TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 01/10 pp881–893
DOI: 10.26599/TST.2021.9010082
Volume 27, Number 6, December 2022

MIX-RS: A Multi-Indexing System Based on HDFS for Remote Sensing Data Storage

Jiashu Wu, Jingpan Xiong, Hao Dai, Yang Wang*, and Chengzhong Xu

Abstract: A large volume of Remote Sensing (RS) data has been generated with the deployment of satellite technologies. The data facilitate research in ecological monitoring, land management and desertification, etc. The characteristics of RS data (e.g., enormous volume, large single-file size, and demanding requirement of fault tolerance) make the Hadoop Distributed File System (HDFS) an ideal choice for RS data storage as it is efficient, scalable, and equipped with a data replication mechanism for failure resilience. To use RS data, one of the most important techniques is geospatial indexing. However, the large data volume makes it time-consuming to efficiently construct and leverage. Considering that most modern geospatial data centres are equipped with HDFS-based big data processing infrastructures, deploying multiple geospatial indices becomes natural to optimise the efficacy. Moreover, because of the reliability introduced by high-quality hardware and the infrequently modified property of the RS data, the use of multi-indexing will not cause large overhead. Therefore, we design a framework called Multi-IndeXing-RS (MIX-RS) that unifies the multi-indexing mechanism on top of the HDFS with data replication enabled for both fault tolerance and geospatial indexing efficiency. Given the fault tolerance provided by the HDFS, RS data are structurally stored inside for faster geospatial indexing. Additionally, multi-indexing enhances efficiency. The proposed technique naturally sits on top of the HDFS to form a holistic framework without incurring severe overhead or sophisticated system implementation efforts. The MIX-RS framework is implemented and evaluated using real remote sensing data provided by the Chinese Academy of Sciences, demonstrating excellent geospatial indexing performance.

Key words: Remote Sensing (RS) data; geospatial indexing; multi-indexing mechanism; Hadoop Distributed File System (HDFS); Multi-IndeXing-RS (MIX-RS)

1 Introduction

Advanced satellites from several space agencies[1–3] are constantly orbiting the planet, generating a massive amount of Remote Sensing (RS) data[4–7]. The size of RS images is typically large as they are captured by top-tier camera devices with multiple bands or layers. As the RS data are produced daily[8] in the era of big data[9, 10], the order of magnitude of the data grows to terabyte[11] or even petabyte[12]. As a result, the Hadoop Distributed File System (HDFS) is often utilised as storage for the RS data[13–16]. Deploying HDFS as the RS data storage can not only bring efficiency and scalability as the amount of data increases but also provide fault tolerance because of the data replication equipped by the

• Jiashu Wu, Jingpan Xiong, and Hao Dai are with Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China, and the University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: {js.wu, jp.xiong, hao.dai}@siat.ac.cn.
• Yang Wang is with Guangdong-HongKong-Macao Joint Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China. E-mail: yang.wang1@siat.ac.cn.
• Chengzhong Xu is with the Faculty of Science and Technology, University of Macau, Macau 999078, China, and Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China. E-mail: czxu@um.edu.mo.
∗ To whom correspondence should be addressed.

Manuscript received: 2021-05-24; revised: 2021-08-26; accepted: 2021-10-06

HDFS[17].

However, the HDFS does not address the inefficiency of the geospatial indexing caused by the ever-growing data volume. Constant research efforts have been directed to optimise the efficacy of geospatial indexing and the utilisation of RS data[18–21]. However, existing frameworks are inefficient because they use Hadoop MapReduce[18, 20, 21], which is time-consuming at start-up[22], requires sophisticated effort to implement and change, and does not attempt to use multi-indexing to optimise the performance.

Given that the RS data are stored on HDFS with data replication enabled, we consider it is natural to deploy a multi-indexing mechanism on top of the HDFS to form a unified framework for better geospatial indexing efficiency. The rationales of using the multi-indexing mechanism are as follows:

- RS data are stored on HDFS with data replication enabled, therefore, unifying the multi-indexing mechanism on top of the data replication is natural and feasible. The storage nodes can also parallelise the index construction and querying, boosting the indexing performance.
- Using multiple light-weighted geospatial indexing algorithms makes the multi-indexing mechanism more resilient to single-point failures. Moreover, as different indexing algorithms may present different performances when processing queries that involve varying amounts of data, utilising the fastest one in the multi-indexing mechanism can boost the indexing performance when tackling different queries.
- RS data are stored in modern data centres equipped with top-tier hardware infrastructures that are highly reliable and less frequent to fail[23]. Therefore, geospatial indices will not be frequently re-constructed due to frequent hardware failures and hence utilising multiple geospatial indices will not cause severe overhead.
- RS data also have characteristics of not being frequently modified, i.e., nearly read-only and stored in a well-structured manner, which benefit geospatial index construction and avoid constant index re-construction. As such, building multiple geospatial indices using relatively light-weighted indexing algorithms is feasible and suitable for RS data and will not cause severe time or space overhead.

In this paper, considering the benefit and suitability of using the multi-indexing mechanism to improve geospatial indexing efficiency, we design a framework named Multi-IndeXing-Remote Sensing (MIX-RS). For the multi-indexing mechanism, two popular and broadly-used geospatial indexing methods, i.e., GeoHash[24–27], QuadTree[28, 29], as well as a traditional indexing method "Orthogonal List", are constructed and unified to form a multi-indexing ensemble. These indexing methods are light-weight, simple, and less computation-intensive compared with other geospatial indexing methods, and hence will not compromise the efficiency in terms of time and space[26, 30, 31]. The MIX-RS then unifies the multi-indexing mechanism on top of the HDFS with data replication. It can improve the geospatial indexing performance and benefit the applicability of RS data while not causing severe overhead. Moreover, the MIX-RS framework does not require fundamental changes and only needs subtle implementation efforts, making it applicable to other applications[32–37]. The prototype of the MIX-RS framework is implemented, and is evaluated using real RS data provided by the Chinese Academy of Sciences[3, 38], demonstrating superior indexing performance over compared indexing methods and frameworks.

In summary, we make the following contributions in this paper:

- We design the MIX-RS framework that naturally unifies the multi-indexing mechanism on top of the HDFS with data replication enabled, which aims for both fault tolerance and geospatial indexing efficiency improvement.
- The proposed MIX-RS framework improves the geospatial indexing performance and the applicability of RS data while not causing severe overhead or requiring sophisticated system implementation.
- We implement the MIX-RS framework and evaluate it using real RS data to validate its excellent geospatial indexing performance.

The rest of this paper is organised as follows: Related geospatial indexing methods, as well as some frameworks that are used to index and query geospatial data, are introduced in Section 2. By analysing these related methods, the rationale and motivation of the proposed MIX-RS framework are introduced, and explained in detail in Section 3. Section 4 presents the empirical evaluation and performance analysis of the MIX-RS framework. Section 5 concludes the paper.

2 Related Work and Opportunity

As the scale of RS data keeps growing rapidly, how to efficiently index the data to satisfy user queries becomes a crucial problem, and thereby has attracted attention from both industry and academic communities. In this section, we first overview widely used geospatial indexing algorithms, then present proposed frameworks that deal with efficient geospatial storage and indexing. Finally, we identify their deficiencies to show the motivations and research opportunities of our proposed multi-indexing mechanism and the MIX-RS framework.

