Multimedia Tools and Applications (2024) 83:59861–59886
https://doi.org/10.1007/s11042-023-17858-6

Research on remote sensing image storage management and a fast visualization system based on cloud computing technology

Lichun Yang[1,2] · Weibing He[3] · Xiaoyong Qiang[3] · Jinjun Zheng[3] · Fang Huang[3]

Received: 26 October 2022 / Revised: 24 September 2023 / Accepted: 13 December 2023 /
Published online: 2 January 2024

Abstract
With the continuous development of remote sensing technology, the data volume of remote sensing images has increased exponentially, resulting in many difficulties in the storage, management, transmission, calculation, and other processes of remote sensing images. In order to solve the above problems, this paper studies the use of the Hadoop Distributed File System (HDFS) and related technologies to design and implement a browser/server (B/S) architecture for a massive, multisource, remote sensing images distributed storage management system. The image data are stored in the HDFS, and the image metadata are stored in a MySQL database. The distributed parallel construction of the image pyramid is completed based on the Spark computing engine, and the Akka framework is used to construct WMTS (Web Map Tile Service) to realize the release of remote sensing images. Finally, the rapid visual display of remote sensing images is carried out using Leaflet. The system also supports image data management, image target detection, user management, and other functions. After testing, this system can support the storage and management of multisource remote sensing image data, and can solve perfectly the problems of insufficient storage space and insufficient computing power of a single server. It is found that the upload and download speeds of a large amount of remote sensing images can be close to the maximum speed of a gigabit local area network (LAN). In the gigabit LAN environment, the average upload speed of a single remote sensing image is 97.74 MB/s, and the average download speed is 87.62 MB/s. In terms of image pyramid construction, the speed of a multi-node parallel construction based on Spark is two times higher than that of a single-node construction. Additionally, compared to similar systems, this system has better data transmission and retrieval speed, better data computing ability, and higher concurrency processing ability.

Keywords Remote sensing image · Storage management system · Hadoop · Distributed storage · Distributed computing

Extended author information available on the last page of the article

1 Introduction

Remote sensing images are widely used in resource exploration, environmental assessment, land management, national defense, and military fields because they can obtain a wide range of ground information in a short time [1]. With the continuous development of remote sensing technology, the imaging resolution and spectral resolution of remote sensing images are becoming increasingly high, which has made the data volume of remote sensing images increase exponentially [2], resulting in many difficulties in the storage, management, transmission, calculation, and other processes of remote sensing images. How to store uniformly, manage, and share remote sensing image data have always been the research focus in this field [3]. File system management is the most traditional way to manage remote sensing images; that is, images are managed by independent files. However, this method is inefficient and mainly depends on human management. It is prone to data redundancy, confusion, and loss, and can only be read and written by a single user. It is not suitable for occasions with high concurrency requirements [4]. In response to the above problems, some researchers have specially developed image data management systems for remote sensing images. The design goal is to achieve efficient spatial database management and access [5]; that is, to facilitate user data sharing and achieve rapid retrieval and access to remote sensing data resources.

In recent years, with the rapid development of remote sensing technology, remote sensing images obtained through remote sensing satellites, unmanned aerial vehicles (UAVs), and other channels have the characteristics of more and more bands, higher and higher imaging resolution, shorter and shorter data acquisition cycles, and the data volume has increased sharply, which can easily reach the TB or even PB level [6]. Such a huge amount of data will lead to many restrictions in data storage, data transmission, data processing, and other aspects of remote sensing images, such as the disk size, memory size, computing power of the server, network bandwidth restrictions between the server and the client, and so on. However, the traditional data storage and computing models have drawbacks such as poor stability, poor sharing ability, poor scalability, high costs, and low computational efficiency, making them unable to cope with the storage challenges brought about by rapid data growth [7].

According to our literature review, the main way to solve the above problems is to use a distributed storage system [3, 8–19], such as HDFS. After analyzing the research on systems using this type of technology, it was found that: (1) distributed file systems are mostly used as storage media for remote sensing images, and different indexing methods are needed to organize data. However, performance degradation may occur when conducting large-scale data retrieval; (2) such systems are unable to support other types of files except remote sensing images, and the system's universality is weak; (3) in building image pyramids, single node computing schemes are mostly used, without using the advantages of distributed architecture, resulting in slower computing speeds; and (4) the previous studies did not provide a good solution for the rapid visualization needs of image pyramids.

To resolve these problems, this research studies a remote sensing image storage and management system that integrates the functionalities of storage, management, retrieval, calculation, and fast visualization using the Hadoop distributed framework and related components. Here, we named this system the Massive Multi-source Remote sensing Images Distributed Storage and Management System (MRSI-DSMS). In this system's design, the HDFS and MySQL are used to realize the storage, management, and retrieval of remote sensing image entity data and metadata. By utilizing Hilbert spatial indexing,

data are stored in separate tables, improving the speed of spatial retrieval. The master–slave replication architecture of the MySQL database allows us to separate the data reading and writing functions, thus improving the system's concurrent access capability. Based on a Spark cluster and the Geotrellis computing framework, an image pyramid is constructed in parallel. The Web Map Tile Service (WMTS) service using Akka is constructed which is combined with the Leaflet framework to realizes the rapid visualization of large data remote sensing images. Compared to previous research, the main contributions of this study are as follows.

(1) The system can run on a cluster composed of computers with low configuration. By virtue of the advantages of the Hadoop ecosystem, it can dynamically expand the number of nodes, and solves problems such as poor scalability of the storage capacity, easy failure of single nodes, and performance bottlenecks in traditional centralized storage.
(2) By storing remote sensing images and metadata information in HDFS and MySQL, respectively, the occupation of directory space by small files is reduced and the data retrieval delay is reduced while the storage efficiency is improved.
(3) The use of Hilbert curves to construct spatial indexes and store data in different data tables according to spatial positions greatly improves the speed of data retrieval. MySQL, which uses master–slave replication, is used to achieve data reading/writing separation and improve the system's high concurrency processing capability.
(4) The distributed construction of the image pyramid by the Spark cluster greatly improves the construction speed and makes full use of the computing resources of each node.
(5) The WMTS service built with the Akka framework makes it possible to visualize remote sensing images on the Web quickly, greatly reducing the network bandwidth required for image transmission and improving the user experience.

Compared with similar studies, the advantages of this system are: (1) the data types supported by this system are not limited to remote sensing image data, but also based on custom metadata files to support drone image data and other third-party types of data, which can meet the various needs of users as much as possible. Other similar studies only support remote sensing image data and do not provide support solutions for other types of data. (2) This system is based on the position information of images and utilizes Hilbert curves to construct spatial indexes. The data are stored in different data tables according to spatial positions, enabling rapid filtering of nontarget information based on spatial position information during remote sensing image retrieval, greatly improving the speed of data retrieval. Although other similar studies have similar designs, they are mostly based on Hash functions and do not utilize the spatial information of the image itself. (3) This study establishes a spatial index to store image data in adjacent hard disk spaces in the same area, ensuring the efficiency of the system's data reading and improving retrieval speed. (4) In terms of data processing, because the processing of remote sensing images involves a large amount of image computation and consumes a large amount of computing resources, most related research adopted a single machine processing mode, making it difficult for these systems to meet various requirements. This study utilizes Spark to construct corresponding computing tasks and allocate them to multiple nodes in the cluster. The reading and writing of data are completed based on the Hadoop framework, which effectively continues the distributed design characteristics of this system, ensuring that data reading, writing, and processing are distributed, reducing the computational pressure on a single node and improving computing speed. Finally, (5) regarding the rapid visualization of images, the WMTS

service constructed in this study can also perform real-time rendering of image pyramid data at different levels in Hadoop as needed, and finally transmit it to users through the network. Compared to the schemes of directly transmitting the entire image used in related studies, our method greatly reduces the network transmission burden between the client and server.

The remainder of this paper is organized as follows. Section 2 gives a brief introduction to related work on how to solve the storage problem of massive remote sensing images with distributed storage systems. Section 3 introduces some key technologies in the system design, such as HDFS, MySQL, HBase, Spark, Geotrellis, Leaflet, etc., for our proposed MRSI-DSMS. Thus, in Sections 4 and 5, we concentrate on the design and realization of the system. In Section 6, we perform validation experiments and discuss the experimental results. Finally, Section 7 draws some conclusions and points out future directions.

2 Related work

In order to solve the storage problem of massive remote sensing images, distributed storage systems have emerged as a powerful solution [8]. A distributed storage system can store maintenance data on multiple nodes in the network to provide faster data access ability, and a small number of node failures will not have a negative impact on the stability of the entire system [9]. This provides us with faster data access speeds, higher fault tolerance, and more flexible scalability, providing new ways to address the challenges of storage of massive remote sensing data [10]. Nowadays, popular distributed data storage systems include Lustre, HDFS, GlusterFS, Ceph, SeaweedFS, etc. [3]. Many researchers have used the aforementioned systems to store remote sensing image data. For instance, Zheng et al. [11] used Hadoop for distributed storage of vector spatial data, effectively improving scalability and avoiding the defect of single node failure. Zhong et al. [12] introduced a key-value distributed storage model for managing massive image data, which stores small documents as large data files. Rajak et al. [13] utilized image pyramid technology to store image data in HBase, improving data management efficiency.

