A Ceph-based storage strategy for big gridded remote sensing data

Xinyu Tang\textsuperscript{a,b}, Xiaochuang Yao\textsuperscript{a,b}, Diyou Liu\textsuperscript{a,b}, Long Zhao\textsuperscript{c,d}, Li Li\textsuperscript{a}, Dehai Zhu\textsuperscript{a,b} and Guoqing Li\textsuperscript{a,c,d}

\textsuperscript{a}College of Land Science and Technology, China Agricultural University, Beijing, China; \textsuperscript{b}Key Laboratory of Remote Sensing for Agri-Hazards, Ministry of Agriculture and Rural Affairs, Beijing, China; \textsuperscript{c}Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing, China; \textsuperscript{d}School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing, China

\textbf{ABSTRACT}

When using distributed storage systems to store gridded remote sensing data in large, distributed clusters, most solutions utilize big table index storage strategies. However, in practice, the performance of big table index storage strategies degrades as scenarios become more complex, and the reasons for this phenomenon are analyzed in this paper. To improve the read and write performance of distributed gridded data storage, this paper proposes a storage strategy based on Ceph software. The strategy encapsulates remote sensing images in the form of objects through a metadata management strategy to achieve the spatiotemporal retrieval of gridded data, finding the cluster location of gridded data through hash-like calculations. The method can effectively achieve spatial operation support in the clustered database and at the same time enable fast random read and write of the gridded data. Random write and spatial query experiments proved the feasibility, effectiveness, and stability of this strategy. The experiments prove that the method has higher stability than, and that the average query time is 38% lower than that for, the large table index storage strategy, which greatly improves the storage and query efficiency of gridded images.

\section{Introduction}

Traditional remote sensing image storage strategies generally use file-based or relational databases to store remote-sensing image data (Barclay, Gray, & Slutz, 2000; Li & Narayanan et al., 2004; Zeiler, 1999). However, major disadvantages of the traditional approach are that it is only designed for stand-alone storage, it is too dependent on physical hardware, and has poor scalability. With the development of remote sensing big data, this method is gradually being eliminated (Jhummarwala, Potdar, & Chauhan, 2014; Jing & Tian, 2018; Wang & Wang, 2010; Wang, Zhong, & Wang, 2019).

Compared with traditional methods, distributed databases have the obvious advantages of using big data, having good scalability, and utilizing concurrency, and are gradually becoming the main storage method for remote sensing images (Galić,
Mešković, & Osmanović, 2017; Li, Lu, Gong, & Zhang, 2014; Liu, Hao, & Yang, 2019; Lv, Li, Lv, & Xiu, 2019; Nishimura, Das, Agrawal, & El Abbadi, 2011; Rajak, Raveendran, Bh, & Medasani, 2015; Wang, Cheng, Wu, Wu, & Teng, 2015; Wang et al., 2019; Weipeng, Dongxue, Guangsheng, & Yiyuan, 2018; Xu, Du, Yan, & Fan, 2020; Zhou et al., 2019). However, unlike ordinary big data products, the characteristics of multi-dimensionality, multi-structure, and multi-access requirements of remote sensing image data also pose great challenges to distributed spatial databases.

To address this challenge, when using distributed storage systems to store remotely sensed images, researchers tend to build distributed spatiotemporal databases using a spatiotemporal index, a technique based on big table indexes (Chang et al., 2008; Cheng, Kotoulas, Ward, & Theodoropoulos, 2014; Wang et al., 2020; Zhang, Sun, Su, Liu, & Liu, 2020). However, this strategy used by researchers often performs poorly in practical applications and suffers from severe degradation in complex situations. This section will introduce the research on the distributed storage systems and the spatiotemporal index used in them.

1.1. Related work

1.1.1. Distributed storage systems

Some researchers have divided the distributed storage system into three categories: distributed file system, distributed relational database, and distributed NoSQL/NewSQL database (Wang et al., 2015, 2019).