2.1 Geospatial indexing algorithm

Eldawy and Mokbel[18] and Aji et al.[39] proposed the Uniform Grid Index, which is one of the most commonly used geospatial indexing algorithms. The Uniform Grid Index performs geospatial indexing by constructing an index table based on the longitude and latitude coordinates. However, the algorithm needs to traverse the entire index table when looking for a specific coordinate, resulting in tremendous space and time consumption. If, in addition, temporal information is added as a new indexing dimension, then the storage complexity will be severely raised, and the searching and indexing efficiency will be heavily impaired.

Whitman et al.[28] put forward the QuadTree indexing algorithm. As illustrated in Fig. 1, the RS data are commonly divided into four quadrants by major RS applications, such as Google Earth, which makes the QuadTree indexing algorithm very suitable to index RS data. The QuadTree indexing algorithm transforms the quadrants into a tree-like structure, and each quadrant is further divided in the same way, forming a QuadTree with several layers, as shown in Fig. 1. Since each subtree in the QuadTree indexing structure contains a piece of the sub-region of the globe, QuadTree is very efficient when performing regional queries. Compared with the Uniform Grid Index algorithm that simply traverses the longitude and latitude index, the QuadTree possesses better efficiency. Moreover, the space overhead caused by the QuadTree indexing algorithm is not heavy.

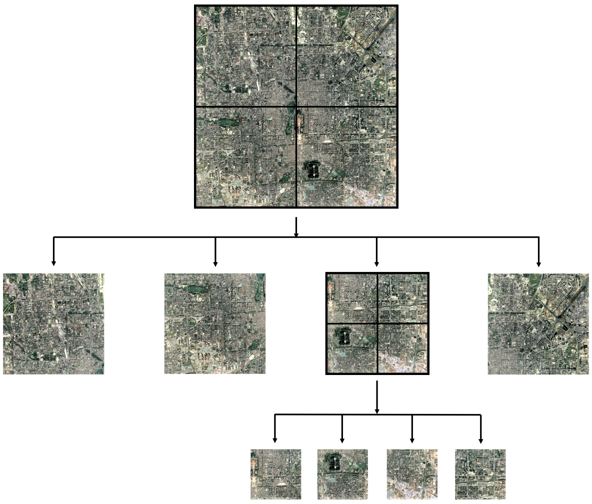


Fig. 1 An illustrative example of the QuadTree algorithm when storing the RS data of Beijing, the capital of China. The RS data are divided into four quadrants and stored into a tree-like structure, where each quadrant represents a sub-region of the city. Each quadrant is then further divided in the same way, forming a QuadTree with several layers.

Fox et al.[24] presented the GeoHash indexing algorithm. GeoHash encodes a geographic location into a string containing letters and digits so that the longer the shared prefix between two geohashes, the spatially closer they are together. The binary coordinate encoding is utilised to convert a coordinate range into a string of binary numbers, which is then grouped into several 6-digit groups, and finally converted using Base32 encoding. An illustrative example of Beijing is presented in Fig. 2. In this example, the Forbidden City (north-west in the lower-right square) and Beijing Central Business District (CBD, north-east in the lower-right square) have their geohashes start with the common prefix “4”. Inside the sub-region with geohash “41”, the Forbidden City and the Qianmen Business District share a common prefix “41” since they are spatially closer to each other. The GeoHash is highly efficient as it reduces the length of the encoding to be stored, leading to significantly

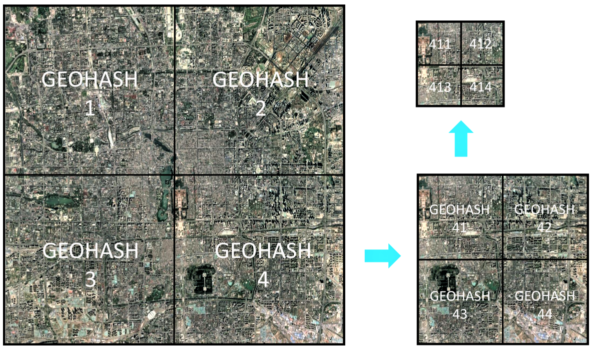


Fig. 2 An illustrative example of the GeoHash algorithm when processing the RS data of Beijing. In the lower-right square, Forbidden City (with GeoHash “41”) and Beijing CBD (with GeoHash “42”) share the same GeoHash prefix “4” as they are spatially close to each other. In the smaller upper-left square, Forbidden City (with GeoHash “411”) and Qianmen Business District (with GeoHash “413”) share a longer common GeoHash prefix which is “41”, since they are geographically closer to each other than the distance between the Forbidden City and Beijing CBD. Hence, GeoHash encodes a geographic location into a string so that the longer the shared prefix between two geohashes, the spatially closer they are together. The GeoHash codes in this example are for illustration purposes only.

better performance, especially when performing regional RS data indexing.

Another widely used algorithm suitable to store data with orthogonal information is the Orthogonal List algorithm[40]. Although not specifically designed for geospatial indexing, the Orthogonal List algorithm can still be a suitable candidate for RS data storage and indexing as its structure can perfectly fit the orthogonal structure such as the longitude and latitude coordinate of a geographic location. As illustrated in Fig. 3, each piece of RS data is linked to its geographically-adjacent region in the Orthogonal List data structure, which makes the Orthogonal List a natural choice to process RS data.

2.2 Geospatial storage and indexing framework

Several research efforts have leveraged the aforementioned popular geospatial indexing algorithms to design geospatial storage and indexing frameworks. Eldawy and Mokbel[18] proposed an RS data storage and indexing framework named SpatialHadoop, which utilises their own designed Uniform Grid Index algorithm. The SpatialHadoop uses the HDFS cluster as the data storage system, and then builds an index on the upper layer of its file system and constructs a MapReduce interface to serve external requests. However, SpatialHadoop suffers from drawbacks caused by executing the MapReduce program, which requires a start-up time that downgraded the performance. Furthermore, SpatialHadoop requires nearly 14 000 lines of code based on Hadoop, which made the implementation very labourious.

The MapReduce-based SHAHED framework was also proposed by Eldawy et al.[21] to query, visualise, and mine large-scale RS data generated by the satellites. Unlike their previous work, SHAHED leverages QuadTree as its indexing method. The SHAHED considers both spatial and temporal aspects of the RS data for effective querying. The query component of the SHAHED constructs the index for querying, and the visualisation component uses the MapReduce program to generate the heatmap related to the user query. Despite these efforts, the SHAHED still suffers from the warm-up time overhead caused by the MapReduce program, which reduces its performance.

Al Naami et al.[20] introduced the Geographic Information System Querying Framework (GISQF) that works on top of the SpatialHadoop framework. The GISQF uses a two-layer geospatial indexing strategy to accelerate query processing. However, the two-layer geospatial indexing not only consumes a tremendous amount of construction time and storage resources, but also requires a sophisticated implementation.