However, although using a distributed file system can solve the storage problem of massive data, there are still some challenges to overcome, such as the performance issues encountered in metadata retrieval and spatial location retrieval for massive remote sensing images, effective data access and service provision issues, efficient image computing issues, and even system fault tolerance and reliability issues.

In response to the problem of low performance in massive remote sensing image retrieval, Zhu et al. [14] used a PostgreSQL database and an HDFS distributed file system as the underlying storage system, and designed the Remote Sensing Image Management and Scheduling System (RSIMSS) based on a multi-layer Hilbert spatial index and an image tile pyramid to organize massive remote sensing image datasets, improving the retrieval performance of the system. Zhou et al. [15] utilized the Ceph distributed object storage system and a multi-level Hilbert spatial index to construct three sub-modules: RSIAPI, RSIMeta, and RSIData, forming the Remote Sensing Image Management System (RSIMS). They also created a geospatial data abstraction library (GDAL) compatible with input/output (I/O) interfaces, which provides data retrieval and access services, and solves the problems of data access and service provision. Regarding image computing issues, Wang et al. [16] integrated the HDFS, MapReduce, and Orfeo toolkits to complete the storage and processing of remote sensing images. Kong et al. [17] stored image metadata,

remote sensing images, and natural resource data in Oracle databases, HDFS, and MongoDB databases, respectively. In addition, the ArcGIS Enterprise platform is also used to provide a basic environment for data calculation. In order to address the issues of the fault tolerance and reliability in the system, Wei et al. [18] proposed a strategy for organizing and managing massive remote sensing tile data based on spatial databases. The data were segmented using Hash functions, and data distribution rules between multiple nodes were established to achieve multi-source heterogeneous remote sensing data management. In addition, a spatial data fragmentation method based on Hash algorithm was designed to improve the system's fault tolerance and ensure the reliability and security of distributed systems. Finally, Wang et al. [19] used the NoSQL database for remote sensing image management, built a distributed cluster structure based on the MongoDB database, and used the GridFS storage mechanism to store the original data files of the images.

In summary, the abovementioned research on massive remote sensing images management system employed distributed file systems, such as HDFS, to solve the storage problem of massive image data, and used different indexing methods to accelerate the speed of spatial retrieval. However, these methods cannot provide good solutions to address issues such as high concurrency access to the system, slow construction speed of image pyramids, and they are unable to achieve fast visualization of images. In addition, due to the wide range of remote sensing image sources, remote sensing images from different sources have metadata files of different file types and formats. Unfortunately, the above research does not provide a response method for the storage management of multi-source remote sensing images.

3 Key technologies of MRSI-DSMS

This research designs a distributed remote sensing image storage and management system based on the Hadoop ecosystem. This research scheme can be summarized as follows: (1) it uses the HDFS distributed file system to store remote sensing image entity data; (2) MySQL is used to store metadata of remote sensing images to realize image management; (3) completion of the distributed and parallel construction of the image pyramid is based on the Spark computing engine; (4) the release of remote sensing images is realized with the help of the Akka framework; and (5) Leaflet is used for the fast visual display of remote sensing images.

HDFS, HBase, Spark, and the other components mentioned above belong to the ecosystem of the Hadoop distributed framework. Hadoop is an open source distributed system framework created by the Apache Software Foundation. It focuses on using multiple computers to form a cluster to solve the storage and computing problems involved with a large amount of data. The storage and computing capacity of a single computer is limited. Based on Hadoop, multiple machines in a general configuration form a cluster, which can greatly improve the storage and computing capacity of the device.

With the increasing amount of remote sensing image data, the storage capacity of a single server often cannot meet the actual storage needs, resulting in insufficient storage space, difficult dynamic expansion of the storage space, easy data loss, slow data access, and other problems. HDFS can perfectly solve the above problems. The biggest feature of HDFS is distribution; that is, it can combine the storage space of multiple computers as a unified storage space with a greater capacity. At the same time, each file block in HDFS has multiple copies and is stored on different nodes, which greatly increases its own

fault tolerance [20]. In addition, HDFS can also increase or decrease the number of nodes according to the actual demand, and dynamically change the storage capacity.

Remote sensing image data usually contain a large amount of metadata information, which needs to be managed by the database. MySQL is one of the most widely used relational databases today, with the characteristics of crossing platform, free, and powerful functionality. It has functions such as master–slave replication, read–write separation, and database and table partitioning, which can undertake the reading and writing of massive data. Therefore, this system chooses a MySQL database to store and manage the metadata of remote sensing images.

The most common operation for processing remote sensing images is to build image pyramids. The huge amount of data in a single remote sensing image leads to the need for strong computing power in the complete computing process, where it is often difficult for single machines to meet the computing requirements [21]. As such, multi-node parallel computing can be a solution. Spark is a fast, general, and scalable big data parallel computing and analysis engine based on memory. Spark is good at processing circular and iterative data streams, and it reduces the computing unit to a resilient distributed dataset (RDD) computing model that is more suitable for parallel computing and reuse [22]. Common remote sensing image data processing, such as building image pyramids, is a typical cyclic iterative data flow process, which is very suitable for processing with Spark.

Geotrellis is a Scala language library for processing raster data based on Apache Spark [23]. It is a distributed high-performance geographic data processing engine. Geotrellis can be used to build image pyramids for remote sensing image data, and the constructed pyramid data files can be layered and rendered into portable network graphic (PNG) images, which facilitate the rapid display of large amounts of remote sensing images, and solve the problem of network bandwidth limitation between the server and client.

HBase is a distributed, scalable, column-oriented NoSQL database that supports massive data storage. Its column storage feature means it only needs to read relevant columns when executing query operations, which can significantly reduce the system I/O throughput and improve the execution speed [24]. By combining Geotrellis with Apache Spark, the constructed image pyramid data can be stored in the HBase database, improving the data accessing speed.

If the user needs to display a constructed image pyramid data in a Web page quickly, a Web map tilt service for the image should be built. Akka is a library written in Scala, which is used to simplify the writing of fault-tolerant and highly scalable actor model applications in Java and Scala [25]. The Akka framework has a simplified HTTPS service component, which can be used to establish a map service for Web access.

Leaflet is a lightweight geographic information system (GIS) front-end visual JavaScript library with high performance and strong scalability, which can easily display vector data and grid data in Web pages [26]. Using Leaflet to access the map service released by Akka can complete the rapid visual display of local remote sensing images.

4 Design scheme of MRSI-DSMS based on the Hadoop ecosystem

4.1 Overall system design

In this study, the objectives and functions that the system needs to achieve are:

(1) Remote sensing image storage and management based on the Hadoop distributed storage architecture and a MySQL database, including image storage, query, download, etc.; the storage capacity for image data needs to have the ability of smooth expansion.
(2) Support multisource image file formats, including typical remote sensing images, visible light images, infrared images, video images, and other data formats.
(3) Support the parallel construction of image pyramids and implement WMTS services to obtain the rapid visual display of remote sensing images.
(4) Support the management of user information.
(5) Provide remote sensing image application cases, e.g., UAV remote sensing images target detection based on deep learning technology.
(6) Furnish a data access interface that can be provided to third-party software.

According to the above requirements, the functional design of this system is shown in Fig. 1.

By combining the functional modules shown in Fig. 1 with the specific implementation techniques, the system function can be further divided into sub-modules, and then the entire framework of the system can be obtained, as shown in Fig. 2.