Popular distributed file systems are mainly used to solve the problems of limited storage space and multiple concurrent accesses, specifically including HDFS and IPFS (Arafa, Barai, Zheng, & Badawy, 2018; Benet, 2014; Borthakur et al., 2008; Hu et al., 2020; Rajak et al., 2015). Although this system is relatively simple and efficient, it is difficult to provide spatiotemporal semantic support for remote sensing image storage and lacks integrity checking. Distributed relational databases are obtained by adding distributed control methods to traditional relational databases, including PostgreSQL clusters, MySQL clusters, etc. (Davies & Fisk, 2006; Martinho, Almeida, Simões, & Sá-Marques, 2020). The main problem with this method is that the relational database design, atomicity consistency isolation durability (ACID) structured transactions, and many other designs are made for single computer systems and do not perform well in a clustered environment. A distributed NoSQL/NewSQL database is a new type of database designed to reduce ACID transactions from relational databases to improve transaction processing performance. Distributed NoSQL/NewSQL databases not only simplify the development and maintenance process but also reduce the total cost of operation when managing various unstructured data. Distributed NoSQL/NewSQL databases are also widely used in many storage systems, such as HBase and Ceph databases (Aghayev et al., 2019; Arafa et al., 2018; Li et al., 2014; Salanio et al., 2015; Weil, Brandt, Miller, Long, & Maltzahn, 2006; Zhou et al., 2019).

In image storage, the distributed NoSQL database is more suitable for the unstructured characteristics of remote-sensing images. It performs well for the storage, management, calculation, and visualization of multi-source remote sensing images and has gradually become the main organization method of remote sensing images (Hajjaji & Farah, 2018; Lv et al., 2019).
1.1.2. The spatiotemporal index in the database

With the shift of remote sensing image storage away from traditional relational databases to distributed NoSQL databases, the fact that distributed NoSQL databases themselves do not support spatiotemporal complex queries in the databases has become a prominent issue. In response to this problem, many researchers and companies have conducted related studies and proposed many excellent solutions, among which the most important storage method is to establish a spatial index using index tables. Meanwhile, remote sensing image grids are widely used in research (Ma et al., 2021; Yan et al., 2021; Yao et al., 2020). To be able to retrieve the gridded image of different spatial ranges, data indexes are established based on the efficient retrieval mechanism of spatial index tables in massive data, and geographically adjacent images are controlled to be adjacent in storage as well through special settings of big table row keys. Representative databases include BigTable proposed by Google, HBase under Hadoop ecology, etc. (Carstoiu, Cernian, & Olteanu, 2010; Chang et al., 2008).

Among these, the most basic method for the distributed storage of remote sensing images is to direct the spatial encoding, such as linear quadtree and Hilbert encoding as database key values, to add spatial support capabilities to the distributed database. The main significance of establishing a spatial index in the distributed database at this stage is to provide spatial support and to control geographically adjacent images to be adjacent in storage as well, improving the performance of scale queries (Li et al., 2014). However, this solution can only provide the simplest spatial support and cannot avoid the problem of multi-conditional queries mentioned above.

To deal with multi-conditional query problems, researchers mainly adopt methods such as establishing multi-level indexes and using coprocessors to strengthen data management. The pioneering study in this regard is the MD-HBase system developed by Nishimura et al. (2011). It notes the weakness of HBase in multidimensional data management and establishes a multi-level index based on the original method to enhance the ability of the database to support spatial data. Based on this study, some scholars have combined two ideas, gridded coding and Hilbert coding, and used an HBase coprocessor to achieve the secondary index, which effectively reduces the query time (Hu et al., 2020; Jing & Tian, 2018; Li et al., 2014; Wang et al., 2015).

In the above research, HBase mainly uses built-in filters, coprocessors, and other functions to solve spatiotemporal queries, maintaining a hash index or geographic index table to reduce the difficulty of a query. However, this scheme has many drawbacks. First, for different application requirements and different multi-conditional query statements, it is necessary to design the corresponding index structure and index table. The development cost is high and the generality is poor, so it cannot cope with a complex application environment. Second, the complex query mechanism at the bottom of HBase cannot be simplified, and the performance cannot be fundamentally improved.

1.2. Research contribution

This study analyzes the reasons for the poor performance of existing strategies and constructs an optimized storage strategy based on gridded image metadata by combining key technologies such as the Ceph system architecture and the CRUSH distribution control algorithm provided by Ceph.
2. **Method**

In this paper, a remote sensing, big data storage strategy is proposed to deal with the multi-conditional query problem encountered in the traditional distributed big table index database storage. First, this section will introduce the problem of the traditional distributed big table index database storage gridded image strategy represented by HBase and analyze the reasons for the problem. Next, this section will specifically introduce a storage strategy to solve the description problem.