2.3 Motivation and research opportunity

By analysing these aforementioned frameworks, the HDFS is utilised for data storage by all of them. However, these frameworks only use a single geospatial indexing method during index construction and querying, which limits the merit of HDFS parallelism and the performance could be limited by utilising only a single index. Moreover, to efficiently utilise MapReduce, these frameworks require sophisticated implementation efforts to make significant modifications to the MapReduce framework. Hence, it naturally leads to the idea of unifying multi-indexing mechanisms on top of the HDFS as it not only enjoys the benefit of parallelism possessed by the HDFS data replication to speed up the remote

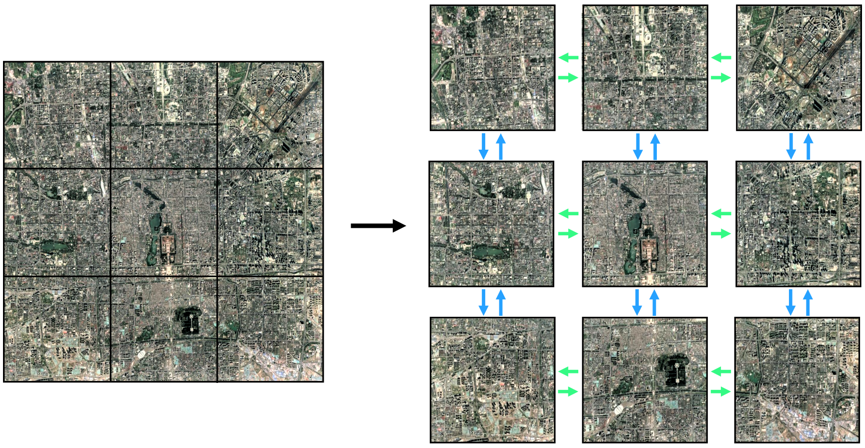


Fig. 3 An illustrative example of the Orthogonal List algorithm when processing the RS data of Beijing. The algorithm stores the RS data into an orthogonal linked list, in which grids are arranged using their coordinate location.

sensing data indexing process, but also requires subtle changes and saves the laborious implementation efforts.

In terms of the proposed multi-indexing mechanism, we utilise two widely-used geospatial indexing algorithms, i.e., GeoHash and QuadTree, as well as a traditional data indexing algorithm, i.e., Orthogonal List. The reasons for adopting these indexing algorithms are as follows:

- The GeoHash and QuadTree algorithms are commonly used by geospatial indexing algorithms for RS data[26, 27, 29]. Although the Orthogonal List is not specifically designed for geospatial indexing, its orthogonal linking structure is a suitable solution to store RS data that possesses orthogonal information, such as longitude and latitude.
- Unlike other indexing algorithms, such as Uniform Grid Index and R-tree[41], the GeoHash, QuadTree, and Orthogonal List algorithms cause subtle time and space overhead, as they do not require complicated indexing data structures. These algorithms are light-weight, simple, and not computation-intensive, which will not compromise efficiency.
- These three indexing algorithms are relatively easy to implement and deploy, requiring lighter implementation efforts.

Hence, by unifying the multi-indexing mechanism on top of the HDFS with data replication, the system efficiency can be increased while not causing severe time or space overhead, and meanwhile avoiding complicated implementation efforts.

3 MIX-RS Framework and Workflow

In this section, we describe the proposed MIX-RS approach in terms of its framework and workflow with a focus on how the multi-indexing is designed on top of the HDFS to improve the geospatial indexing efficiency.

3.1 MIX-RS framework

The framework of the proposed MIX-RS is illustrated in Fig. 4. Its workflows between constituting layers are as follows:

Remote sensing data storage layer: The RS data captured by satellites are handed over to the RS data storage layer, which serves as a preprocessing component to preprocess the data, including image-band separation and metadata extraction. The multi-index of the RS data is then constructed by triggering the geospatial indexing layer, and the constructed multi-index is regarded as part of the metadata. Each piece of RS data with its metadata is replicated into three replicas. The RS data are stored in the underneath HDFS structurally using its geographical coordinate to form a tree directory structure, so that it can benefit the data retrieval and indexing performed later. The RS metadata are stored in PostgreSQL database that will be used to decide involved RS data upon receiving the user query (see Section 3.2.1).

Query interface layer: Users interact with the query interface layer and submit their query by specifying the region using longitude, latitude, and time period. The users also need to specify the type of information that they want. The query is then sent to the RS data storage layer for data retrieval, and the results produced by the geospatial indexing layer are displayed to the users via the query interface layer. An example query is illustrated in Fig. 5 (see Section 3.2.2).

Query calculation layer: Once the data involved in the submitted query are retrieved by the RS data storage layer, the query calculation layer will process it to produce the required information, like the vegetation rate in our experiment, or to calculate the information such as drought rate, etc. (see Section 3.2.3).

Geospatial indexing layer: Upon receiving the resulting RS data from the query calculation layer, three geospatial indexing algorithms are used to transform the RS images to form a holistic view of the area that they cover. For better efficiency, each indexing method will be conducted on one of the data replicas in the HDFS for parallelised computation, and the outcome produced by the fastest method will be utilised as the result (see Section 3.2.4).

3.2 MIX-RS workflow

A detailed explanation of each constituting layer in this section is provided using the framework design, i.e., the remote sensing data storage layer, the query interface layer, the query calculation layer, and the geospatial indexing layer.

3.2.1 Remote sensing data storage layer

The remote sensing data storage layer in the MIX-RS framework is used to preprocess, replicate, and store the RS data received from the satellite. As shown in Step ① in Fig. 4, the RS data are downloaded from the satellite. Since the RS data compose of several bands, such as Normalised Difference Vegetation Index (NDVI), Ratio Vegetation Index (RVI), and Difference Vegetation Index (DVI), etc., hence during preprocessing, this 3D RS data will be flattened into 2D so that bands are separated. The