From Fig. 2, the system is mainly divided into a (1) user management module (UMM), (2) data management module (DMM), and (3) data processing module (DPM). The UMM has different functions for ordinary users and administrator users: ordinary users can only view user information, while administrator users can manage users. The DMM includes the upload and download functions of remote sensing image and

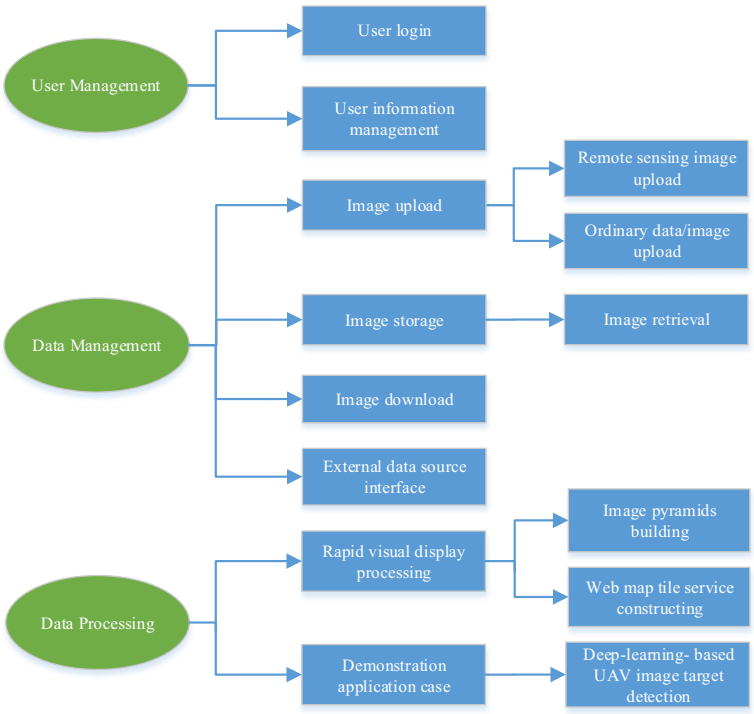

Fig. 1 System function module design

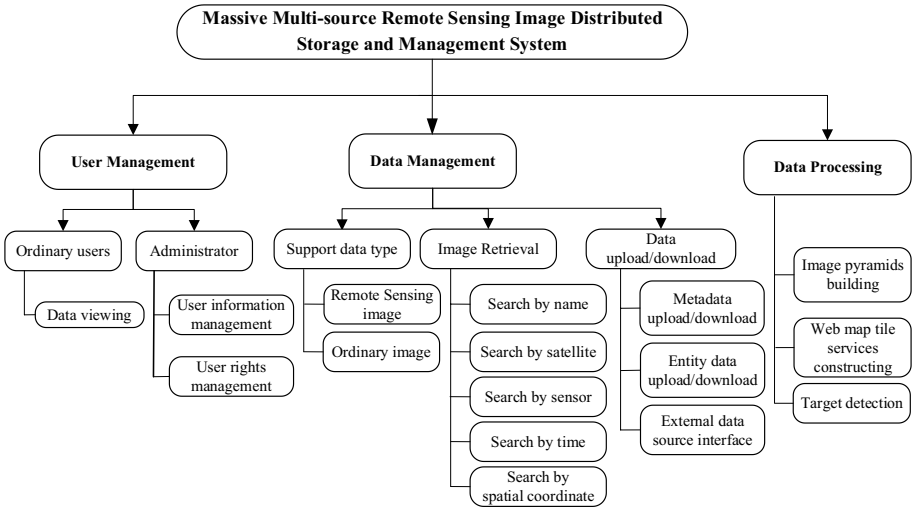

Fig. 2 System framework design

ordinary image data, and supports image queries based on retrieval conditions such as attribute information, spatial information, and time information. The DPM includes the parallel construction of the image pyramid, the release of the image pyramid, and the UAV image target detection case.

By combining Figs. 1 and 2, the operating principles and mutual logic of each module in the entire system can be obtained, as shown in Fig. 3.

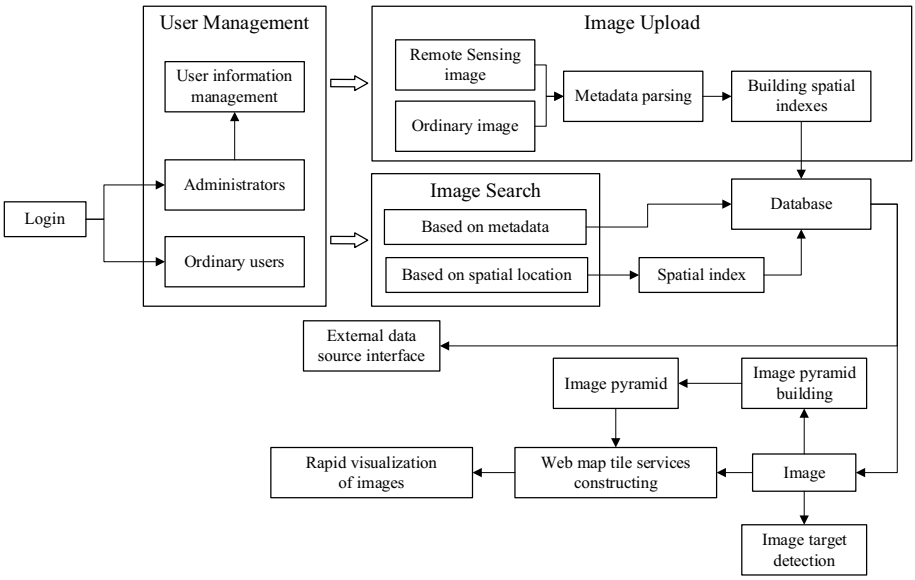

Fig. 3 System working principle and mutual logic of each module

From Fig. 3, it can be found that, starting from user login, the user types are first divided into ordinary users and administrators, where the latter can have the authority to manage user information. Image uploading can be divided into remote sensing image uploading and ordinary image uploading. The uploaded images need to go through the steps of metadata parsing, building spatial indexes, and inputting into the database. Image retrieval is divided into metadata-based retrieval and spatial-location-based retrieval. The latter requires calculating the spatial index of the image based on search criteria, and then searching based on the index. The search results can be provided to third-party software through application program interfaces (APIs). Image pyramids need to be built first. By establishing WMTS services, image pyramid data at different levels can be rendered into images of different resolution sizes, achieving rapid visualization of images. In addition, the system also provides an image target detection service for UAV remote sensing images as the demo case.

According to the above design of the system framework and functionality working principle, the architecture of the whole system has a hierarchical design, as shown in Fig. 4.

From Fig. 4, it can be seen that the whole system is based on a three-tier structure mode of a browser/server (B/S) architecture, which is divided into a (1) presentation layer, (2) business logic layer, and (3) data layer. The presentation layer is the user layer, and the specific presentation form is a Web browser, which mainly realizes the visual display of the system, the interaction between users and the background, and other functions. The main function of the business logic layer is to receive the requests transmitted by users from the presentation layer and complete the corresponding business logic. In this system, the business logic layer mainly includes a servlet program, business logic processing program, Spark program, Geotrellis program, and so on. The lowest data layer provides support for data source access in the business logic layer,

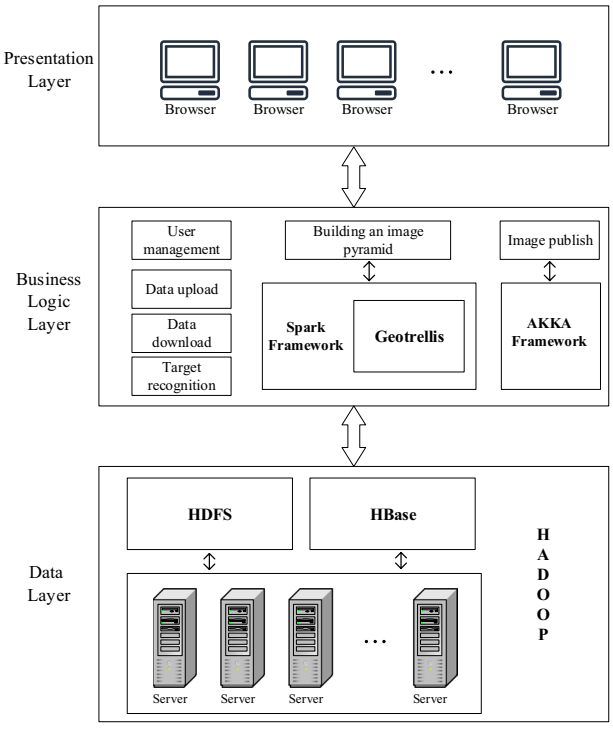

Fig. 4 System architecture diagram

including data reading and writing. In this system, the data layer is HDFS, MySQL and HBase, and all data reading and writing are completed here. The following sections will describe the design of the technical points and details of the system in sequence.

4.2 Design of the distributed storage and retrieval of massive remote sensing image data

For the management of remote sensing images, common processes include warehousing, storage, searching, and downloading. Our system also covers these functions, but there are additional issues that need to be clarified regarding specific steps when implementing the distributed storage of massive remote sensing images and fast retrieval using HDFS, as described below.

(1) In order to support multi-source remote sensing images, this system's design includes a metadata parser for parsing different types of remote sensing image metadata files. In order to support as many types of metadata files as possible, this system also provides a universal model template, allowing users to customize the metadata information of images, making it convenient for any type of image data type to be stored.
(2) Due to the design intention of this system of being able to cope with massive remote sensing image data, storing all data in one table will increase the burden on the database and result in unsatisfactory retrieval speed. In response to this issue, this study uses a Hilbert curve to divide the entire Earth plane into multiple regions, and the image data in the same region are stored in the same data table.
(3) To enable the system to cope with high concurrency access situations, this study uses the master–slave replication mode in the MySQL database, deploying the MySQL database on three nodes, forming a pattern of one master node and two slave nodes. The writing operation of data is carried out on the master node, while the reading operation is carried out on the slave node, achieving the separation of reading and writing of data and reducing the reading and writing burden of the database in the single node database mode.

