

1) GeoSOT-ST-Based HBase Table Schema Construction: After the unified spatio-temporal identification of a remote-sensing tile is constructed, it can be used as the RowKey of an HBase table to achieve key access. That is, the RowKey is composed of the GeoSOT-ST code, satellite and sensor code, and tile ID, with each field connected by a hyphen ("-"). This can be represented as follows:

$$\begin{aligned}\text{RowKey} = \text{GeoSOT} - \text{ST} + \text{Satellite and Sensor Code}\\ + \text{tileID}\end{aligned}$$

The GeoSOT-ST coding is calculated from the space-time information of the remote-sensing tiles according to the calculation method of GeoSOT-ST. The satellite and sensor code is obtained by combining the binary coding of the respective satellites and sensors. For example, the satellite and sensor code for the Landsat1 satellite and MSS sensor is 000000. The tileID is the unique identifier of the remote-sensing tile.

As for the time stamp of each row record, it is composed of the long code of the collection time of the remote-sensing image, including year, month, day, hour, minute, and second. In the metadata query process, the collection-time query condition can be filtered according to the timestamp.

2) GeoSOT-ST-Based HBase Asymmetrical Spatiotemporal Index Construction: While building the HBase table, the table will be prepartitioned into a number of regions at the same time. The partition rules of an HBase region are based on the prefix of the GeoSOT-ST code. When the GeoSOT-ST encoding prefix is $0, 1$, or 2, the number of regions is set as $1, 9$, and 65, respectively. Too many HBase regions will affect the performance of the system, and the number of partitions is related to the amount of data actually stored.

In addition, the partition of the HBase region is equivalent to the balanced division of the spatial regions. In other words, the GeoSOT-ST-based HBase region partition realizes the balanced storage and indexing of the metadata with a nonuniform temporal and spatial distribution. For example, if the HBase region is divided according to the spatial section level 1, then the number of divided regions is $8^1 + 1$. If the startKey of the HBase table is set to *m*, and the endKey is set to *M*, the range of the divided region partitions are, respectively, $< m, 0 >, < 0, 1 >, < 1, 2 >, < 2, 3 >, < 3, 4 >, < 4, 5 >,$ $< 5, 6 >, < 6, 7 >$, and $< 7, M >$. When data are inserted, the RowKey starting with 01 will hit the region $< 0, 1 >$.

However, by default, HBase arranges RowKeys in lexicographic order to form a large table and splits them into multiple regions in turn according to the number of rows. Due to the asymmetrical temporal and spatial distribution of remote-sensing data, the tile metadata associated with temporal and spatial logic will be scattered and stored in multiple regions if the HBase default partitioning mode is followed directly. The spatiotemporal query process involves the traversal of multiple regions, which greatly reduces the query efficiency. Therefore, on the premise of prepartitioning, the HBase coprocessor is used to further partition those regions that are still asymmetrically distributed on the server side. If the amount of data in a region is too small or not available, the region will be merged with other partitions; if the data volume of a certain region is too large, it will be counted first and will be further divided into many regions if it exceeds a certain threshold. After a series of splitting and merging operations, the index of remote-sensing tiles can be evenly distributed in different regions, and each region will form an ordered index tree. The specific process is shown in Fig. 3.

After the construction of the HBase table schema and asymmetrical index is completed, the massive remote-sensing tile metadata can be directly stored in the database. This study uses BulkLoad to write the data directly into the HFile file; its data writing method improves the efficiency of this operation. In this way, a large amount of multisource heterogeneous remote-sensing tile metadata can be successfully inserted into the HBase table. The data in the table will show a balanced distribution, and hot spots and data skew will not occur when accessing the same region or a small number of regions.

D. Distributed Storage Optimization of Remote-Sensing Tiles Based on Consistent Hash Coding of Spatiotemporal Information

Single-scene remote-sensing images are often large in volume. When HDFS is used for storage, large images will be automatically divided into multiple blocks and then randomly stored on different nodes.

However, the HDFS default storage strategy does not consider the spatio-temporal logical correlation characteristics of remote-sensing tiles. Remote-sensing tiles from the same or adjacent areas or time range may be allocated to different data nodes, thereby reducing the access performance of the associated data.

In order to effectively improve the access efficiency of massive tiles, this study optimizes the default HDFS storage strategy and proposes a distributed storage optimization strategy for remote-sensing data based on a consistent hash coding of spatiotemporal information. The basic idea of this strategy is to construct a consistent hash [45] code based on the unified time and space identification of remote-sensing tiles and establish a mapping relationship with distributed storage nodes based on the space-time hash code. In this way, the tiles that are logically related in time and space are stored in the same or adjacent cloud storage nodes to achieve the distributed and optimized storage of massive tiles, thereby improving the efficiency of distributed data access.

It should be noted that in view of the default three-replica storage mechanism of HDFS, this research needs to take into account the temporal and spatial characteristics of the remote-sensing data. The three factors of data access proximity, time relevance, and spatial relevance should be considered while formulating a three-replica storage strategy.

The specific three-replica storage strategy is as follows.