2.1. **The technical shortcomings of big table index**

Before introducing the problems arising from big table index storage for multi-conditional queries, this section will first briefly introduce the storage strategy adopted by big table index storage databases on gridded images. Since HBase is a typical database that adopts a big table index storage strategy and is also widely used in various studies due to its open-source feature, this section will use HBase as an illustrative example. HBase is a column-oriented open-source database where data are stored through tables with rows and columns. However, unlike the structure of traditional relational data tables, HBase tables are multi-dimensional, sparse mappings. They use a key value as the basic unit of data storage, i.e. each datum is identified by a unique corresponding key value to store the location, and the data are indexed by a sequential scan operation on the key value. A specific key value is determined by rowkey, timestamp, column family, column name, and column qualifier. When stored, HBase stores the data rows in key order. In practice, highly related or often simultaneously read data are generally assigned similar rowkey values to improve the performance of bulk reads. Therefore, when using HBase to store gridded images, researchers generally use the spatial index to assign similar rowkey values to spatially similar gridded images and store the attributes and gridded remote sensing data in the columns of the data rows. This design optimizes the scanning operation, and remote sensing images are stacked in the order of spatial extent, attribute information, and time. When it is necessary to find remote sensing images within a certain range, specifying the starting and ending key values of the search, all images within the range can be found easily and quickly.

The problem is that although this design stores spatially adjacent grids together to improve retrieval efficiency, there are still some problems in certain retrieval scenarios. Take a typical rowkey design, as shown in (Figure 1 and 2), as an example. When the conditions of the query are not in the optimal position of the rowkey design, HBase will have to split a query into multiple small-scale queries to operate. For example, the top-level data type code retrieval. When you need to query all image grids of the specified data type code, the query needs to locate the km grids code, record each grid with the specified data type code, and then calculate the starting and ending key values of each h-km grids code within the specified data type code range and the ending key values within the specified data type code range. This complicated operation makes the query more difficult, and also makes the design of similar grids with similar key values ineffective.

When the query conditions are extended to multi-conditional queries, the situation is even worse. HBase may have to mobilize all the nodes of the cluster to process a complex query statement, and after executing a large number of simple conditional queries, the results of
that query are then filtered twice to get the final result. At the same time, due to the limitations of HBase based on the distributed file system HDFS, this type of large-scale random read and write is often difficult to complete quickly.

From the analysis in this section, we can find that there are two main performance bottlenecks in the big table index strategic distributed spatial database: spatial object retrieval and spatial object location. The performance of spatial object retrieval design will degrade sharply with the complexity of the spatial query, and it is difficult to get them involved in the query object quickly according to the query conditions. Spatial object location, on the other hand, is limited by the random read and write performance of the big table index, and it is difficult to quickly locate the query object and its copy storage location by query object name, as shown in (Figure 3).

In a distributed storage system such as the one shown on the left side of (Figure 3) a writing task or a reading task can be disassembled into the pattern shown on the right side of (Figure 3) with the two main bottlenecks described above. The next sections of this paper will focus on how to optimize or avoid these two performance bottlenecks.

2.2. Storage strategy of CRUSH location algorithm

To solve the spatial object location bottleneck mentioned in Section 1, this paper provides a high-performance remote image reading and writing method for locating the storage location of gridded objects based on their names to break through the random read and
write performance bottleneck of big table indexes. The overall idea of the method comes from the CRUSH algorithm provided by the Ceph storage system that determines how to store and retrieve data by calculating the data storage location. How to use the CRUSH algorithm to realize the large-scale storage of remote sensing images for querying and reading in the cloud environment has become the core content of the research study in this method.

Among other things, Ceph implements a controlled replication under scalable hashing (CRUSH) algorithm, described as follows: the back-end of Ceph is a reliable, autonomic distributed object store (RADOS), and all the data are divided into several objects according to their size, with each object having an object id (Oid) (Weil et al., 2006; Weil, Brandt, Miller, & Maltzahn, 2006; Zhang, Gaddam, & Chronopoulos, 2015). With this Oid, a list of nodes (similar to Region1, Region2, Region3) is obtained by the client executing the CRUSH algorithm. Then, they are filtered to find the storage nodes that meet the requirements based on the number of copies. After finding which object storage devices (OSDs) the data should be stored on, the client then initiates an IO request to the target OSD, which stores the object according to the request. The specific purpose of CRUSH is that it maintains a CRUSH map (map) describing the current state of the cluster resources according to certain rules used for Oid in the map to perform a hash-like computation to obtain a list of OSDs. This study uses the CRUSH-HASH method to calculate a random number for each OSD, multiplying that random number by the weight of the corresponding OSD and saving the object in the OSD corresponding to the largest value of the result. A dedicated class hash algorithm can also be defined in the implementation according to other application requirements.