Thus, after parsing the metadata information, the system will calculate the sub-region of the image based on the longitude and latitude coordinates of the four corners, and store the data in the corresponding data table of the sub-region. Additionally, the metadata files will be stored in HDFS, and the storage location will also be recorded in the database. Meanwhile, when conducting a location-based spatial search, the system will calculate the Hilbert index of the area where the images to be searched belong based on the search criteria, and then query the data from the corresponding data table according to the index. The above retrieval methods will filter out a large amount of useless data, greatly improving the efficiency of retrieval. In addition, after the storage of image metadata is completed, the system will require users to upload image entity data to HDFS and record the storage path in the database. When the users need to download a file, a simply search for the storage path of the file in the database will be operated and then the file will be downloaded. The specific image distributed storage and retrieval flowchart is shown in Fig. 5.

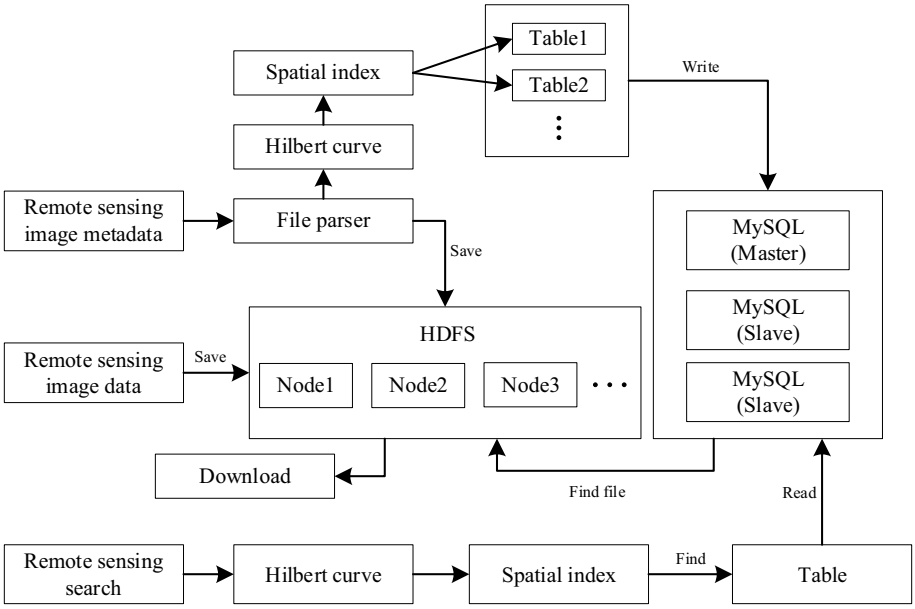

Fig. 5 Schematic diagram of distributed storage and retrieval for massive remote sensing images

4.3 Parallel construction of the image pyramid and the design of the rapid visualization display for massive remote sensing image data

Image pyramids are an important technology used for processing and managing multi-resolution remote sensing image data. An image pyramid is a data structure composed of multiple resolution levels, each of which contains different resolution versions of the original image. Due to the involvement of image operations, the construction process of the pyramid consumes a lot of computational resources. In order to accelerate the construction speed of the pyramid, this system uses Geotrellis and Spark for distributed pyramid construction, and stores the established pyramid data in HBase. Finally, the Akka framework is used to establish a WMTS service to achieve the rapid visualization of images. The parallel construction and rapid visualization display architecture of the image pyramid are shown in Fig. 6.

First, the system establishes a request based on the image pyramid and queries the MySQL database to obtain the storage location of the image in HDFS. Next, it submits a Spark task and assigns the construction process of the image pyramid to the computing nodes. Then each node reads image data from HDFS and executes according to the assigned task. After the final task is completed, the constructed pyramid data will be stored in HBase.

Regarding visualization, when the system receives an image visualization request, it will search for the corresponding level of image pyramid data in the HBase database. At the same time, using the Akka framework, a WMTS service based on the HTTPS protocol will be constructed to render the image pyramid data into images according to the request. By specifying the service's IP address and port number, the visualization effect of the image can be presented to the user.

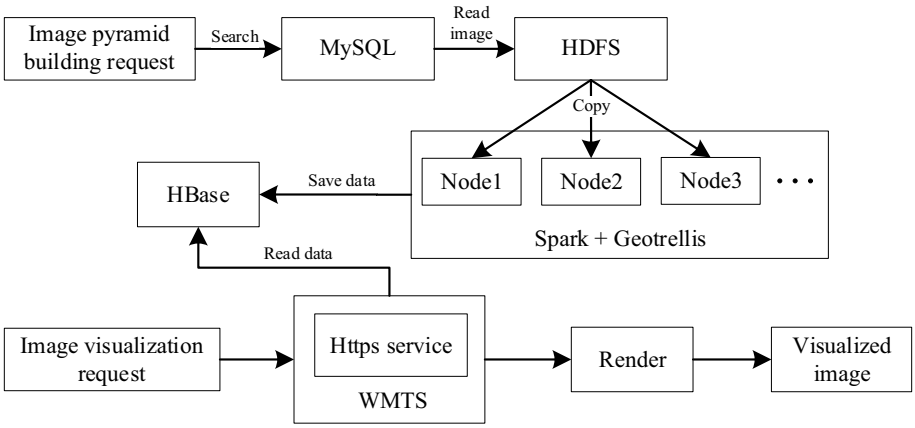

Fig. 6 Schematic diagram of parallel construction of image pyramid and rapid visualization for massive remote sensing images

4.4 System's auxiliary functionality design

Regarding the implementation of UAV image target detection based on deep learning technology, we use the improved and optimized YOLOv4 algorithm, named K-YOLOv4 [27]. We refer the reader to the mentioned research on how to optimize the algorithm and how to use it for target detection with UAV images.

The process of encapsulating a UAV target detection module into our system is demonstrated in Fig. 7. Similar to satellite remote sensing images, UAV remote sensing images are also stored in HDFS, and their metadata information is still managed by MySQL. After receiving a target detection request, the system first searches for the storage location of the image to be detected in MySQL, and then transfers it to the graphic processing unit (GPU) server. The K-YOLOv4 algorithm is deployed on the GPU server, as well as a shell script program that calls the algorithm for target detection. After the target detection is completed, the GPU server will upload the detection results to HDFS, and then send the results back to the user for display.

Additional functional design also includes user management and so on, which are relatively simple. Regarding user login, this system stores user information in the MySQL

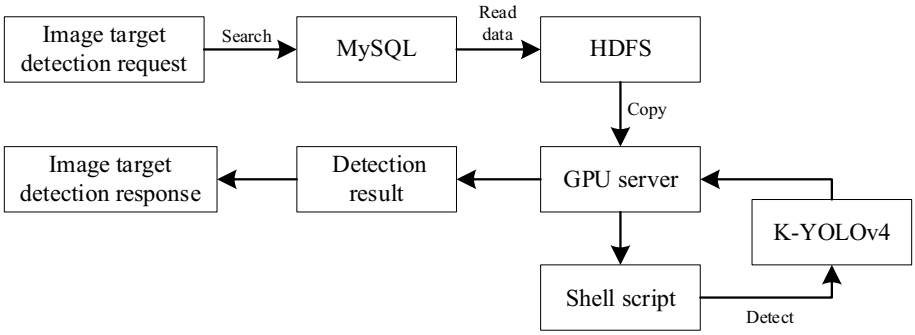

Fig. 7 Process of encapsulating the UAV target detection module into our system

database and encrypts information with higher security levels such as passwords using the MD5 algorithm to ensure data security.

5 System implementation and development

The design of this system focuses on the distributed storage and retrieval of remote sensing images, the parallel construction of image pyramids, and rapid visual display. Therefore, the implementation of modules like remote sensing image distributed storage and retrieval, the parallel construction and fast visual display of the image pyramid are described in detail, and implementation of the remaining modules is briefly described.

5.1 Remote sensing image distributed storage and retrieval

A remote sensing image is usually divided into physical image data of various bands, metadata files, and other files, of which the most important is the metadata file, which records all the metadata information of the image and is an important basis for image management and retrieval. All data files of the image are stored in the HDFS according to fixed rules, but storing metadata information in the form of files will lead to lower reading and writing efficiency. Therefore, it is necessary to parse each metadata file separately and write the parsed fields and attribute information into an image table in the HBase database. In addition, the storage location of image data in HDFS will also be written into the image table as additional metadata information. When downloading images, the system only needs to go to HBase to obtain the download path, which reduces the file-retrieval time. For the system, a class diagram of the remote sensing image distributed storage and retrieval module is shown in Fig. 8.