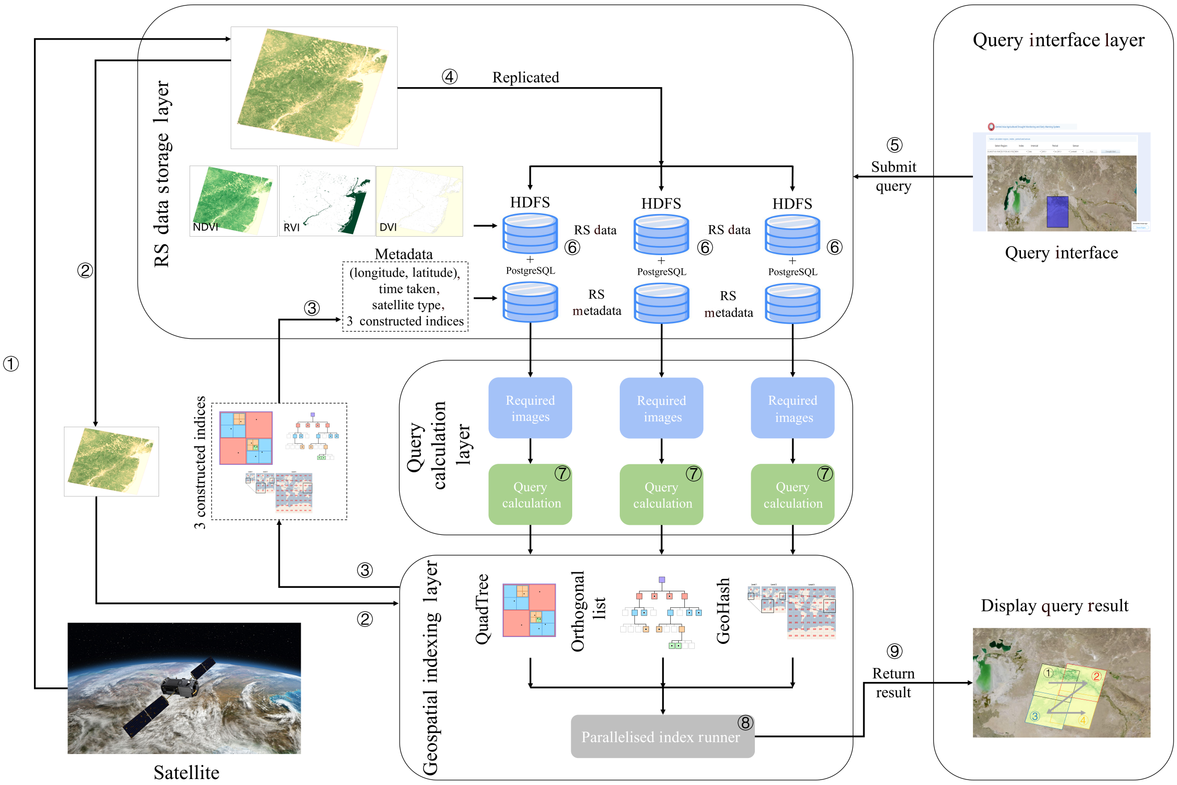


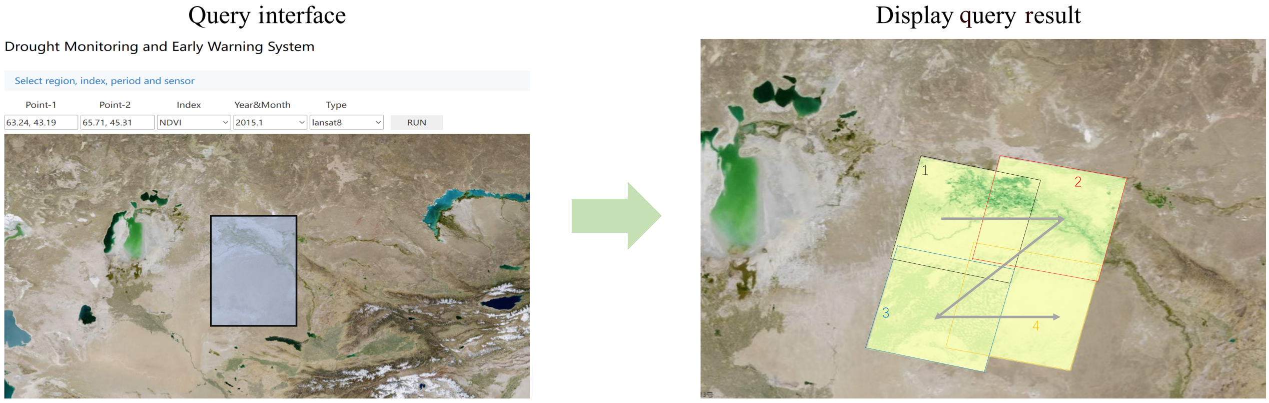
Fig. 4 MIX-RS framework. The framework constitutes four layers, i.e., the RS data storage layer, query interface layer, query calculation layer, and geospatial indexing layer. Step ①: the RS data are downloaded to the RS data storage layer for preprocessing and storage; Step ②: the RS data are directed to the geospatial indexing layer to construct three geospatial indices; Step ③: the three constructed geospatial indices are directed back to the RS data storage layer and are treated as part of the metadata; Step ④: the preprocessed RS data and its metadata are replicated. The RS data are stored into the HDFS structurally based on its geographical coordinate to form a tree structure, as this kind of directory structure can improve the data retrieval and indexing performed in later steps. The RS metadata are stored into the PostgreSQL database; Step ⑤: the submitted user query will be sent from the query interface layer to the RS data storage layer; Step ⑥: the PostgreSQL database will decide which pieces of RS data are involved in the range specified by the submitted query based on the RS metadata. Then, the required RS data are retrieved from the HDFS and used by the query calculation layer; Step ⑦: The query calculation layer calculates the required information; Step ⑧: three geospatial indexing algorithms will be run in parallel, and the result produced by the fastest one will be utilised; Step ⑨: The result will be sent to the query interface layer for visualisation.

metadata of the RS images including longitude, latitude, time taken, and satellite type are also extracted. To prepare for the multi-indexing mechanism used later, the RS image metadata will be processed by the geospatial indexing layer to construct three indices as shown in Step ②. These three constructed indices are then sent back in Step ③ illustrated in Fig. 4 and are treated as part of the metadata. Finally, as indicated by Step ④, the RS data and its metadata will then be replicated for the purpose of fault tolerance. The RS data will be stored into HDFS structurally using its geographical coordinate to form a tree directory structure, since such kind of structure can benefit the data retrieval and indexing performed later. The RS metadata are stored in the PostgreSQL database.

Additionally, as indicated by Step ⑥ in Fig. 4, the remote sensing data storage layer is also responsible for retrieving the RS data required by the submitted query, i.e., RS data that cover the specified geographical range. For instance, in Fig. 5, the specified blue user query shown on the left is covered by four pieces of RS data as shown on the right. The submitted query will specify the longitude and latitude range of interest, which is used by the PostgreSQL database to determine which pieces of RS data are required based on the metadata stored in it. Once the involved RS data are decided, it will be retrieved from the HDFS.

3.2.2 Query interface layer

Once the data preprocessing and storage are completed by the remote sensing data storage layer (Steps ① & ④) and the multi-index is constructed by the geospatial indexing layer (Steps ② & ③), the system is ready to



Query interface

Display query result

Drought Monitoring and Early Warning System

Select region, index, period and sensor

Point-1	Point-2	Index	Year&Month	Type	
63.24, 43.19	65.71, 45.31	NDVI	2015.1	lansat8	RUN

Fig. 5 Illustration of an example query. The user queries the region of interest in the query interface on the left by specifying the longitude, latitude, time period, and image source. The user also needs to specify the type of information they need, in this example, it is NDVI. The user query is then processed by the MIX-RS framework. In this example, the specified region involves four RS images, hence the remote sensing data storage layer needs to firstly retrieve the images involved in the query. The retrieved images are then processed by the query calculation layer to calculate the required information. After calculating the information that the user needs, these four remote sensing images are placed in their designated position based on their geographic metadata by the multi-indexing mechanism. Finally, the constructed query result is returned and visualised by the user interface, as shown on the right-hand side.

receive queries via the query interface layer. Figure 5 presents the query interface and an example user query. When submitting the query, the user needs to specify the longitude and latitude of the area that they want to retrieve. Furthermore, the time period and type of information they are interested in will also be specified. The user query interface is illustrated on the left-hand side of Fig. 5. The query will then be submitted to the remote sensing data storage layer, as shown in Step ⑤ in Fig. 4. In the end, the final result will be returned to the query interface layer in Step ⑨ in Fig. 4 for result display and visualisation, as shown on the right-hand side of Fig. 5.