1) For the first replica, the default consideration is to store the data node nearby to where the write request is located. If the data node cannot be stored nearby, an available data node is arbitrarily selected on the same rack for storage.
2) The second replica storage considers time proximity; a consistent hash value based on the time code of the tile is constructed. It is mapped with the hash value of the data node, and it is stored nearby in a clockwise direction.
3) The third replica storage considers spatial proximity; a consistent hash value according to the spatial coding of the tile is constructed. It is mapped with the hash value

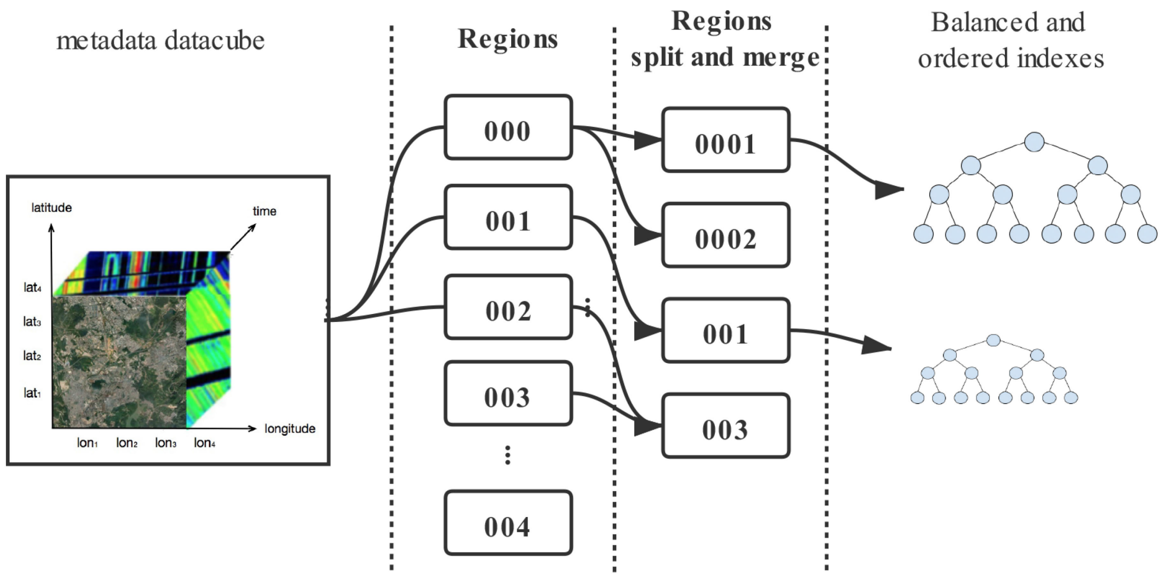


Fig. 3. HBase asymmetrical spatiotemporal index construction.

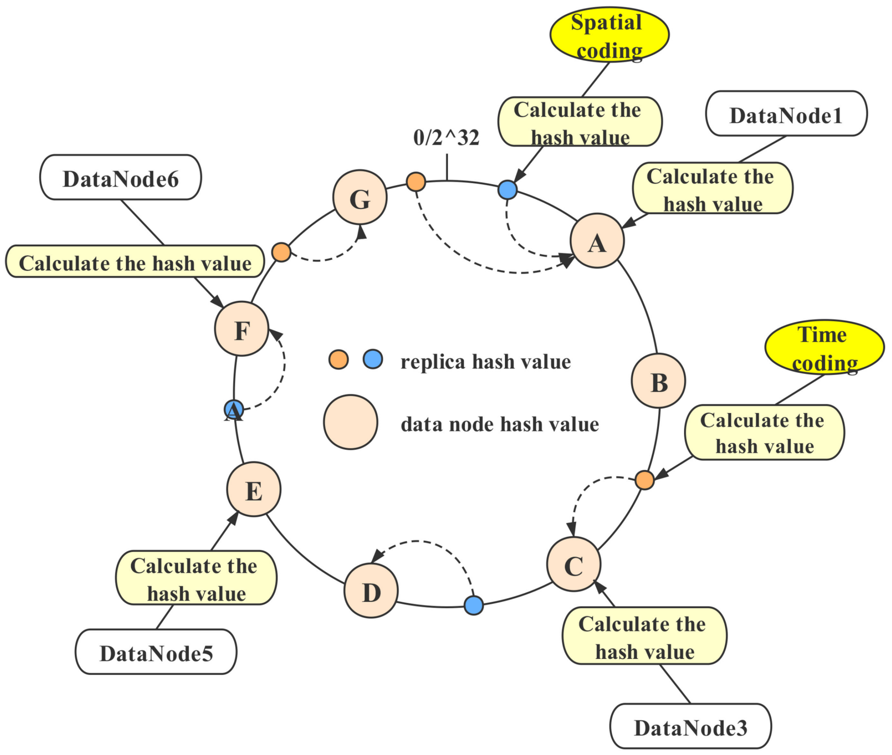


Fig. 4. Replica placement based on the consistent hash coding of the temporal and spatial information.

of the data node, and it is stored in the nearest available space in a clockwise direction.

What needs to be added is that the unique identification of each data node in the Hadoop cluster is a combination code of the host name and port, and the hash value of the data node is the consistent hash code of the identification.

The consistent hash mapping of each replica and data node is shown in Fig. 4.

IV. Experiments

In order to verify the effectiveness of the large-scale and long-term remote-sensing data organization strategy in a cloud-computing environment proposed in this study, relevant experiments on the massive remote-sensing metadata asymmetrical index under the HBase architecture and remote-sensing tile access under the HDFS architecture were carried out. The experimental environment was a Hadoop service cluster with five nodes. Each

TABLE III
REMOTE-SENSING IMAGE DATA DETAILS

Satellite	Sensor	Time Range	Data Amount
Landsat1	MSS	1972～1978	147,417
Landsat2	MSS	1975～1982	247,206
Landsat3	MSS	1978～1983	123,683
Landsat4	MSS	1982～1992	171,646
	TM	1982～1993	46,990
Landsat5	MSS	1984～2001	638,416
	TM	1984～2012	2,864,061
Landsat7	ETM	1999～2018	2,585,203
Landsat8	OLI	2016～2018	4,069
	TIRS	2013～2018	3,498
	OLI_TIRS	2013～2018	1,404,435
Total number of metadata records			8,236,624