Therefore, in the storage process of massive remote sensing images, the grid codes directly related to the spatiotemporal information of remote sensing image grids can be used as data indexes, which, on the one hand, makes the gridded image evenly distributed in the cluster, avoids the accumulation of remote sensing images of similar time and space on the same server, and avoids the problem of local hotspots in the cluster; on the other hand, it establishes the correlation between the spatiotemporal information of the grid and the
storage location, and can directly determine the grid codes involved in spatiotemporal retrieval through remote sensing image metadata and further directly calculate the storage location of the corresponding grid to achieve high-performance distributed random read/write positioning.

Algorithm 1  A pseudo-code example of CRUSH algorithm
\begin{verbatim}
Input: ObjectID
Output: OSDList:[OSD1, OSD2, OSD3,...] result = empty_list
for num_reps do
    bucket = take; item = random_pick_from_list(ObjectID, in bucket);
    if item.type not failure_domain then
        in_bucket.append(item);
    if item not in result then
        result[rep] = item;
    end if
end for
return result
\end{verbatim}

In summary, this study designs a method based on the Ceph storage system. The method uses the CRUSH algorithm to determine the storage location of data objects to break the performance bottleneck of random read/write. As long as the Oid corresponding to the image to be queried can be obtained, the storage location can be calculated by the CRUSH strategy to achieve linear time consumption for the location of gridded image objects.

2.3. Grid metadata management strategy

With the storage strategy of the CRUSH location algorithm, the problem of multi-conditional query in remote sensing image storage is transformed into the problem of how to determine the data Oid based on the query conditions. In this paper, a storage strategy based on the metadata of gridded images is proposed to solve this problem.

A gridded image in a spatial database is a data organization that represents spatial features in regular arrays, with each datum in the organization representing observed spectral, radar, and other non-geometric attribute features. Grid metadata, on the other hand, is a description of this spatial data structure and attributes, which involves information on all aspects of gridded data production, storage management, distribution, and data characteristics. It can be noted that when conducting a multi-conditional spatiotemporal query of a remote sensing image, the actual query operation only involves metadata such as the geo-referenced information of the relevant remote sensing image and does not require the identification of the remote sensing image content itself. The metadata management strategy provided in this paper is organized in an object-based relational model. The strategy stores the gridded images and the grid metadata with two objects, respectively, and stores the grid metadata exclusively within the meta database of the cluster. This meta database will not only help the database to organize and manage spatial information but also help users to query the required spatial information and participate in spatial operations in spatiotemporal queries.

The storage strategy is based on the metadata of gridded images provided in this paper; before storing remote sensing images, the strategy first parses the remote sensing image files to obtain the metadata of the gridded images, including spatiotemporal information, image quality, etc. Among them, the georeferencing information and metadata of gridded images may be stored in other small text files attached to them, and it is necessary to reconstruct the
reference information format uniformly, list all the descriptive data item by item, represent them as a reference information dictionary, and obtain the formatted metadata that meet the objectification requirements. Through the above parsing process, a set of key-value pairs can be obtained, where key is the attribute name of the attribute information and value is the corresponding data content. When storing gridded images, remote sensing image objects are uploaded to Ceph object gateway through RESTful API using remote sensing image grid code as the unique identifier required by the API, using the grid code with spatiotemporal information and other data descriptions as Oid through a CRUSH algorithm, and performing hash-like calculations of Oid in the map according to certain rules to determine the storage location of data entities within the cluster. At the same time, the corresponding grid metadata are stored in the metadatabase in the cluster.

When a multi-conditional spatiotemporal query is needed to find the gridded image, the back-end data service, after receiving the query request initiated by the client, first retrieves all the remote-sensing image data involved in this query according to the retrieval conditions and spatiotemporal range in the meta database, and each gridded image datum is identified by a unique identifier (Oid). Subsequently, using the CRUSH algorithm, it calculates the storage locations of remote sensing images that meet the query conditions in the cluster, returns the metadata of the gridded images involved to the client, and then the client makes a read request of the storage locations of remote sensing images in the cluster according to the application requirements, and requests the required data from the back-end, as shown in (Figure 4).

Through this strategy, the cluster can prioritize the query in the lightweight and small grid metadata database after receiving the spatial query and spatial calculation commands without involving the huge gridded image ontology, and can directly determine the involved gridded object Oid through the retrieval conditions, breaking the performance bottleneck of spatial object retrieval, especially multi-conditional spatiotemporal retrieval.