3.2.3 Query calculation layer

As illustrated in Fig. 4, the remote sensing data storage layer will prepare the required RS data for the query calculation layer. The query calculation layer can calculate different types of geospatial information (Step ⑦), such as NDVI as shown in the following:

$$\text{NDVI} = \frac{\text{NIR} - \text{RED}}{\text{NIR} + \text{RED}} \tag{1}$$

Note that NDVI is one of the indicator metrics that are valuable to assessing the rate of vegetation coverage to reflect the vegetation and nutrition condition of the area. The NIR is the infrared reflection rate and RED is the red-light reflection rate. Both the NIR and RED are two out of ten bands in every piece of RS data. The calculation is performed pixel-wise, and the result will still be a matrix with the same dimension as each original band of the RS data. The calculations are also performed in parallel between each processing node of the HDFS.

3.2.4 Geospatial indexing layer

The idea of multi-indexing is applied in the geospatial indexing layer to speed up the construction of the result. The calculated result produced by the query calculation layer will be indexed by three geospatial indexing methods, i.e., GeoHash, QuadTree, and Orthogonal List, and are performed in parallel. As shown in Step ⑧ in Fig. 4, the parallelised index runner monitors the indexing progress of each processing node and utilises the fastest one that produces the result. By leveraging the multi-indexing as an ensemble on top of the HDFS with data replication, the delay on one or two of the processing nodes will not affect the overall performance since the fastest indexing will produce the result to satisfy the user query. Hence, it is reasonable that the multi-indexing mechanism can boost the overall system performance and outperform those systems that only use a single index, even though the data replication of the HDFS provides a perfect prerequisite to utilise multi-indexing. Once the result is generated, it will be returned to the query interface layer in Step ⑨, as shown in Fig. 4.

3.3 Prototype implementation detail

To verify the effectiveness of our proposed MIX-RS framework and to make it become applicable in practice, we follow the architecture as depicted in Fig. 4 to implement the proposed MIX-RS framework with corresponding designed layers and steps. The details of the prototype implementation are illustrated in Fig. 6.

To receive the user queries, we implement a web-based user interface that allows the users to specify the region of interest by giving the longitude and latitude range of the region. To complete the query, the users also need to provide the time period, image source, and the type of information they need. The submitted query is then sent to the RequestHandler residing in the master server via Kafka request message. Upon receiving the user query, the RequestHandler will ask the PostgreSQL metadata database to determine which pieces of RS data are involved in this query. Once completed, the master server broadcasts the information of the required data to the HDFS servers for data retrieval and geospatial indexing. Three HDFS servers will execute the geospatial indexing in parallel, and the fastest result received by the RequestHandler from these HDFS servers will be utilised and sent back to the user interface via Kafka messaging. Finally, the result will be displayed on the web user interface.

4 Empirical Evaluation

To validate the effectiveness of the MIX-RS framework, comprehensive evaluations are performed on real RS data captured by the LandSat8 satellite. We verify the superiority of both the multi-indexing mechanism, which is compared with the single-indexing techniques, and the overall MIX-RS system, which is compared with other widely-used systems, such as SpatialHadoop[18], GISQF[20], and SHAHED[21].

4.1 Experimental setup and dataset

To verify the efficacy of the MIX-RS system, real RS data captured by the LandSat8 satellite are utilised. The data are from the Central Asian Ecology and Environment Research Centre of the Chinese Academy of Sciences[3, 38] and contain about 9000 pieces of RS data, with a total size of around 4TB. The RS data are in the *geotiff* file format, with a dimension of 7000 × 7000 × 10. Each experiment is repeated 50 times and the corresponding results have been plotted with error bars.

In terms of system hardware configurations, the MIX-RS system is deployed on five servers, one as the HDFS master node and three as the HDFS slave nodes, where each HDFS slave node stores one replica of the RS data. One extra server is used to deploy the query user interface. All servers used during experiments have the same hardware and Operating System (OS) configuration as shown in Table 1. In terms of software configurations, all servers have Kafka 2.3.1 installed to build a message queue for query-result communications, and have PostgreSQL 11.2 installed to serve as the RS metadata database. Besides, all HDFS servers have

Table 1 Hardware and OS configuration of the server infrastructure (The same configuration is applied for all servers.).

Item	Configuration
CPU	Intel(R) Xeon(R) CPU E5-2630 v4@2.20 GHz (10 cores, 20 threads)
Number of CPUs	2
Memory size	64 GB
Disk space	25 TB
OS	Ubuntu 16.04.5 LTS
Cluster configuration	1 HDFS master node + 3 HDFS slave nodes + 1 query server

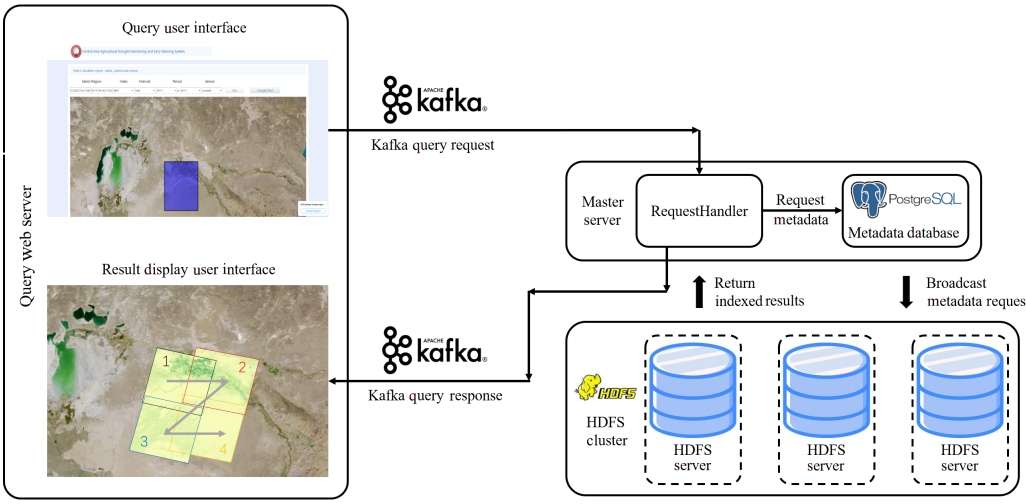


Fig. 6 Illustration of the MIX-RS prototype implementation.

Hadoop 2.9.2 installed.

4.2 Performance evaluation of the multi-indexing mechanism

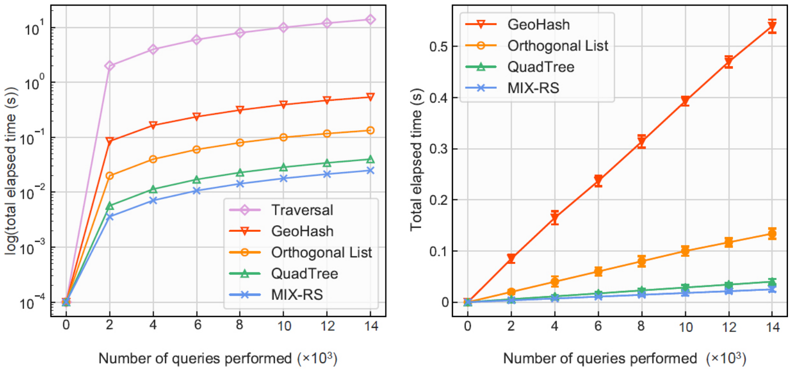
To demonstrate the excellent performance and efficiency of the proposed multi-indexing mechanism against single indexing methods, the experimental results of the comparison are shown in Fig. 7. As illustrated in Fig. 7a, the line-chart clearly indicates that the multi-indexing mechanism outperforms all other single indexing methods when handling different number of random queries, indicated by the shortest amount of time elapsed. The GeoHash, Orthogonal List, and QuadTree indexing methods cost nearly 2060%, 436%, and 60% more processing elapsed time when processing different number of user queries than the multi-indexing mechanism, respectively. Therefore, the performance boost provided by the multi-indexing mechanism is very significant and can benefit the overall MIX-RS framework in terms of RS data indexing.