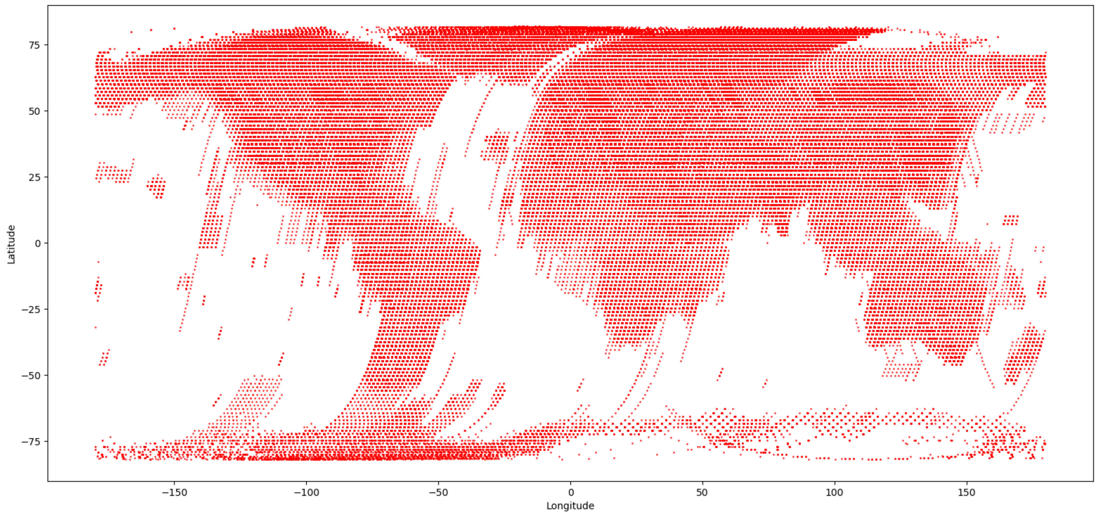


Fig. 5. Distribution map of the Landsat-series satellite data.

node had 64 GB of memory, 2 TB of disk, and a CentOS 7.5 operating system. The Hadoop version was 2.7.3, and the HBase version was 1.4.0.

A. GeoSOT-ST-Based Asymmetrical Spatiotemporal Index Experiments in HBase

1) Experimental Data: The GeoSOT-ST-based asymmetrical spatiotemporal index experiments in HBase used 8 236 624 Landsat remote-sensing metadata records provided by the U.S. Geological Survey as the experimental data (see Table III). According to the longitude and latitude of the center point of each image (see Fig. 5), it can be seen that the data have obvious asymmetrical temporal and spatial distribution characteristics.

2) HBase-Based Asymmetrical Spatio-Temporal Index Experiments: In order to test the efficiency of the remote-sensing metadata retrieval, a series of spatial range query experiments were carried out on the remote-sensing metadata stored in the HBase table. The query conditions of the satellite and sensor were set to Landsat5 and TM. The time range query was set to the entire range; that is, the filtering conditions were from January 1, 1970 to December 31, 2020. In terms of spatial query conditions, they were created by randomly selecting points in the center of the world map and then constructing a query area of 1/2, 1/4, 1/8, 1/16, and 1/32 of the global space (GS). In summary, Landsat5, TM, and 1970/01/01 to 2020/12/31 were used as the fixed satellite, sensor, and time query conditions, and the spatial query conditions were changed to GS/2, GS/4, GS/8, GS/16, and GS/32. Five experiments were carried out to evaluate the effectiveness of the proposed model in terms of the time for a query to produce results, defined here as the time consumption. It should be added that in this experiment, due to the large amount of metadata, the prepartition level of the HBase region was set to level 2. Therefore, we named our solution for this experiment GeoSOT-ST2.

In addition, in order to compare and verify the effectiveness of the GeoSOT-ST2 method proposed in this study, comparison experiments were carried out with a traditional “latitude + longitude + time” query, “GeoSOT + time” query, and the GeoSOT-ST query. What needs further explanation is that

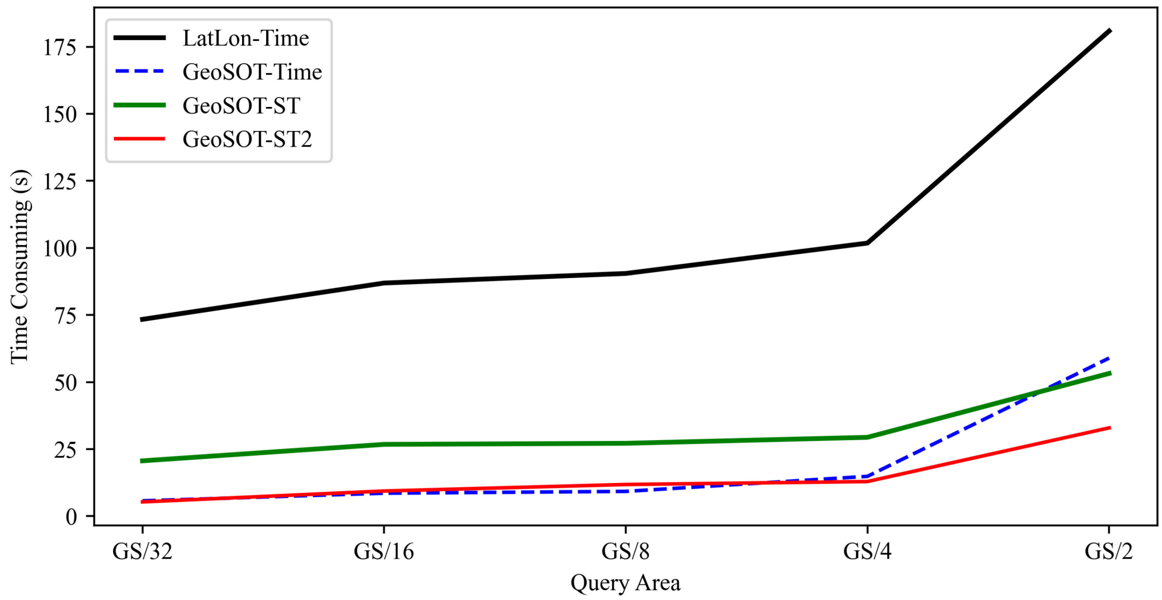


Fig. 6. Time consumption of the four data organization methods under different spatial query conditions.

the GeoSOT-ST experimental program only used the GeoSOT-ST grid code to generate the HBase RowKey, and the prepartitioning of the HBase region was not performed during the table creation process.