There are three advantages to this design: first, distinguishing remote sensing images by data product number and spatiotemporal grid, ensuring that the unique object of the database has a unique Oid, which guarantees the correct database logic; second, by providing powerful query flexibility, considering the application demand of querying remote sensing data by range, and filtering the Oid corresponding to the grid code, it is possible to quickly attain spatiotemporal, multi-conditional data, meaning that, thirdly, the gridded images are distributed as evenly as possible among all the nodes of the cluster. The CRUSH algorithm will distribute the images evenly among all nodes according to the Oid as far as possible so that the hard disk and computing resources of all nodes can be fully utilized when making complex spatiotemporal queries to prevent the phenomenon of hotspots.

3. Experimental study

To evaluate the effectiveness of the metadata combined with the CRUSH algorithm storage strategy proposed in this paper, we conducted a comparison experiment between this strategy and other common HBase-based storage strategies. This section describes the experimental environment and experimental data of the comparison experiments. Then, the experimental results are analyzed and evaluated.
3.1. Experimental environment and data

In this experiment, the metadata management of gridded images based on Ceph and the IO performance of the hash-like positioning algorithm for data objects were tested. For the HBase cluster, the native Apache Hadoop, Apache HBase, and the Hadoop ecology CDH (Cloudera distribution including Apache Hadoop) are used to test the performance of the HBase storage strategy when building the relevant test cluster. To solve the connectivity problem of different programming languages in the HBase ecosystem and extend the application scope, the HBase ThriftServer service was also enabled in this experiment. Ceph cluster construction was chosen to install and test on the stable version 10.2.11 of Ceph distribution, and the availability of the storage system was briefly tested on a small cluster. In the Ceph cluster, to address network communication and improve user availability, the Amazon S3 service port was deployed to enable users to access data through the standard S3 API format. The specific configuration of the experimental environment is shown in Table 1. To exclude accidental errors such as network fluctuations, all data in this experiment are taken as the average result of multiple tests.

The data used in this experiment are from the image data of GaoFen-1, with a total number of 13,831 gridded images and a total size of about 75 Gb. The read and write performance tests are performed at multiple levels with 4,000, 7,000, and 10,000 gridded images in the cluster to verify the performance of the storage policy under different storage sizes. The test operations include both writes and spatiotemporal queries, and the number of gridded images involved in each operation ranges from 1 to 1,000.

3.2. Performance comparison results and analysis

To better analyze the data writing performance of the Ceph-based remote sensing image storage strategy, this experiment conducted 3,000 random writes and compared the writing time of the Ceph cluster and HBase cluster at different database sizes. Since there is an obvious correlation between write time and the amount of data written, this comparison graph is drawn after fitting the experimental results to the data. The write performances when the
4. Discussion

4.1. Analysis of results

The experimental results show that, in terms of data fluctuations, the Ceph cluster has fewer time-consuming fluctuations, reflecting the relative stability of the strategy. In terms of overall performance, the performance of the Ceph cluster degrades slower than that of the HBase cluster as the data volume increases. When the grid of storage in the cluster is 7,000–10,000, the read performance and write performance of the Ceph cluster is higher than that of the HBase cluster, which can take advantage of the Ceph cluster storage strategy and significantly reduce the data read and write overhead. However, as the amount of data in the cluster decreases, the performance of the Ceph cluster begins to decline relatively, and in some cases is even lower than the HBase cluster. We enumerate the specific reasons for this below:

(1) The advantages brought by the Ceph cluster storage strategy are overshadowed by the additional network communication overhead of the gateways. As the amount of data stored in the cluster decreases, the proportion of time spent on object location, reading, and writing gradually shrinks, and the proportion of time spent

| Parameters       | Value          |
|------------------|----------------|
| Number of nodes  | 3              |
| Operating System | Ubuntu16.04    |
| Memory           | 32 G           |
| Number of CPU    | 4              |
| Hadoop Version   | 3.1.2          |
| HBase Version    | 2.2.2          |
| Ceph Version     | 10.2.11        |
on inter-cluster gateway communication gradually increases, and the advantages of object location reading and writing in terms of time spent gradually cancel out, resulting in lower-than-expected performance.

(2) The metadata management strategy performs the encapsulation operation and object parsing, which can substantially optimize the database retrieval operation when the data scale is larger and the structure is more complex and suitable for the object-oriented processing of the computer itself. However, when the scale becomes smaller, the additional time cost of object encapsulation may be nonnegligible.