In the above class diagram, the *BaseDao* class and *ImageDao* class are both database access interfaces, which define the relevant methods for interacting with the database. The *BaseDao* class contains some general database operation methods, while the *ImageDao* class focuses on the database operations related to image data. The *BaseDaoImpl* class is the implementation of the *BaseDao* interface, the *ImageDaoImpl* class is the implementation of the *ImageDao* interface, and the *ImageDaoImpl* class inherits from *BaseDaoImpl* class. The *ImageMetadata* class defines an image object, including attributes such as image name, ID, type, and the corresponding get() and set() methods. The implementation of the methods in the *ImageDaoImpl* class depends on the *ImageMetadata* class. The *ImageAction* class, *ImageDownloadAction* class, *SearchAction* class, and *ImageUploadAction* class respectively realize the deletion, downloading, retrieval, and uploading of images, and the above operations are implemented by the *ImageDaoImpl* class.

The image retrieval interface of the system is shown in Fig. 9. The left half of the interface lists the retrieval conditions, such as image name, type, satellite name, etc., which the user can input and check as required. The final retrieval results will be listed below. The right half of the interface is a Web map, which is used to display the geographical location of the image in the search results, so as to facilitate the user's identification and rapid positioning.

The image uploading interface of the system is shown in Fig. 10. First, the user needs to upload the metadata file of the image and select the satellite category of the image. The system natively supports Landsat 7, Landsat 8, ZY-3, GF series, and other types of satellite data. If more types of images need to be uploaded, the system also provides a general metadata file template (see Fig. 11). Users can download, edit, and upload them by themselves. After the system successfully parses the metadata file, the user can select one or more image entity data files to upload.

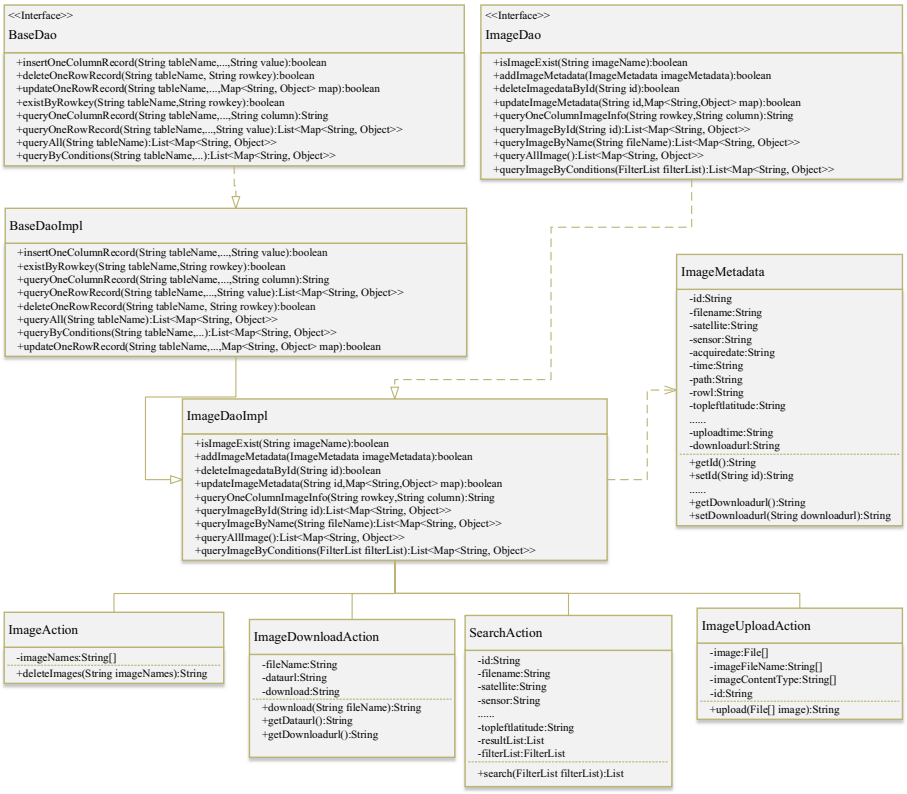

Fig. 8 Class diagram of the image distributed storage and retrieval module

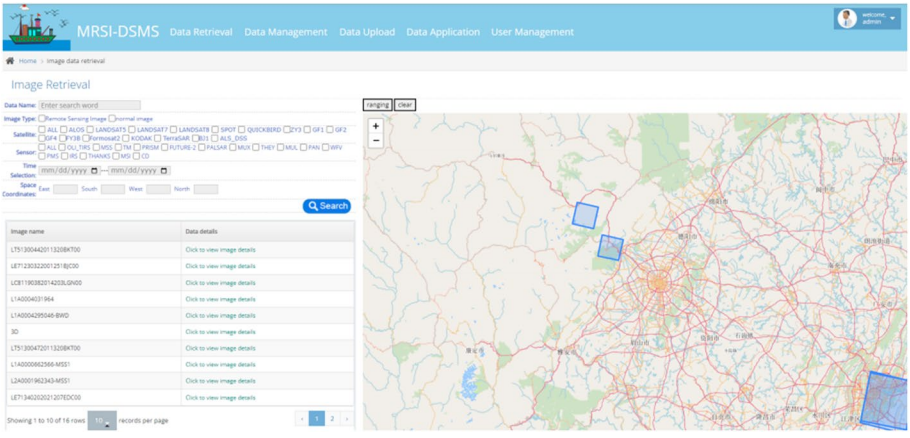

Fig. 9 Image retrieval interface

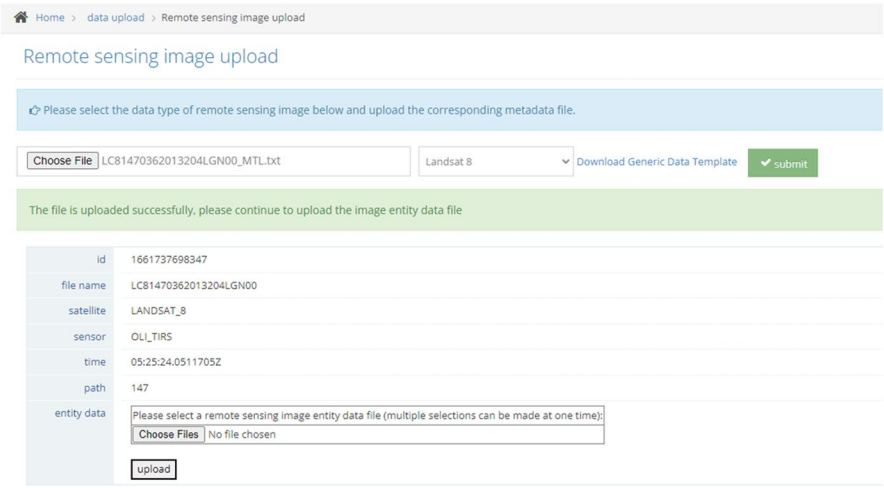

Fig. 10 Image data upload interface

```
filename=example
satellite=LANDSAT_5
sensor=TM
acquiredate=2011-11-16
time=03:28:12.3770310Z
path=130
rowl=042
TOPLEFTLATITUDE=26.91092
TOPLEFTLONGITUDE=100.77135
TOPRIGHTLATITUDE=26.86143
TOPRIGHTLONGITUDE=103.13632
BOTTOMRIGHTLATITUDE=25.00515
BOTTOMRIGHTLONGITUDE=103.07189
BOTTOMLEFTLATITUDE=103.13632
BOTTOMLEFTLONGITUDE=100.74369
/*Please modify the metadata information and save it as a txt file*/
```

Fig. 11 General Metadata file template

5.2 Parallel construction and fast visual display of the image pyramid

The parallel construction of the image pyramid is realized by using the Geotrellis grid data processing library under the Spark parallel computing framework. The code of the image pyramid parallel builder has been packaged and compiled in advance, and placed on the main node of the Hadoop cluster, which is called by a shell script. The pseudocode of the whole parallel construction process is shown in Table 1.

For the image pyramid data built above, it is necessary to publish a map service using the Akka framework before it can be accessed by the front end for rapid visualization. The specific publishing process roughly includes building HTTP services, defining image

Table 1 Image pyramid parallel construction pseudocode

Input: The storage path of the pyramid image to be built in HDFS *inputpath*, pyramid layer name *layer-name*
Output: Image pyramid construction information, *result*

1. Remote SSH connection to the master node.
2. String command = " ./home/user/pyramid.sh"+" "+inputPath + " " + layerName
   // Define remote execution commands and pass input parameters at the same time.
3. ssh.execCommand(command);
   // Execute the command and remotely execute the pyramid builder on the master node
4. Set spark task parameters and establish SparkContext
5. val inputRdd: RDD[(ProjectedExtent, Tile)] = sc.hadoopGeoTiffRDD(inputPath)
   // Read the medium image data of HDFS and generate the corresponding RDD computing unit
6. Block RDD and re project to web Mercator projection respectively.
7. Build HBase read and write objects; Calculate the number of image pyramid layers.
8. Build image pyramid hierarchically and write data hierarchically into HBase database.
9. Get the result information of building image pyramid.

rendering methods, defining HTTP return data types, and rendering images into PNG format images according to the request range.