The time consumption of the four data organization methods under different spatial query conditions is shown in Fig. 6.

It can be seen from Fig. 6 that under the same query conditions, the GeoSOT-ST2 data organization model, in which the prepartition level of the HBase region is set to level 2, had the highest data retrieval efficiency; when the spatial query range is less than GS/4, the query efficiency of the GeoSOT-Time solution is always better than that of the GeoSOT-ST data organization method without HBase region prepartitioning; however, when the spatial query range is greater than GS/4, the query efficiency of the GeoSOT-Time solution drops sharply, while the query efficiency of the GeoSOT-ST method does not drop much; and when the spatial query range is close to GS/2, the query efficiency of the GeoSOT-Time method is already lower than that of the GeoSOT-ST; and the worst was the traditional "Latitude + Longitude + Time" data organization method.

The possible reasons are as follows.

1) The "Latitude + Longitude + Time" method required three conditions to filter, and, thus, the query efficiency was low.
2) For the GeoSOT-Time method, the data retrieval process is mainly realized by the RowKey and time field filtering joint retrieval. The RowKey is constructed by GeoSOT coding, with shorter length. When the spatial query range is relatively small, the GeoSOT-Time method will first use the RowKey constructed by GeoSOT encoding to lock a smaller range of rows, and then only needs to perform time field filtering on the selected rows, which can greatly improve the retrieval efficiency.
3) For the GeoSOT-ST method, the data retrieval process is mainly realized by the RowKey constructed by GeoSOT-ST coding. The composition sequence of the RowKey coding is "GeoSOT-based spatial coding + time coding + satellite and sensor coding + tileID," with longer length. Since the time query range is the entire time range, even when the query space range is small, the GeoSOT-ST method still needs to traverse all RowKeys in lexicographic order, so the GeoSOT-ST query efficiency is lower than the GeoSOT-Time method. With the increase of the query space range, the number of rows that GeoSOT-Time needs to filter twice increases, and the number of RowKeys traversed by the GeoSOT-ST method is not much different, so the query efficiency will be higher than that of the GeoSOT-Time method.
4) Due to the prepartition level of the HBase region is set to level 2, the GeoSOT-ST2 data organization model effectively realized the uniform distribution of the massive remote-sensing metadata in the HBase region. During data retrieval, the asymmetrical spatiotemporal distribution of remote-sensing metadata will not cause hotspot access problems, so the data query efficiency of the GeoSOT-ST2 method is higher than that of GeoSOT-ST, in which the HBase region prepartition is not performed.

B. Remote-Sensing Tile Storage Optimization Experiment

In the remote-sensing tile storage optimization experiment, MODIS data were selected from 2000 to 2010, and the original storage format was HDF. The DataCube data ingest API was called for the physical segmentation to ensure that the size of each slice was less than 128 MB and that the storage in the HDFS was not diced, so as not to affect the time and space correlation of the tiles.

1) Spatiotemporal Relevance Analysis of the Tiles in HDFS: The segmented tiles were uploaded to the HDFS and were distributed to multiple data nodes in an HDFS cluster. By

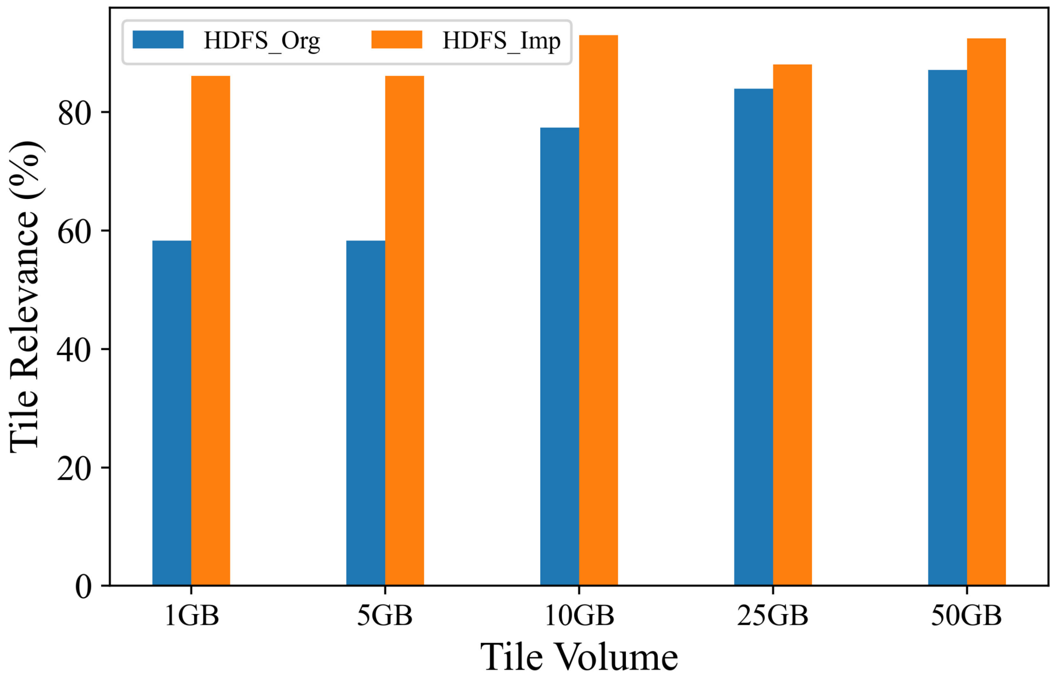


Fig. 7. Correlation degree of the tiles in HDFS varied with the volume of the file.

reading the tile metadata of each data node and calculating the time and space information of all the tiles in the node, the spatiotemporal correlation degree of all the tiles in the node could be calculated. If the temporal and spatial correlation degree of tiles stored in a data node was higher, the probability of them being accessed at the same time would be greater, thereby avoiding a large number of disk searches and greatly improving the efficiency of a data query. Measuring the temporal and spatial correlation of the storage tiles of each data node can effectively verify the effectiveness of the data organization scheme based on GeoSOT-ST spatiotemporal grid coding proposed in this study.