This result proves that the Ceph and metadata management-based storage strategy has higher stability than the big table index storage strategy. When large-scale gridded image storage is necessary, the strategy proposed in this paper is a better choice. Additionally, the larger the scale of the grid storage, the better the effect of the strategy proposed in this paper.
This experiment compared the performance difference between the proposed storage strategy and the traditional large table index storage strategy in the distributed storage of gridded remote sensing data and focused on the comparison of the performance of the two strategies in spatiotemporal query and random reading and writing. However, there are still some problems that need to be further tested and solved in future research. This section presents the limitations of the study and puts forward some directions for future research:

1. There is no in-depth comparative experiment on the two storage strategies proposed in this paper, and the performance of the two strategies in the storage process is tested separately. Future work will design experiments to test the performance when only one of the strategies is used.

Figure 6. Inquiry time differences between nodes in the different storage strategy scenarios.

4.2. Deficiency and prospects

(a) Relationship between grid size and object query time when there are 4,000 grids in the cluster.
(b) Relationship between grid size and object query time when there are 7,000 grids in the cluster.
(c) Relationship between grid size and object query time when there are 10,000 grids in the cluster.
(d) Average query time with the number of storage grids in the cluster.
(2) There is no detailed test on the cost of objectification. The current experiments show that the additional cost brought by objectification is less than the performance improvement brought by the strategy, but it is still necessary to control and test the performance cost of objectification in detail to clarify the scope of the application of the strategy.

(3) There is no performance test in different size clusters, and the current results were only obtained with a three-node cluster. Experiments will be conducted with larger clusters in the future.

5. Conclusions

When storing gridded images in a large-scale cluster using a distributed storage system, the traditional big table index storage strategy is inefficient. To improve the read and write performance of distributed gridded image storage, this paper analyzes the reasons for the poor performance of the traditional big table index storage strategy and proposes a storage strategy based on Ceph and metadata management. This research is of great importance due to the increasing demand for distributed gridded image storage in clustered environments due to the increasing data volume, complexity, and distributed computing applications of gridded images. The method encapsulates remote sensing images in the form of objects through metadata management strategies to achieve spatiotemporal retrieval of gridded images. The clustered location of gridded images is achieved through hash-like calculation. The method can effectively achieve spatial operation support in the clustered database, and at the same time realize fast random read and write of the gridded image objects. Random write and spatial query experiments prove the feasibility, effectiveness, and stability of the strategy. In the experiments, the existing geospatial tool Geospatial Data Abstraction Library (GDAL) is used in this study and combined with the Ceph platform to achieve the desired functionality. The strategy outperforms the traditional indexing strategy for big table storage in both read and write performance when storing larger amounts of data. However, the approach does not show significant superiority over the traditional scheme when the number of stored remote sensing images is small, so further work will be carried out on objectified cost control in future work.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was funded by the Department of Science and Technology of Henan Province through grant 201400210100 and the National Key R&D Program of China through grant 2019YFE0127000. And this work was supported by National Supercomputing Center in Zhengzhou.
Notes on contributors

**Xinyu Tang** is an M.S. student in resource and environmental science with a concentration in GIS Development at China Agricultural University. He earned his B.A. degree in remote sensing science and technology from Wuhan University, China. He is currently conducting research in remote sensing data store.

**Xiaochuang Yao** is a lecturer in College of Land Science and Technology, China Agricultural University. His primary research interests include spatial cloud computing and spatio-temporal big data.

**Diyou Liu** has just obtained a doctorate in engineering from the School of Land Science and Technology of China Agricultural University in June 2021, and is about to start his postdoctoral work at China Agricultural University.

**Li Li** is an associate professor in College of Land Science and Technology, China Agricultural University. She is currently conducting research in microwave remote sensing and its applications.

**Dehai Zhu** is a professor in College of Land Science and Technology, China Agricultural University. He is the vice chairman of the China Geographic Information Industry Association’s Spatial Big Data Technology and Application Working Committee.
Guoqing Li (Senior Member, IEEE) is a professor in Aerospace Information Research Institute, Chinese Academy of Sciences and the Director of National Earth Observation Data Center (NODA). His research interests include high-performance geo-computation, big Earth data management, and spaceborne disaster mitigation.

Long Zhao is a PhD candidate in Aerospace Information Research Institute, Chinese Academy of Sciences. He earned a Bachelor of Science degree in cartography and geographical information system from China Agricultural University. He is currently conducting research in discrete global grid system and spatial cloud computing.