A class diagram of the parallel construction of the image pyramid and fast visual display module is shown in Fig. 12.

In the class diagram in Fig. 12, the *Pyramid* class defines an image pyramid object, including attribute information such as the name and ID of the image pyramid. The pyramid() method and publish() method in the *PyramidAndPublish* class are used to perform the image pyramid construction and image publishing functions, respectively, and implementation of the functions depends on the pyramid class. When building an

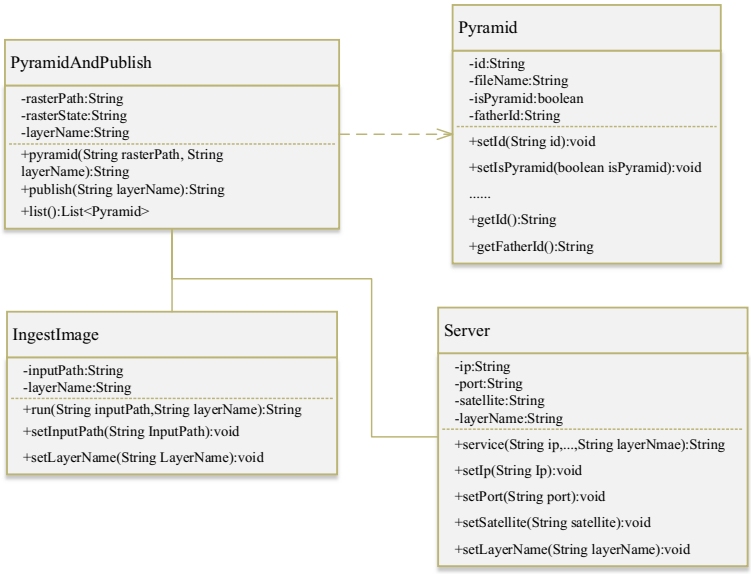

Fig. 12 Class diagram of the parallel construction and fast visual display module of the image pyramid

image pyramid, the *PyramidAndPublish* class will pass on the image storage location, pyramid name, and other parameter information, and call the run() method in the *IngestImage* class. When publishing images, the *PyramidAndPublish* class will build a server object and publish a map service according to parameters such as IP and port.

The image pyramid construction and release interface of the system is shown in Fig. 13. All remote sensing images in the system and their image pyramid construction are listed in the table. Users can select images to build pyramids according to their needs. For images that have been built as pyramids, users can click the "publish" button to publish the images. The system will build corresponding map services and display the images on the basemap, as shown in Fig. 14.

5.3 Realization of other functions

The image target detection module supports the detection of vehicles in the remote sensing images of UAV, and the detection algorithm uses the improved YOLOv4 algorithm [27]. The whole detection algorithm is deployed to the GPU node in the cluster in advance. The system only needs to call the corresponding shell script and pass on relevant parameters to complete the target detection by remote connection (as Secure Shell, i.e., SSH). The pseudo code of this module is shown in the Table 2.

The image target detection interface is shown in Fig. 15. The user can upload local images for target detection, select images in the system from the list on the right for direct detection, and select historical detection records for quick display. After detection is completed, the original image and the detected image will be displayed in the center of the interface at the same time.

For the final data source interface function, the system will place the data accessed by the third-party software into the additional startup file server and generate an access URL (Uniform Resource Locator).

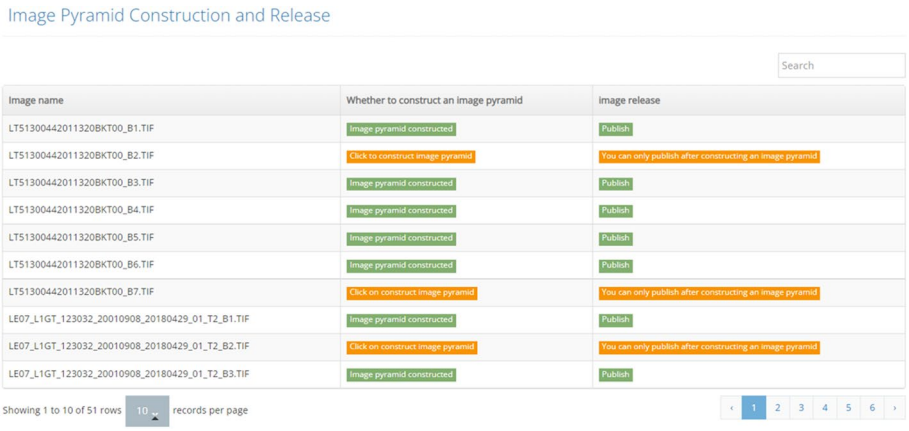

Fig. 13 Image pyramid construction and release interface

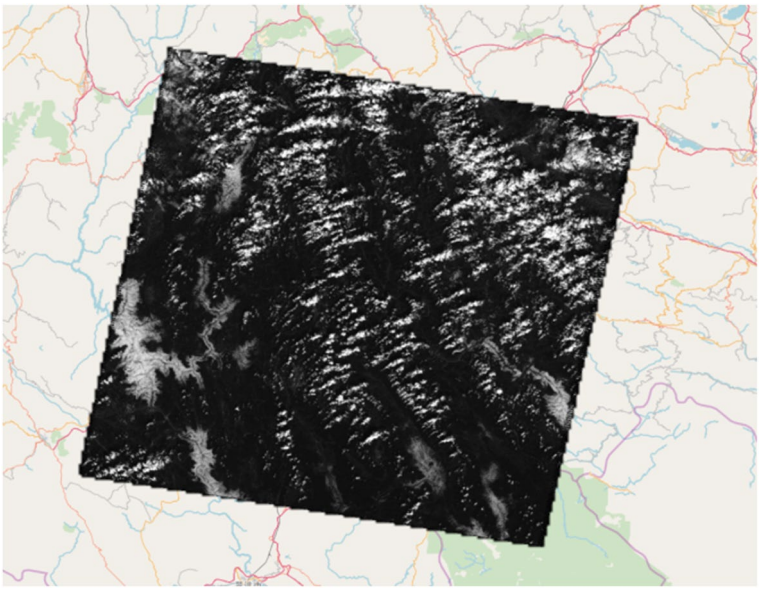

Fig. 14 Visual display of an image pyramid

Table 2 Pseudo code of image target detection

Input: Storage path of the image to be recognized in HDFS *inputpath*, to be recognized *image*, image category *type*
Output: Image recognition results, *outputImage*
1. Judge the image source *type*
2. If (the image is uploaded by the user)
3. Upload images to HDFS
4. Write image metadata information to HBase
5. Remote SSH connection GPU node
6. Pass the parameter *inputpath* and execute the identification script remotely
7. Obtain identification results *outputImage*
8.Else// The image comes from inside the system
9. Remote SSH connection GPU node
10. Pass the parameter *inputpath* and execute the identification script remotely
11. Obtain identification results *outputImage*

6 Experiment and analysis of MRSI-DSMS

6.1 Experimental software and hardware configuration and data

The whole system was built on an ordinary obsolete PC. The actual cluster configuration of the system is shown in Table 3. Among them, C5 has a GTX 750Ti GPU, and uses the six nodes listed in Table 3 to build the Hadoop cluster. The software installation of each node is shown in Fig. 16. C22 is the master node, and the other nodes are slave nodes. All nodes are connected through a gigabit LAN.

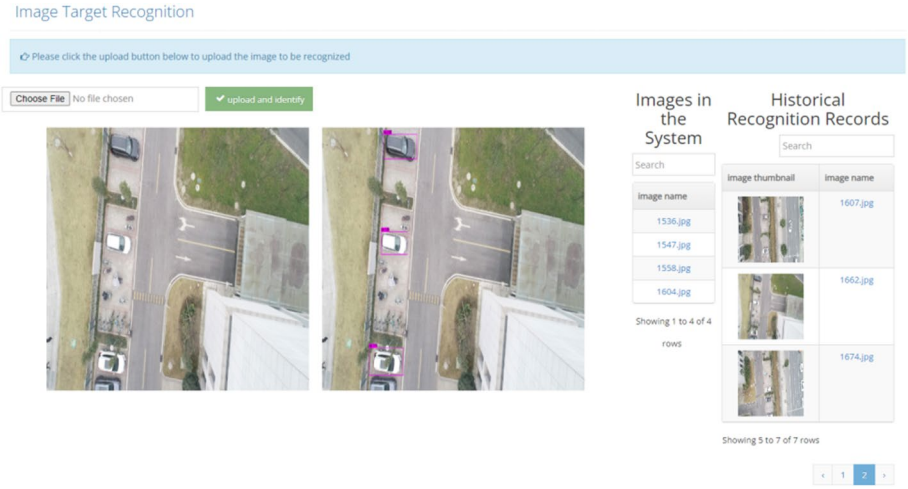

Fig. 15 Image target detection interface

Table 3 Configuration details of the Hadoop cluster constructed by PCs

No.	Host Name	CPU	RAM	Disk	Peak computing capability (GFlops)	OS	Network
1	C22	AMD A10-7800B	16GB DDR3	4 TB	57.6	CentOS 7.9	Gigabit LAN
2	C1	Core i5 3570	4GB DDR3	1 TB	56.0		
3	C2	Core i5 3570	4GB DDR3	1 TB	54.4		
4	C3	Core i5 3470	4GB DDR3	1 TB	54.4		
5	C4	Core i5 3470	4GB DDR3	500GB	51.2		
6	C5	Core i7 4790	8GB DDR3	2 TB	51.2+1387.5 (GPU)		

In this study, the system was tested in terms of storage capacity, upload and download speeds, and image pyramid construction speed. In terms of system storage capacity, it can be seen from Table 3 that the total storage space of the nodes except the master node is 5.5 TB, and HDFS consolidates the storage space of all nodes into one, providing storage support for this system; therefore, the maximum storage capacity supported by the system is approximately 5 TB.