Assuming that the spatiotemporal association rate is $\mathrm{ST}_{\mathrm{AssociationRate}}$, the number of spatiotemporal weak-association data is wr, the total number of files is F, the number of replicas is N, and the number of data nodes is C, then the calculation formula for the spatiotemporal association rate is as follows:

$$\mathrm{ST}_{\text{Association Rate}} = \left(1 - \frac{\mathrm{wr} \times C}{F \times N}\right) \times 100\%. \quad (4)$$

What needs to be added is that calculating the number wr of files with weak spatiotemporal associations on a node mainly depends on the spatiotemporal identification of each tile. If the number of spatiotemporal identifiers with the same code is less, the degree of spatiotemporal correlation is considered to be weaker. In this experiment, if the spatiotemporal identification codes of the two tiles are not the same in 4 or more bits, the two tiles will be considered to be weak spatiotemporal associations. In this article, the calculated spatiotemporal association rate $\mathrm{ST}_{\mathrm{AssociationRate}}$ will be used as an evaluation index to measure the spatio-temporal correlation of the tiles in the HDFS. A total of 50 GB was cut from the MODIS data, and it was then uploaded to the HDFS; this was repeated five times. After each upload, the HDFS data volume was 1 GB, 5 GB, 10 GB, 25 GB, and 50 GB, respectively, and the spatiotemporal correlation of all the HDFS tiles was calculated. In order to effectively illustrate the effectiveness of the GeoSOT-ST-based tile distributed-storage optimization scheme proposed in this study, it was compared with the traditional HDFS default replica placement scheme. The results of comparing the spatio-temporal logic correlations are shown in Fig. 7. The GeoSOT-ST-based replica placement strategy proposed in this article is an improvement to the native HDFS system, so the solution proposed in this study was named HDFS_Imp and the native HDFS replica placement scheme was named HDFS_Org.

It can be seen from Fig. 7 that the improved HDFS replica placement strategy always made the spatiotemporal correlation of files higher than the original HDFS. When the data volume was small, the file association rate of the improved HDFS replica placement strategy was much higher than that of the original HDFS; when the data volume gradually increased, the file association rate of the two strategies gradually approached each other. This may be due to the fact that the experimental HDFS cluster had too few nodes. When the amount of tiles is too large, the temporal and spatial correlation between the tiles will increase. If the number of cluster nodes of the server increases, the data will be evenly distributed to multiple nodes, and the improved HDFS replica placement strategy will be greatly improved than that of the original strategy.

2) Tile Reading and Writing Experiments: The image tile reading and writing experiments were performed on the improved HDFS system, and its efficiency was compared that of the native HDFS system. For the data-writing experiment, 1 GB, 5 GB, 10 GB, 25 GB, and 50 GB data upload trials were carried

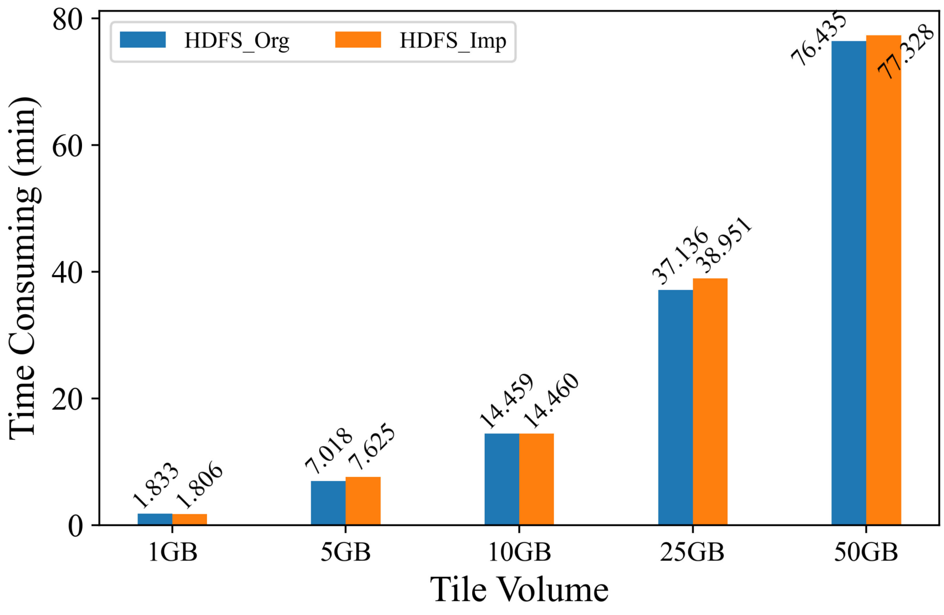


Fig. 8. Comparison chart of file-writing efficiency in HDFS.