ORCID
Xiaochuang Yao http://orcid.org/0000-0001-8068-9415
Guoqing Li http://orcid.org/0000-0003-0654-5426

Data availability statement
Data used in this paper are available upon request from the corresponding author. http://gaofenplatform.com/.

References
Aghayev, A., Weil, S., Kuchnik, M., Nelson, M., Ganger, G. R., & Amvrosiadi, G. (2019). File systems unfit as distributed storage backends: Lessons from 10 years of ceph evolution. In Proceedings of the 27th acm symposium on operating systems principles, New York, NY, USA, (pp. 353–369).
Arafa, Y., Barai, A., Zheng, M., & Badawy, A.-H. A. (2018). Evaluating the fault tolerance performance of hdfs and ceph. In Proceedings of the practice and experience on advanced research computing, New York, NY, USA, (pp. 1–3).
Barclay, T., Gray, J., & Slutz, D. (2000). Microsoft terraserver: A spatial data warehouse. In Proceedings of the 2000 acm sigmod international conference on management of data, New York, NY, USA, (pp. 307–318).
Benet, J. (2014). Ipfs-content addressed, versioned, p2p file system. arXiv Preprint, arXiv, 1407.3561.
Borthakur, D. (2008). HDFS architecture guide. Hadoop Apache Project, 53(1–13), 2.
Carstoiu, D., Cernian, A., & Olteanu, A. (2010). Hadoop hbase-0.20.2 performance evaluation. In 4th international conference on new trends in information science and service science, Gyeongju, Korea (South), (pp. 84–87).
Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., ... Gruber, R. E. (2008). Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), 26(2), 1–26.
Cheng, L., Kotoulas, S., Ward, T. E., & Theodoropoulos, G. (2014). Robust and efficient large-large table outer joins on distributed infrastructures. In European conference on parallel processing, Switzerland, (pp. 258–269).
Davies, A., & Fisk, H. (2006). MySQL clustering. Indianapolis, Indiana, USA: Sams Publishing.

Galić, Z., Mešković, E., & Osmanović, D. (2017). Distributed processing of big mobility data as spatio-temporal data streams. Geoinformatica, 21(2), 263–291.

Hajjaji, Y., & Farah, I. R. (2018). Performance investigation of selected NoSQL databases for massive remote sensing image data storage. In 2018 4th international conference on advanced technologies for signal and image processing (atsip), Sousse, Tunisia, (pp. 1–6).

Hu, F., Yang, C., Jiang, Y., Li, Y., Song, W., Duffy, D. Q., … Lee, T. (2020). A hierarchical indexing strategy for optimizing apache spark with hdfs to efficiently query big geospatial raster data. International Journal of Digital Earth, 13(3), 410–428.

Jhummarwala, A., Potdar, M., & Chauhan, P. (2014). Parallel and distributed GIS for processing geo-data: An overview. International Journal of Computer Applications, 106(16), 18602–19881.

Jing, W., & Tian, D. (2018). An improved distributed storage and query for remote sensing data. Procedia Computer Science, 129, 238–247.

Li, J., & Narayanan, R. M. (2004). Integrated information mining and image retrieval in remote sensing. Recent Advances in Hyperspectral Signal and Image Processing, 1, 449–478.

Li, Q., Lu, Y., Gong, X., & Zhang, J. (2014). Optimizational method of hbase multidimensional data query based on hilbert space-filling curve. In 2014 ninth international conference on p2p, parallel, grid, cloud and internet computing, Guangdong, China, (pp. 469–474).

Liu, X., Hao, L., & Yang, W. (2019). Bigeo: A foundational paas framework for efficient storage, visualization, management, analysis, service, and migration of geospatial big dataa case study of Sichuan province, China. ISPRS International Journal of Geo-Information, 8(10), 449.

Lv, Z., Li, X., Lv, H., & Xiu, W. (2019). Bim big data storage in webvrgs. IEEE Transactions on Industrial Informatics, 16(4), 2566–2573.

Ma, Y., Li, G., Yao, X., Cao, Q., Zhao, L., Wang, S., & Zhang, L. (2021). A precision evaluation index system for remote sensing data sampling based on hexagonal discrete grids. ISPRS International Journal of Geo-Information, 10(3), 194.

Martinho, N., Almeida, J.-P. D., Simões, N. E., & Sá-Marques, A. (2020). Urbanwater: Integrating epanet 2 in a postgresql/postgis-based geospatial database management system. ISPRS International Journal of Geo-Information, 9(11), 613.