In terms of the scalability of the multi-indexing method, Fig. 7b is the magnified version of Fig. 7a without showing the brute force traversal method and hence the log-scale can be removed for better illustration. When the number of queries being processed keeps increasing, the total elapsed time increases linearly. The linear trend indicates that the method scales well in terms of query workloads.

4.3 Overhead of the multi-indexing mechanism

An excellent indexing design should enjoy superb performance and possess an acceptable overhead. To verify that the multi-indexing leveraged in the MIX-RS does not have a severe overhead, the memory consumption and the index construction time are measured to demonstrate the method is space-efficient, and the index construction overhead is acceptable.

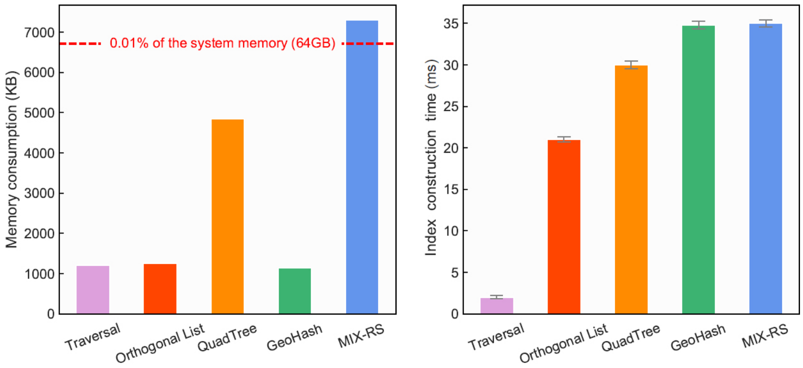
As shown in Fig. 8a, the memory consumption of both the multi-indexing mechanism and single indexing methods are measured. It is natural to observe that



(a)

(b)

Fig. 7 Performance comparisons between the multi-indexing and single indexing methods when processing different number of queries. Since the performance of the brute force traversal method is much worse than other methods, hence in (a), log-scale is applied on the *y*-axis. To better visualise the relationships between methods and the linear trends, the brute force traversal is eliminated in (b) and the log-scale in *y*-axis is also removed.



(a)

(b)

Fig. 8 Evaluation of space and time overhead of multi-indexing and single indexing methods. (a) Amount of memory consumed by each method. The total system memory for each server is 64 GB, and the red dotted line marks 0.01% of the system memory. (b) Index construction time of each method, i.e., how long it takes for each method to complete the construction of index/indices it needs.

the multi-indexing mechanism consumes most memory, which is around 7100 KB. Apparently, constructing three indices should consume more memory than constructing a single index. Despite the highest memory consumption, the memory overhead of the multi-indexing mechanism is totally acceptable. Since for modern server infrastructures, nearly all servers are equipped with a memory that is orders of magnitude greater than the memory consumed by the multi-indexing mechanism. The servers we used have a memory of 64 GB. As indicated by the red dotted line that marks 0.01% of the system memory, the memory consumption of the multi-indexing mechanism only consumes approximately 0.011% of the system memory. Therefore, the memory overhead is subtle and negligible.

For the index construction time shown in Fig. 8b, the multi-indexing mechanism achieves nearly the same performance compared with the slowest single indexing method GeoHash, which is approximately 35 ms. The three indices of the multi-indexing mechanism are constructed in parallel in each processing node, hence, it is natural to observe that the multi-indexing achieves nearly the same construction overhead with the slowest single method, which is not too long. Hence, the index construction overhead of the multi-indexing mechanism is not severe.

Therefore, the multi-indexing mechanism brings significant performance improvement while not causing a noticeable overhead, demonstrating that using the multi-indexing mechanism is promising.

4.4 Performance evaluation at the framework level

To verify the performance of the overall framework, we compare the MIX-RS with three widely-used geospatial information systems, i.e., SpatialHadoop, GISQF, and SHAHED. The performance comparison in terms of query processing time, system establishment time, and system memory consumption are presented in Figs. 9a, 9b, and 9c, respectively.

As shown in Fig. 9a, the MIX-RS has the fastest query time among all frameworks. The SpatialHadoop, GISQF, and SHAHED cost 64, 49, and 1.5 times more query processing time when processing a relatively large query that involves around 50 GB of RS data than the MIX-RS, respectively. This is due to the MIX-RS not only utilises multi-indexing on top of the HDFS with the benefit brought by data replication and parallel computation, but also avoids the warm-up overhead caused by the MapReduce, unlike other compared frameworks. Moreover, the query elapsed time linearly correlates with the size of the data involved in a single query, which justifies that the MIX-RS presents excellent scalability when the amount of data involved in the query varies.

As shown in Fig. 9b, the MIX-RS is the fastest to be established, i.e., the time it takes from loading the data to finish constructing the index. The SpatialHadoop, GISQF, and SHAHED cost 980%, 340% and 600% more system establishment time than the MIX-RS, respectively. The evaluation result further demonstrates that the MIX-RS stands out by having the lowest system establishment time overhead.

In terms of system memory consumption overhead, it is natural to observe from Fig. 9c that the MIX-RS requires the highest amount of memory. However, compared with the memory capacity possessed by

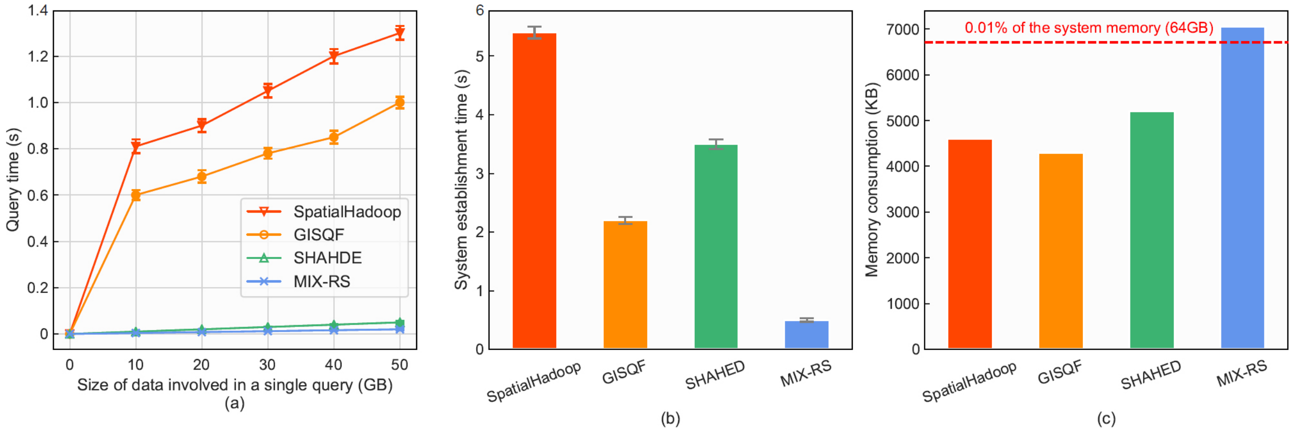


Fig. 9 Performance and overhead comparisons at the framework level. (a) Amount of time taken to complete queries that involve different RS data sizes. (b) System establishment time of each system to indicate the time overhead. (c) Memory consumption of each system to show the space overhead of each method. The total system memory for each server we used is 64 GB, and the red dotted line marks 0.01% of the system memory.