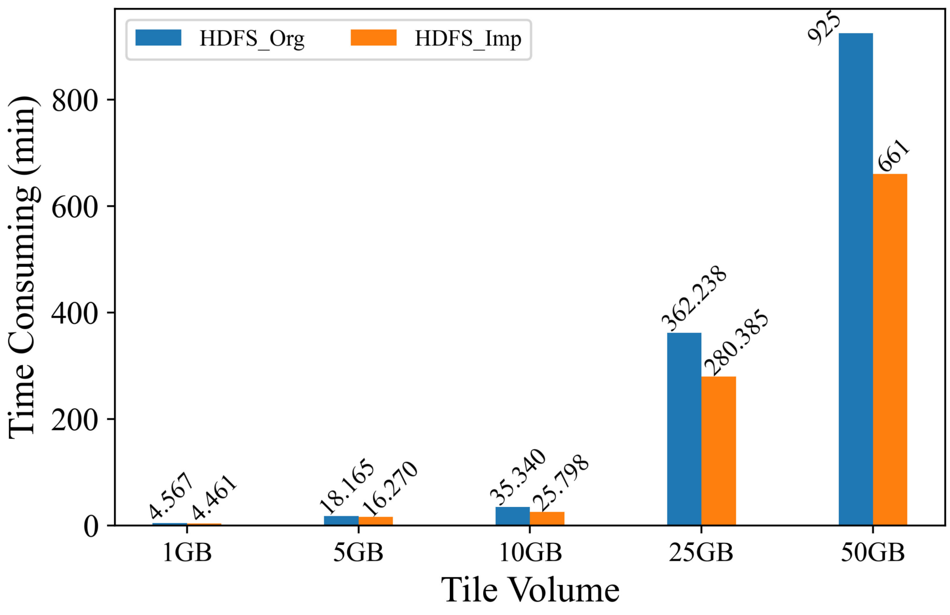


Fig. 9. Comparison chart of file-reading efficiency in HDFS.

out on both the improved HDFS system and the native HDFS system, and their respective upload times were recorded. The final experimental results are shown in Fig. 8.

As can be seen from Fig. 8, when writing files, the efficiency of HDFS_Imp was not much different from that of HDFS_Org, and even its writing efficiency was slightly lower than that of HDFS_Org. The reason is that the replica storage strategy based on the consistent hashing algorithm proposed in this article needs to calculate the hash value of the time and space information when selecting the data replica. It also needs to extract the attribute information from all the data nodes, and, finally, hash value mapping is needed to complete the copy placement. All of this processing is very time-consuming. Therefore, the strategy proposed in this article was lower than the default HDFS copy storage strategy in terms of data storage efficiency. However, given the "write once, read many" operating mode of the big data framework, a slight write performance degradation can be tolerated.

For the data-reading experiment, 1 GB, 5 GB, 10 GB, 25 GB, and 50 GB of data uploaded to the HDFS system was read, and the time taken was recorded. The final experimental results are shown in Fig. 9.

As can be seen from Fig. 9, the file-read efficiency of HDFS_Imp was always higher than that of HDFS_Org. When the amount of data was small, the file reading efficiency of the two methods was not much different. However, when the amount of data was large, the improved HDFS strategy proposed in this article greatly improved the efficiency of the data reading. Specifically, compared with the original HDFS storage strategy, the improved HDFS strategy in this article improved the data access efficiency by about 40%. With the gradual increase in the amount of access data, the efficiency improvement was increasingly obvious, which strongly demonstrates the superiority of the replica placement strategy based on the GeoSOT-ST space-time hash-coding proposed in this research.

V. Conclusion

The high-efficiency organization model of large-scale and long-term remote-sensing data in a cloud-computing environment proposed in this study eliminates the metadata differences between different satellite sensors by constructing a unified metadata field, eliminates the differences in the spatial reference and storage format of multisource remote-sensing data through DataCube physical segmentation and format conversion, constructs a GeoSOT-ST-based asymmetrical spatio-temporal index to optimize the metadata storage strategy in HBase, and realize a distributed storage optimization of the remote-sensing tiles based on a consistent hash coding of the spatiotemporal information. Through the massive metadata retrieval experiment in HBase and the remote-sensing tile reading and writing experiments in HDFS, it was demonstrated that the proposed data organization method can effectively improve the retrieval and access efficiency of large-scale and long-term remote-sensing data.

However, in order to maintain the consistency of the block size in HDFS, this study uniformly divided the multisource remote-sensing data into 128 MB tiles. However, as the types of multisource remote-sensing data gradually increase, the spatial resolution between different types of remote-sensing data will be quite different. For example, the spatial resolution of MODIS is 1 km, while the spatial resolution of the GaoFen-series satellite data is 1 m or even sub-meter. In this way, the 128 MB tile segmentation mode was uniformly adopted, and the space covered by the segmented tiles will vary greatly. Using the space-time coding information of the center point proposed in this study to construct the tile identification will cause the reduction of data query accuracy. Therefore, how to set an appropriate slice size according to the difference in the spatial resolution of the remote-sensing data or how to establish an organization strategy for remote-sensing data with different spatial resolutions will be the focus of future work. In addition, this study is mainly designed for the efficient organization of long time-series remote sensing data under the Hadoop ecosystem, and it slightly reduces the writing efficiency in order to obtain efficient data acquisition. If you want to use it for the organization and management of vector data, such as trajectory data, road network data, etc., or apply the design to other distributed file system or distributed database, such as Ceph, Canssandra, etc., or greatly improve the efficiency of data reading and writing at the same time, still needs further improvement and optimization.

References

[1] M. C. Hansen *et al.*, “High-resolution global maps of 21st-century forest cover change,” *Science*, vol. 342, no. 6160, pp. 850–853, 2013.

[2] X. Liu *et al.*, “Global urban expansion offsets climate-driven increases in terrestrial net primary productivity,” *Nature Commun.*, vol. 10, no. 1, pp. 1–8, 2019.

[3] G. Zhang, Y. Shi, and M. Xu, “Evaluation of LJ1-01 nighttime light imagery for estimating monthly PM 2.5 concentration: A comparison with NPP-VIIRS nighttime light data,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 3618–3632, Jun. 2020.

[4] A. Lewis *et al.*, “The Australian geoscience data cube-foundations and lessons learned,” *Remote Sens. Environ.*, vol. 202, pp. 276–292, 2017.

[5] X. Zhang, L. Liu, X. Chen, S. Xie, and Y. Gao, “Fine land-cover mapping in China using landsat datacube and an operational speclib-based approach,” *Remote Sens.*, vol. 11, no. 9, pp. 1056–1073, 2019.