In order to test the remote sensing image storage capacity, retrieval speed, multi-source image support, and image pyramid construction speed of this system, we downloaded and obtained remote sensing images from different satellite sources, such as Landsat 5, Landsat 7, Landsat 8, ZY-3, and GF-2, with a total size of nearly 3000 GB, as the test data for this experiment. At the same time, some UAV remote sensing images, ContextCapture 3D model data, and other non–remote sensing image data were also used for compatibility testing of third-party data in the system. Detailed information of the experimental test data is shown in Table 4.

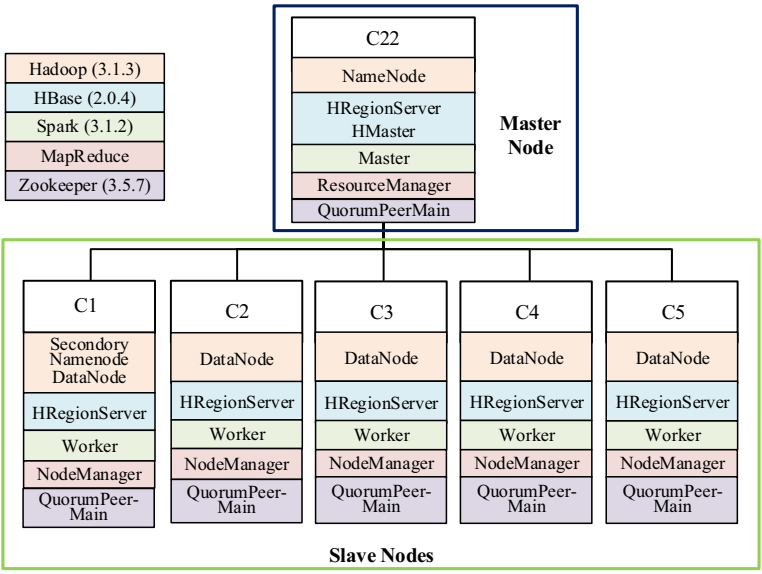

Fig. 16 Software installation of each node

Table 4 Details of the multi-source remote sensing datasets

Dataset name	Data format	Data Size (M)
LandSat5	TIF	53,910
LandSat7	TIF	225,262
LandSat8	TIF	113,783
ZY-3	tiff	1,177,969
GF-2	tiff	2,308,238
UAV	jpg	745
3D models	3mx	1,050

From Table 4, it can be seen that regarding the compatibility testing of this system for multi-source remote sensing images, remote sensing images from different sources were selected for storage in the experiment. In addition to the common Landsat series, ZY-3, and GF-2 remote sensing image data in various bands, the system also supports different types of third-party data files such as UAV remote sensing images and ContextCapture 3D model data. The warehousing of the third-party data is based on user-defined metadata files.

6.2 Performance testing of MRSI-DSMS

In terms of parallel construction of the image pyramid, the pyramid construction speed under different conditions was tested, and the results are shown in Table 5.

From Table 5, it can be seen that the time spent on building a single node of the image pyramid is approximately proportional to the size of the image data. The larger the image,

Table 5 Performance test of image pyramid construction

Image size (KB)	Serial image pyramid construction (s)	Multi-nodes image pyramid construction (s)	Performance improvement rate (X)
56,352	124.38	67.02	1.86
121,052	172.87	98.63	1.75
225,262	316.49	150.73	2.10
484,019	442.77	255.67	1.73
742,470	605.23	313.51	1.93

the longer the time spent on building. Using the Spark framework for distributed parallel processing of pyramid construction tasks can significantly improve the construction speed. Under the hardware conditions of this experiment, the obtained speedup can reach nearly 2× in the hardware configuration (shown in Table 5).

In terms of image retrieval capabilities, image retrieval can be divided into retrieval based on the spatial location of the image and retrieval based on image metadata information. The retrieval based on the spatial positions of images is to search for the images contained in the specified spatial range coordinates. In order to simulate the retrieval of massive images, this experiment used algorithms to simulate the metadata of 20 million remote sensing images, of which 10 million were stored in a traditional way in a single data table. The other 10 million pieces were stored in different tables according to the storage method designed by this system, based on the calculated Hilbert spatial index. Using the same search conditions, the data were searched under the above two conditions, and the test results are shown in Table 6.

From Table 6, it can be seen that in the case of a data volume of 10 million images, the retrieval speed of this system is 94.46%, 94.65%, and 93.92% faster than traditional single table storage methods for coordinate-based spatial range queries, image-name based attribute queries, and hybrid queries that combine spatial range and attribute queries, respectively. Regardless of the retrieval method, storing data in tables based on spatial indexing can filter out data from non-retrieval targets as much as possible, greatly improving the speed of retrieval.

Regarding the file transfer performance of this system, this experiment selected the Multi Source Remote Sensing Image Management System (MSRSIMS) studied by Zhu et al. [28] for comparative testing. The MSRSIMS system uses a single node server, and

Table 6 Details of the massive data retrieval experiment

Data storage method	Search conditions	Total	Number of search results	Retrieval time consumption (s)
Traditional single table storage	Spatial range	10 million	340	13.314
	Image attribute		1	10.822
	Spatial range + Image attribute		54	14.432
Partitioned Table Storage Based on Spatial Index	Spatial range	10 million	280	0.738
	Image attribute		1	0.579
	Spatial range + Image attribute		78	0.877

was precisely compared to the distributed multi-node server of this system. The testing was conducted in the same network environment, all within a gigabit local area network. The experimental data consisted of remote sensing image data from the ZY-3 satellite and UAV remote sensing images. The remote sensing image data were used to detect the system's transmission performance for large files, while the UAV image data were used to detect the system's transmission performance for massive amounts of small files. The results of the experimental test are shown in Table 7.

From Table 7, it can be found that the MRSI-DSMS system is faster than MSRSIMS in uploading a single large file, but the download speeds of the two are basically the same. For the download speed of multiple small files, MRSI-DSMS has the advantage over MSRSIMS. However, in terms of uploading multiple small files, MRSI-DSMS is slower than MSRSIMS. This is because distributed file systems perform steps such as file partitioning and multi-node distribution, which increases the total uploading time.

Regarding testing the number of concurrent users, this experiment simulated random mixed reading and writing requests from different numbers of users on the database. The response times were recorded in MySQL single node deployment and master–slave synchronous deployment, respectively. The detailed data are shown in Table 8.

From Table 8, it can be found that the master–slave deployment method exhibits certain performance advantages compared to the single node deployment method under different concurrent user numbers. Under the same number of concurrent users, the average response time of the master–slave mode is usually slower than that of the single node deployment mode. When the number of concurrent users is five, the average response time of this system is only 2% faster than the single node mode. However, when the number of concurrent users reaches 200, the average response time of this system is 10.1% faster than the single node mode. As the number of concurrent users increases, the performance advantage of master–slave mode becomes more apparent. The above test results indicate that the MySQL database deployed in master–slave mode achieved the successful separation of reading and writing requests, and has improved response speed to a certain extent under high concurrency requests.

6.3 Experiment summary

From the above experiments, it was demonstrated that the designed metadata parser can successfully support the proposed system for many types of remote sensing image data.

Table 7 Comparative experiment on file transfer performance

System	Transmission type	Number of Files	File size (GB)	Time consuming (s)	Transmission speed (MB/s)
MSRSIMS	Upload	1	3.73	48.562	78.65
	Upload	1938	1.45	34.354	42.99
	Download	1	3.73	42.453	89.97
	Download	1938	1.45	35.359	41.99
MRSI-DSMS	Upload	1	3.73	39.048	97.74
	Upload	1938	1.45	128.146	11.59
	Download	1	3.73	43.593	87.618
	Download	1938	1.45	24.963	59.48

The use of the HDFS distributed file system to store image data solves the distributed storage problem of massive image data and enhances the data transmission ability. Utilizing the Spark cluster for image pyramid construction allows the system to divide tasks into multiple nodes, thereby improving construction speed. By utilizing the MySQL database to store image metadata and employing the Hilbert curve to construct spatial indexes, we achieved the horizontal table storage of data, quickly filtered out data from non-retrieval targets, and greatly improved the data retrieval speed. By utilizing MySQL's master–slave synchronization mode, the system's ability to cope with high concurrent user access has been enhanced.