modern hardware infrastructures we used, the amount of memory consumed by MIX-RS is only slightly greater than 0.01% of the entire 64 GB system memory, which is only a tiny portion. As such, despite naturally the MIX-RS consumes the highest amount of memory, the ample space makes the overhead negligible. By demonstrating a low time and space overhead, the excellent performance and subtle overhead make the MIX-RS very applicable.

5 Conclusion

In this paper, we propose the MIX-RS framework for more efficient RS data indexing. The MIX-RS naturally unifies the multi-indexing mechanism on top of the HDFS with data replication enabled. This unification is natural when data replication presents since it can utilise parallelism to boost the indexing performance. Hence, this holistic framework combines the fault tolerance provided by the HDFS and the indexing performance speedup provided by the multi-indexing mechanism. Moreover, the MIX-RS framework requires very little effort in terms of framework implementation and has a low modification complexity. The framework causes very little time and space overhead to construct, making it feasible and applicable. Comprehensive experiments using real RS data are performed to verify the effectiveness of the multi-indexing mechanism, demonstrating the superior performance of the MIX-RS system. Furthermore, both time and space overhead are evaluated, demonstrating the applicability of the MIX-RS system.

Acknowledgment

This work was supported in part by Key-Area Research and Development Program of Guangdong Province (No. 2020B010164002) and the Fundamental Research Foundation of Shenzhen Technology and Innovation Council (No. KCXFZ20201221173613035).

References

[1] The National Aeronautics and Space Administration, https://www.nasa.gov/, 2021.

[2] European Space Agency, https://www.esa.int/, 2021.

[3] LandSat Science, Landsat 8 overview, https://landsat.gsfc.nasa.gov/landsat-8, 2021.

[4] J. W. Wang, X. Huang, J. Y. Zheng, C. Rajapakshe, S. Kay, L. Kandoor, T. Maxwell, and Z. B. Zhang, Scalable aggregation service for satellite remote sensing data, in *Proc. 20th Int. Conf. Algorithms and Architectures for Parallel Processing*, New York, NY, USA, 2020, pp. 184–199.

[5] Y. B. Huang, Z. X. Chen, T. Yu, X. Z. Huang, and X. F. Gu, Agricultural remote sensing big data: Management and applications, *J. Integrat. Agric.*, vol. 17, no. 9, pp. 1915–1931, 2018.

[6] D. M. Huang, X. N. Liu, B. M. Song, J. Chen, S. Masae, Y. S. Wang, T. Shigeo, H. Yoshimichi, and Y. Yasuo, Vegetation spatial heterogeneity of different soil regions in Inner Mongolia, China, *Tsinghua Science and Technology*, vol. 12, no. 4, pp. 413–423, 2007.

[7] D. M. Huang, Y. S. Wang, S. Masae, X. N. Liu, B. M. Song, J. Chen, T. Shigeo, H. Yoshimichi, and Y. Yasuo, Spatial heterogeneity of vegetation in China, *Tsinghua Science and Technology*, vol. 12, no. 4, pp. 424–434, 2007.

[8] J. Y. Liang and D. S. Liu, Estimating daily inundation probability using remote sensing, riverine flood, and storm surge models: A case of hurricane harvey, *Remote Sens.*, vol. 12, no. 9, p. 1495, 2020.

[9] M. Chen, S. W. Mao, and Y. H. Liu, Big data: A survey, *Mobile Netw. Appl.*, vol. 19, no. 2, pp. 171–209, 2014.

[10] M. Li, J. S. Wu, J. B. Dai, Q. S. Jiang, Q. Qu, X. L. Huang, and Y. Wang, A self-contained and self-explanatory DNA storage system, *Sci. Rep.*, vol. 11, p. 18063, 2021.

[11] J. M. Haut, M. E. Paoletti, S. Moreno-Álvarez, J. Plaza, J. A. Rico-Gallego, and A. Plaza, Distributed deep learning for remote sensing data interpretation, *Proc. IEEE*, vol. 109, no. 8, pp. 1320–1349, 2021.

[12] M. S. Warren, S. P. Brumby, S. W. Skillman, T. Kelton, B. Wohlberg, M. Mathis, R. Chartrand, R. Keisler, and M. Johnson, Seeing the earth in the cloud: Processing one petabyte of satellite imagery in one day, in *Proc. of the 2015 IEEE Applied Imagery Pattern Recognition Workshop* (*AIPR*), Washington, DC, USA, 2015, pp. 1–12.

[13] L. H. Li, W. P. Jing, and N. H. 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.

[14] B. E. B. Semlali and C. El Amrani, Big data and remote sensing: A new software of ingestion, *Int. J. Electr. Comput. Eng.*, vol. 11, no. 2, pp. 1521–1530, 2021.

[15] Z. C. Xing and G. M. Li, Intelligent classification method of remote sensing image based on big data in spark environment, *Int. J. Wirel. Inf. Netw.*, vol. 26, no. 3, pp. 183–192, 2019.

[16] P. Y. Wang, J. Q. Wang, Y. Chen, and G. Y. Ni, Rapid processing of remote sensing images based on cloud computing, *Future Gener. Comput. Syst.*, vol. 29, no. 8, pp. 1963–1968, 2013.

[17] A. K. Karun and K. Chitharanjan, A review on hadoop-HDFS infrastructure extensions, in *Proc. of the 2013 IEEE Conf. Information & Communication Technologies*, Thuckalay, India, 2013, pp. 132–137.

[18] A. Eldawy and M. F. Mokbel, SpatialHadoop: A MapReduce framework for spatial data, in *Proc. of the 2015 IEEE 31st Int. Conf. Data Engineering*, Seoul, Republic of Korea, 2015, pp. 1352–1363.

[19] A. Eldawy, Y. Li, M. F. Mokbel, and R. Janardan, Cg_Hadoop: Computational geometry in MapReduce, in *Proc. 21st ACM SIGSPATIAL Int. Conf. Advances in Geographic Information Systems*, Orlando, FL, USA, 2013, pp. 294–303.

[20] K. M. Al Naami, S. Seker, and L. Khan, GISQF: An efficient spatial query processing system, in *Proc. of the 2014 IEEE 7th Int. Conf. Cloud Computing*, Anchorage, AK, USA, 2014, pp. 681–688.