[6] H. Ostadabbas, H. Weippert, and F.-J. Behr, “Using the synergy of QField for collecting data on-site and QGIS for interactive map creation by ALKIS data extraction and implementation in postgreSQL for urban planning processes,” *Int. Archives Photogrammetry Remote Sens. Spatial Inf. Sci.*, vol. 43, pp. 679–683, 2020.

[7] N. H. Quang *et al.*, “Synthetic aperture radar and optical remote sensing image fusion for flood monitoring in the Vietnam lower Mekong basin: A prototype application for the Vietnam open data cube,” *Eur. J. Remote Sens.*, vol. 52, no. 1, pp. 599–612, 2019.

[8] G. Giuliani *et al.*, “Building an earth observations data cube: Lessons learned from the swiss data cube (SDC) on generating analysis ready data (ARD),” *Big Earth Data*, vol. 1, no. 1/2, pp. 100–117, 2017.

[9] D. Youssefi *et al.*, “Cars: A photogrammetry pipeline using Dask graphs to construct a global 3D model,” in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2020, pp. 453–456.

[10] J. Liu, Y. Xue, K. Ren, J. Song, C. Windmill, and P. Merritt, “High-performance time-series quantitative retrieval from satellite images on a GPU cluster,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 8, pp. 2810–2821, Aug. 2019.

[11] L. Zhou, N. Chen, Z. Chen, and C. Xing, “Roscc: An efficient remote sensing observation-sharing method based on cloud computing for soil moisture mapping in precision agriculture,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 12, pp. 5588–5598, Dec. 2016.

[12] Z. Wu, Y. Li, A. Plaza, J. Li, F. Xiao, and Z. Wei, “Parallel and distributed dimensionality reduction of hyperspectral data on cloud computing architectures,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 6, pp. 2270–2278, Jun. 2016.

[13] G. Cheng, X. Xie, J. Han, L. Guo, and G.-S. Xia, “Remote sensing image scene classification meets deep learning: Challenges, methods, benchmarks, and opportunities,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 3735–3756, Jun. 2020.

[14] G. Cheng, C. Yang, X. Yao, L. Guo, and J. Han, “When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative CNNs,” *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 5, pp. 2811–2821, May 2018.

[15] J. Fan, J. Yan, Y. Ma, and L. Wang, “Big data integration in remote sensing across a distributed metadata-based spatial infrastructure,” *Remote Sens.*, vol. 10, no. 1, pp. 7–26, 2018.

[16] K. McKusick and S. Quinlan, “GFS: Evolution on fast-forward,” *Commun. ACM*, vol. 53, no. 3, pp. 42–49, 2010.

[17] L. Li, W. Jing, and N. Wang, “An improved distributed storage model of remote sensing images based on the HDFS and pyramid structure,” *Int. J. Comput. Appl. Technol.*, vol. 59, no. 2, pp. 142–151, 2019.

[18] Y. Ma *et al.*, “Remote sensing Big Data computing: Challenges and opportunities,” *Future Gener. Comput. Syst.*, vol. 51, pp. 47–60, 2015.

[19] D. Emmanouil and D. Nikolaos, “Big data analytics in prevention, preparedness, response and recovery in crisis and disaster management,” in *Proc. 18th Int. Conf. Circuits, Syst., Commun. Comput.*, 2015, vol. 32, pp. 476–482.

[20] S. Wang, G. Li, X. Yao, Y. Zeng, L. Pang, and L. Zhang, “A distributed storage and access approach for massive remote sensing data in MongoDB,” *ISPRS Int. J. Geo- Inf.*, vol. 8, no. 12, pp. 533–548, 2019.

[21] A. N. Vinogradov, E. P. Kurshev, and S. Belov, "Open source file system selection for remote sensing data operational storage and processing," in *Proc. Int. Conf. Cyber-Phys. Syst. Control*, 2019, pp. 290–304.

[22] X. Wang, H. Zhang, J. Zhao, Q. Lin, Y. Zhou, and J. Li, "An interactive web-based analysis framework for remote sensing cloud computing." *ISPRS Ann. Photogrammetry*, *Remote Sens. Spatial Inf. Sci.*, vol. 2, pp. 43–50, 2015.

[23] Y. Luo, S. Luo, J. Guan, and S. Zhou, "A RAMCloud storage system based on HDFS: Architecture, implementation and evaluation," *J. Syst. Softw.*, vol. 86, no. 3, pp. 744–750, 2013.

[24] Q. Qin, Y. Wng, and M. Huang, "Storage of massive remote sensing image data based on MongoDB," *J. Beijing Univ. Civil Eng. Architecture*, vol. 31, no. 1, pp. 62–66, 2015.

[25] S. Kuang, "Research on replication strategy of distributed file system in cloud storage," Master's thesis, School Inf. Softw. Eng., Univ. Electron. Sci. Technol. China, Chengdu, China, 2013.

[26] G. Yang, "Research of cloud storage system optimization based on HDFS," Master's thesis, School Comput. Sci. Softw., Hebei University of Technol., Hongqiao District, China, 2016.

[27] M. Zhou, "Research and optimization of distributed storage based on HDFS," Master's thesis, School Comput. Inf. Sci., Hefei Univ. Technol., Hefei, Anhui, China, 2017.

[28] W. Zhao, L. Meng, J. Sun, Y. Ding, H. Zhao, and L. Wang, "An improved data placement strategy in a heterogeneous hadoop cluster," *Open Cybern. Systemics J.*, vol. 8, no. 1, pp. 957–963, 2014.

[29] Q. Xu, "Research and implement of massive GiS spatio-temporal data storage method in cloud environment," Master's thesis, School Comput. Sci., Xidian Univ., Xidian, China, 2018.