7 Conclusion and future works

This research mainly studied how to design a distributed remote sensing image storage and management system for a huge number of images based on the Hadoop distributed system framework and its related components, as well as the MySQL database. Using the Hadoop framework can effectively solve the problem of weak storage and computing power of a single server due to the large amount of remote sensing image data. After testing in a gigabit LAN environment, the average upload speed of a single remote sensing image was 97.74 MB/s and the average download speed was 87.62 MB/s. Based on the Hilbert curve, the spatial index is calculated and the data are stored in separate tables, resulting in an average improvement of 94.34% in data retrieval speed compared to traditional single table storage methods, solving the problem of slow retrieval when faced with massive data sets. Deploying MySQL with master–slave replication achieves data reading and writing separation, reducing the pressure on a single server under high user concurrency. When multiple users make reading and writing requests simultaneously, the average response time of the system is up to 10.1% higher than the traditional single node mode. In terms of image pyramid construction, the speed of the multi-node parallel construction based on Spark was two times higher than that of the single-node construction. The WMTS services were

Table 8 Experiment on concurrent user access and the response time

MySQL deployment method	Number of concurrent users	Total response time (s)	Average response time (s)
Single node	5	1.047	0.210
	20	2.725	0.136
	50	7.902	0.158
	100	19.286	0.193
	150	28.484	0.190
	200	39.773	0.199
Master–slave synchronization mode (1 master and 2 slaves)	5	1.023	0.205
	20	2.684	0.134
	50	7.531	0.151
	100	17.342	0.173
	150	25.579	0.171
	200	35.754	0.179

built based on the Akka framework, which achieved the fast rendering and visualization of image pyramid data at any level according to user needs. It was found that the system could perfectly meet the functions of image management, uploading and downloading, image pyramid construction, and fast visualization. It has good data transmission and retrieval speed, good data computing ability, and higher concurrency processing ability.

However, MRSI-DSMS needs to be further improved and optimized to give better play to Hadoop's distributed storage and Spark's parallel computing capabilities, where some factors have not been considered, and some tests need to be further conducted. For example, (1) due to the relatively backward cluster configuration, the speed of building image pyramids in parallel can be further improved; (2) due to the limitation of the large physical memory of the device, the image pyramid only supports images with a maximum size of 1 GB; (3) support for multiband images needs to be improved; and (4) the rendering effect of some types of remote sensing images also needs to be optimized.

The following issues need to be dealt with in future research: (1) Spark parameters can be optimized according to the actual configuration of the cluster to maximize the computing power of the cluster; (2) on the existing basis, the pyramid construction of multiband images was studied; and (3) a universal rendering scheme of remote sensing image was given.

Acknowledgements This study was mainly supported by the National Science Foundation of China (Grant No. 42271390) and the Technological Innovation R&D Project of Chengdu Science and Technology Bureau (Grant No. 2022-YF05-00967-SN). This work was partial funded by the Fundamental Research Funds for the Central Universities (Grant Nos. ZYGX2019J069 and ZYGX2019J072) and Hubei Provincial Key Laboratory of Intelligent Geo-information Processing (China University of Geosciences; Grant Nos. KLIGIP-2018A08).

Data availability Due to the nature of this research, the participants of this study do not agree for their data to be shared publicly. Hence, supporting data are not available.

Declarations

Conflict of interest The authors declare that they have no conflicts of interest.

References

1. Blaschke T (2010) Object based image analysis for remote sensing. ISPRS J Photogramm Remote Sens 65(1):2–16
2. Deren LI, Liangpei Z, Guisong X (2014) Automatic analysis and mining of remote sensing big data. Acta Geodaetica Cartogr Sin 43(12):1211
3. Chi M, Plaza A, Benediktsson JA et al (2016) Big data for remote sensing: challenges and opportunities. Proc IEEE 104(11):2207–2219
4. Huang YQ (2019) The concept and development trend of spatial database management system. China Manage Informationization 22(08):165–166
5. Lü XF, Cheng C, Gong J et al (2011) Review of data storage and management technologies for massive remote sensing data. Sci China Technol Sci 54:3220–3232
6. Yan J, Liu Y, Wang L et al (2021) An efficient organization method for large-scale and long time-series remote sensing data in a cloud computing environment. IEEE J Sel Top Appl Earth Observ Remote Sens 14:9350–9363

7. Wang L, Ma Y, Yan J et al (2018) PipsCloud: high performance cloud computing for remote sensing big data management and processing. Futur Gener Comput Syst 78:353–368
8. Cheng Y, Zhou K, Wang J, Yan J (2020) Big earth observation data integration in remote sensing based on a distributed spatial framework. Remote Sens 12(6):972. https://doi.org/10.3390/rs12060972
9. Jing W, Tian D (2018) An improved distributed storage and query for remote sensing data. Procedia Comput Sci 129:238–247
10. Li J, Zhang P, Li Y et al (2017) A data-check based distributed storage model for storing hot temporary data. Futur Gener Comput Syst 73:13–21
11. Zheng K, Fu Y (2013) Research on vector spatial data storage schema based on Hadoop platform. Int J Database Theory Appl 6(5):85–94
12. Zhong Y, Sun S, Liao H, Zhao Y, Fang J (2011) A novel method to manage very large raster data on distributed key-value storage system. In: 2011 19th International Conference on Geoinformatics. Shanghai, China, pp 1–6. https://doi.org/10.1109/GeoInformatics.2011.5980711
13. Rajak R, Raveendran D, Bh MC, Medasani SS (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. https://doi.org/10.1109/CCEM.2015.16
14. Zhu J, Zhang Z, Zhao F et al (2023) Efficient management and scheduling of massive remote sensing image datasets. ISPRS Int J Geo-Information 12(5):199
15. Zhou X, Wang X, Zhou Y et al (2021) Rsims: large-scale heterogeneous remote sensing images management system[J]. Remote Sens 13(9):1815
16. Wang C, Hu F, Hu X et al (2015) A Hadoop-based distributed framework for efficient managing and processing big remote sensing images. ISPRS Ann Photogramm Remote Sens Spat Inf Sci 2:63–66
17. Kong W, Wang T, Liu L et al (2023) A novel design and application of spatial data management platform for natural resources. J Clean Prod 411:137183
18. Wei H, Yuheng Z (2017) The massive remote sensing data organization and management strategies// MATEC Web of Conferences. EDP Sciences 128:02011
19. Wang S, Li G, Yao X et al (2019) A distributed storage and access approach for massive remote sensing data in MongoDB. ISPRS Int J Geo-Information 8(12):533
20. Rathore MM, Ahmad A, Paul A et al (2016) Urban planning and building smart cities based on the internet of things using big data analytics. Comput Netw 101:63–80
21. Shan TJ, Zhong HW et al (2019) Building of remote sensing images tile pyramid based on Spark. Intell Comput Appl 9(04):226–229
22. Zaharia M, Xin RS, Wendell P et al (2016) Apache spark a unified engine for big data processing. Commun ACM 59(11):56–65
23. Kini A, Emanuele R, Geotrellis (2014) Adding geospatial capabilities to Spark. In: Spark Summit 2014, from https://docs.huihoo.com/apache/spark/summit/2014/Geotrellis-Adding-Geospatial-Capabilities-to-Spark-Ameet-Kini-Rob-Emanuele.pdf
24. Chen X, Zhang C, Ge B et al (2016) Efficient historical query in HBase for spatio-temporal decision support. Int J Comput Commun Control 11(5):613–630
25. Jonasson M (2014) The Akka-board–performing mobility, disability and innovation. Disabil Soc 29(3):477–490
26. Farkas G (2017) Applicability of open-source web mapping libraries for building massive web GIS clients. J Geogr Syst 19(3):273–295
27. Huang F, Chen S, Wang Q et al (2023) Using deep learning in an embedded system for real-time target detection based on images from an unmanned aerial vehicle: vehicle detection as a case study[J]. Int J Digit Earth 16(1):910–936
28. Zhu Q, Huang F, Lu J et al (2017) Research on the implementation of multi-source remote sensing image management system based on B/S architecture. 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). IEEE, pp 5233–5236

Authors and Affiliations

Lichun Yang[1,2] · Weibing He[3] · Xiaoyong Qiang[3] · Jinjun Zheng[3] · Fang Huang[3]

✉ Lichun Yang
yanglc2003@buaa.edu.cn

[1] School of Transportation Science and Engineering, Beihang University, Beijing 100191, China

[2] Jiangsu Automation Research Institute, Lianyungang, Jiangsu Province 222061, China

[3] School of Recourses and Environment, University of Electronic Science and Technology of China (UESTC), Chengdu, Sichuan Province 611731, China