[21] A. Eldawy, M. F. Mokbel, S. Alharthi, A. Alzaidy, K. Tarek, and S. Ghani, SHAHED: A MapReduce-based system for querying and visualizing spatio-temporal satellite data, in *Proc. of the 2015 IEEE 31st Int. Conf. Data Engineering*, Seoul, Republic of Korea, 2015, pp. 1585–1596.

[22] M. W. Ding, L. Zheng, Y. C. Lu, L. Li, S. Guo, and M. Y. Guo, More convenient more overhead: The performance evaluation of Hadoop streaming, in *Proc. 2011 ACM Symp. Research in Applied Computation*, Miami, FL, USA, 2011, pp. 307–313.

[23] X. F. Lü, C. Q. Cheng, J. Y. Gong, and L. Guan, Review of data storage and management technologies for massive remote sensing data, *Sci. China Technol. Sci.*, vol. 54, no. 12, pp. 3220–3232, 2011.

[24] A. Fox, C. Eichelberger, J. Hughes, and S. Lyon, Spatio-temporal indexing in non-relational distributed databases, in *Proc. of the 2013 IEEE Int. Conf. Big Data*, Silicon Valley, CA, USA, 2013, pp. 291–299.

[25] I. S. Suwardi, D. Dharma, D. P. Satya, and D. P. Lestari, Geohash index based spatial data model for corporate, in *Proc. of the 2015 Int. Conf. Electrical Engineering and Informatics* (*ICEEI*), Denpasar, Indonesia, 2015, pp. 478–483.

[26] K. Y. Huang, G. Q. Li, and J. Wang, Rapid retrieval strategy for massive remote sensing metadata based on GeoHash coding, *Remote Sens. Lett.*, vol. 9, no. 11, pp. 1070–1078, 2018.

[27] J. J. Liu, H. R. Li, Y. Gao, H. Yu, and D. Jiang, A GeoHash-based index for spatial data management in distributed memory, in *Proc. of the 2014 22nd Int. Conf. Geoinformatics*, Kaohsiung, China, 2014, pp. 1–4.

[28] R. T. Whitman, M. B. Park, S. M. Ambrose, and E. G. Hoel, Spatial indexing and analytics on Hadoop, in *Proc. 22nd ACM SIGSPATIAL Int. Conf. Advances in Geographic Information Systems*, Dallas, TX, USA, 2014, pp. 73–82.

[29] C. Xu, X. P. Du, Z. Z. Yan, and X. T. Fan, ScienceEarth: A big data platform for remote sensing data processing, *Remote Sens.*, vol. 12, no. 4, p. 607, 2020.

[30] P. Petrov, P. Dimitrov, and S. Petrova, GEOHASH-EAS—A modified geohash geocoding system with equal-area spaces, in *Proc. of the 18th Int. Multidisciplinary Scientific GeoConference SGEM2018*, Bulgaria, Russia, 2018, pp. 187–194.

[31] N. Guo, W. Xiong, Y. Wu, L. Chen, and N. Jing, A geographic meshing and coding method based on adaptive Hilbert-Geohash, *IEEE Access*, vol. 7, pp. 39815–39825, 2019.

[32] V. Mithal, A. Khandelwal, S. Boriah, K. Steinhaeuser, and V. Kumar, Change detection from temporal sequences of class labels: Application to land cover change mapping, in *Proc. 2013 SIAM Int. Conf. Data Mining*, Austin, TX, USA, 2013, pp. 650–658.

[33] J. H. Faghmous, M. Le, M. Uluyol, V. Kumar, and S. Chatterjee, A parameter-free spatio-temporal pattern mining model to catalog global ocean dynamics, in *Proc. of the 2013 IEEE 13th Int. Conf. Data Mining*, Dallas, TX, USA, 2013, pp. 151–160.

[34] T. Yu, N. Chawla, and S. Simoff, *Computational Intelligent Data Analysis for Sustainable Development*. New York, NY, USA: CRC Press, 2013.

[35] W. W. Jiang and L. Zhang, Geospatial data to images: A deep-learning framework for traffic forecasting, *Tsinghua Science and Technology*, vol. 24, no. 1, pp. 52–64, 2019.

[36] Z. Y. Zhang, X. N. Tong, K. T. McDonnell, A. Zelenyuk, D. Imre, and K. Mueller, An interactive visual analytics framework for multi-field data in a geo-spatial context, *Tsinghua Science and Technology*, vol. 18, no. 2, pp. 111–124, 2013.

[37] S. Li, B. H. Xie, J. S. Wu, Y. Zhao, C. H. Liu, and Z. M. Ding, Simultaneous semantic alignment network for heterogeneous domain adaptation, in *Proc. 28th ACM Int. Conf. Multimedia*, Seattle, WA, USA, 2020, pp. 3866–3874.

[38] RCEECA CAS, Central Asian Ecology and Environment Research Center of Chinese Academy of Sciences, http://www.egi.cas.cn/yjpt/zgkxyzystyhjyjzx_163317/, 2021.

[39] A. Aji, F. S. Wang, H. Vo, R. Lee, Q. L. Liu, X. D. Zhang, and J. Saltz, Hadoop GIS: A high performance spatial data warehousing system over MapReduce, *Proc. VLDB Endow.*, vol. 6, no. 11, pp. 1009–1020, 2013.

[40] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[41] T. Zhang, L. H. Yang, D. H. Shen, and Y. L. Fan, An efficient in-memory R-tree construction scheme for spatio-temporal data stream, in *Proc. of the ADMS*, *ASOCA*, *ISYyCC*, *CloTS*, *DDBS*, and *NLS4IoT*, Hangzhou, China, 2019, pp. 253–265.

Jiashu Wu received the BS degree in computer science and financial mathematics & statistics from the University of Sydney, Australia in 2018, and the MEng degree in artificial intelligence from the University of Melbourne, Australia in 2020. He is currently a PhD candidate at the University of Chinese Academy of Sciences. His research interests include big data and cloud computing.

Jingpan Xiong received the BEng degree in software engineering from Wuhan Engineering University in 2017. He is now a master student in computer science at the University of Chinese Academy of Sciences. His research interests include big data storage, big data processing, and machine-learning applications.

Hao Dai received the BEng and MEng degrees in communication and electronic technology from Wuhan University of Technology in 2015 and 2017, respectively. He is currently a PhD candidate at Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. His research interests include mobile edge computing, federated learning, and deep reinforcement learning.

Yang Wang received the BS degree in applied mathematics from Ocean University of China in 1989, the MEng degree in computer science from Carleton University, Canada in 2001, and the PhD degree in computer science from the University of Alberta, Canada in 2008. He is currently a professor at Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. His research interests include cloud computing, big data analytics, and Java virtual machine on multicores. He is an Alberta Industry R&D Associate (2009–2011) and a Canadian Fulbright Scholar (2014–2015).

Chengzhong Xu received the PhD degree from the University of Hong Kong, China in 1993. He is currently the dean of Faculty of Science and Technology, University of Macau, China, and the director of the Institute of Advanced Computing and Data Engineering, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. His research interests include parallel and distributed systems and cloud computing. He has published more than 200 papers in journals and conferences. He serves on a number of journal editorial boards, including *IEEE TC*, *IEEE TPDS*, *IEEE TCC*, *JPDC*, and *China Science Information Sciences*. He is a fellow of the IEEE.