[30] F. Li, Y. Song, G. Wei, and Y. Wang, "Research on massive remote sensing data retrieval technology based on elasticsearch," *Comput. Netw.*, vol. 47, no. 5, pp. 57–61, 2021.

[31] P. Li and H. Jia, "Spatio-temporal block index for traffic data based on Hbase," *Inf. Technol.*, vol. 12, pp. 116–120, 2019.

[32] X. Zhao, X. Huang, J. Qiao, R. Kang, N. Li, and J. Wang, "A spatio-temporal index based on skew spatial coding and r-tree," *J. Comput. Res. Develop.*, vol. 56, no. 3, pp. 666–676, 2019.

[33] J. N. Hughes, A. Annex, C. N. Eichelberger, A. Fox, A. Hulbert, and M. Ronquest, "Geomesa: A distributed architecture for spatio-temporal fusion," *Proc. SPIE*, vol. 9473, 2015, Art. no. 94730F.

[34] K.-W. Lee and S.-G. Kang, "Open source remote sensing of ORFEO toolbox and its connection to database of PostGIS with NIX file importing," *Korean J. Remote Sens.*, vol. 26, no. 3, pp. 361–371, 2010.

[35] S. Jiang, "Spatial data organization and management of ice remote sensing data," Master's thesis, Beijing Univ. Civil Eng. Architecture: Yuan, Xicheng District, China, 2019, pp. 23–35.

[36] B. Cao, H. Feng, J. Liang, and X. Li, "Hilbert curve and cassandra based indexing and storing approach for large-scale spatiotemporal data," *Geomatics Inf. Sci. Wuhan Univ.*, vol. 46, no. 5, pp. 620–629, 2021.

[37] C. Zhang, X. Chen, Z. Shi, and B. Ge, "Algorithms for spatio-temporal queries in Hbase," *J. Chin. Comput. Syst.*, vol. 37, no. 11, pp. 2409–2415, 2016.

[38] C. Qian, C. Yi, C. Cheng, G. Pu, X. Wei, and H. Zhang, "Geosot-based spatiotemporal index of massive trajectory data," *ISPRS Int. J. Geo- Inf.*, vol. 8, no. 6, pp. 284–295, 2019.

[39] N. Ritter and M. Ruth, "The GeoTiff data interchange standard for raster geographic images," *Int. J. Remote Sens.*, vol. 18, no. 7, pp. 1637–1647, 1997.

[40] M. Schramm *et al.*, "The openeo API-harmonising the use of Earth observation cloud services using virtual data cube functionalities," *Remote Sens.*, vol. 13, no. 6, pp. 1125–1145, 2021.

[41] E. H. Bair, T. Stillinger, and J. Dozier, "Snow property inversion from remote sensing (spires): A generalized multispectral unmixing approach with examples from MODIS and Landsat 8 oli," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 9, pp. 7270–7284, Sep. 2021.

[42] J. Liu, R. Liu, K. Ren, X. Li, J. Xiang, and S. Qiu, "High-performance object detection for optical remote sensing images with lightweight convolutional neural networks," in *Proc. IEEE 22nd Int. Conf. High Perform. Comput. Commun.; IEEE 18th Int. Conf. Smart City; IEEE 6th Int. Conf. Data Sci. Syst.*, 2020, pp. 585–592.

[43] A. Lewis *et al.*, "The Australian geoscience data cubefoundations and lessons learned)," *Remote Sens. Environ.*, vol. 202, pp. 276–292, 2017.

[44] T. Qu *et al.*, "STGI: A spatio-temporal grid index model for marine Big Data," *Big Earth Data*, vol. 4, no. 4, pp. 435–450, 2020.

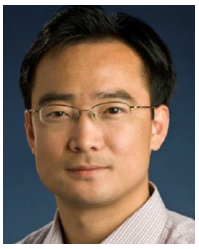
[45] Y. Li, Y. Zhang, X. Huang, H. Zhu, and J. Ma, "Large-scale remote sensing image retrieval by deep hashing neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 2, pp. 950–965, Feb. 2018.

Jining Yan received the Ph.D. degree in signal and information processing from the University of Chinese Academy of Sciences, Beijing, China, in 2017.

He is currently an Associate Professor of School of Computer Science, China University of Geosciences, Wuhan, China. His research is focused on remote sensing data management and time-series analysis.

Yuanxing Liu received the B.E. degree in Computer Science and Technology from China University of Geosciences, Wuhan in 1997. He is currently working toward the Ph.D. degree in geoscience information engineering.

His research interests include big data management and visualization.



Lizhe Wang (Fellow, IEEE) received the B.E. and M.E in electrical engineering and automation from Tsinghua Univ. in 1998 and 2001, and the D.E. in applied computer science from Univ. Karlsruhe, Germany, in 2007.

He is the Dean of the School of Computer Science, China University of Geosciences, Wuhan, China. His research interests include remote sensing data processing, digital earth, big data computing.

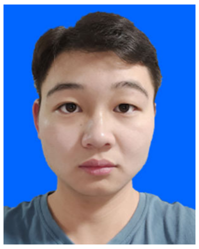
Mr. Wang is a fellow of IET and BCS, and an Associate Editor for *Remote Sensing* and *International Journal of Design Engineering*.

Zhipeng Wang received the master's degree in computer technology from the China University of Geosciences, Wuhan, China, in 2021.

He is a Graduate Student with the School of Computer Science, China University of Geosciences. His research is focused on remote sensing data storage, organization, and management.

Xiaohui Huang received the B.E. degree in network engineering from China University of Geosciences, Wuhan in 2015. He is currently working toward the Ph.D. degree in geoscience information engineering.

His research interests include geoscience big data management.



Hong Liu received the B.E. degree in software engineering from North China University of Water Resources and Electric Power in 2020. He is currently working toward the M.E. degree in electronic information.

His research is focused on spatial-temporal big data organization index and management analysis.