Nishimura, S., Das, S., Agrawal, D., & El Abbadi, A. (2011). Md-hbase: A scalable multidimensional data infrastructure for location aware services. In 2011 ieee 12th international conference on mobile data management, Lulea, Sweden, (Vol.1, pp. 7–16).

Rajak, R., Raveendran, D., Bh, M. C., & Medasani, S. S. (2015). High resolution satellite image processing using Hadoop framework. In 2015 ieee international conference on cloud computing in emerging markets (ccem), Bangalore, India, (pp. 16–21).

Salanio, K. A. E., Santos, C., Magturo, R., Quevedo, G. P., Virtucio, K., Langga, K., … Paringit, E. (2015). Development of data archiving and distribution system for the Philippines’ lidar program using object storage systems. In Free and open source software for geospatial (foss4g) conference proceedings, Seoul, South Korea, (Vol. 15, p. 53).

Wang, L., Cheng, C., Wu, S., Wu, F., & Teng, W. (2015). Massive remote sensing image data management based on hbase and geosot. In 2015 ieee international geoscience and remote sensing symposium (igrrss) (pp. 4558–4561).

Wang, S., Zhong, Y., & Wang, E. (2019). An integrated GIS platform architecture for spatiotemporal big data. Future Generation Computer Systems, 94, 160–172.

Wang, X., Wang, R., Zhan, W., Yang, B., Li, L., Chen, F., & Meng, L. (2020). A storage method for remote sensing images based on google s2. IEEE Access, 8, 74943–74956.

Wang, X., Zhang, H., Zhao, J., Lin, Q., Zhou, Y., & Li, J. (2015). An interactive web-based analysis framework for remote sensing cloud computing. In ISPRS annals of photogrammetry, remote sensing and spatial information sciences, Milan, Italy, (pp. 43–50). IEEE.

Wang, Y., & Wang, S. (2010). Research and implementation on spatial data storage and operation based on Hadoop platform. In 2010 second iita international conference on geoscience and remote sensing, Qingdao, China, (Vol.2, pp. 275–278).
Weil, S., Brandt, S., Miller, E., & Maltzahn, C. (2006). Crush: Controlled, scalable, decentralized placement of replicated data. In *Sc’06: Proceedings of the 2006 acm/ieee conference on supercomputing, Tampa, FL, USA*, (pp. 31–31).

Weil, S. A., Brandt, S. A., Miller, E. L., Long, D. D., & Maltzahn, C. (2006). Ceph: A scalable, high-performance distributed file system. In *Proceedings of the 7th symposium on operating systems design and implementation, Berkeley, CA, USA*, (pp. 307–320).

Weipeng, J., Dongxue, T., Guangsheng, C., & Yiyuan, L. (2018). Research on improved method of storage and query of large-scale remote sensing images. *Journal of Database Management (JDM)*, 29(3), 1–16.

Xu, C., Du, X., Yan, Z., & Fan, X. (2020). Scienceearth: A big data platform for remote sensing data processing. *Remote Sensing*, 12(4), 607.

Yan, S., Yao, X., Zhu, D., Liu, D., Zhang, L., Yu, G., ... Yun, W. (2021). Large-scale crop mapping from multi-source optical satellite imageries using machine learning with discrete grids. *International Journal of Applied Earth Observation and Geoinformation*, 103, 102485.

Yao, X., Li, G., Xia, J., Ben, J., Cao, Q., Zhao, L., ... Zhu, D. (2020). Enabling the big earth observation data via cloud computing and dGGS: Opportunities and challenges. *Remote Sensing*, 12(1), 62.

Zeiler, M. (1999). *Modeling our world: The esri guide to geodatabase design, Redlands, California, USA*, (Vol. 40). ESRI.

Zhang, L., Sun, J., Su, S., Liu, Q., & Liu, J. (2020). Uncertainty modeling of object-oriented biomedical information in hbase. *IEEE Access*, 8, 51219–51229.

Zhang, X., Gaddam, S., & Chronopoulos, A. (2015). Ceph distributed file system benchmarks on an openstack cloud. In *2015 ieee international conference on cloud computing in emerging markets (ccem), Bangalore, India*, (pp. 113–120).

Zhou, H., Chen, X., Wu, X., Dai, W., He, F., & Zhou, Z. (2019, CN110491478-A 22 Nov 2019 G16H-030/20 201998 Pages: 8 Chinese). *Ceph based image file distributed storage system has metadata storage module which is configured to store related information of image file, including unique identification and metadata information of image file*. (IDS Number: 2019A0226F)