remote sensing

MDPI

Article

Distribution-Based Approach for Efficient Storage and Indexing of Massive Infrared Hyperspectral Sounding Data

Han Li [1,2,3], Mingjian Gu [1,3,*], Guang Shi [1,2,3,4], Yong Hu [1,3] and Mengzhen Xie [1,3]

[1] Key Laboratory of Infrared Science and Technology, Shanghai Institute of Technical Physics, Chinese Academy of Sciences, Shanghai 200083, China; lihan2019@mail.sitp.ac.cn (H.L.); sg980223@mail.ustc.edu.cn (G.S.); huyong@mail.sitp.ac.cn (Y.H.); xiemengzhen@mail.sitp.ac.cn (M.X.)
[2] University of Chinese Academy of Sciences, Beijing 100049, China
[3] Shanghai Institute of Technical Physics, Chinese Academy of Sciences, Shanghai 200083, China
[4] Department 23, School of Microelectronics, University of Science and Technology of China, Hefei 230023, China
* Correspondence: gumingjian@mail.sitp.ac.cn

Abstract: Hyperspectral infrared atmospheric sounding data, characterized by their high vertical resolution, play a crucial role in capturing three-dimensional atmospheric spatial information. The hyperspectral infrared atmospheric detectors HIRAS/HIRAS-II, mounted on the FY3D/EF satellite, have established an initial global coverage network for atmospheric sounding. The collaborative observation approach involving multiple satellites will improve both the coverage and responsiveness of data acquisition, thereby enhancing the overall quality and reliability of the data. In response to the increasing number of channels, the rapid growth of data volume, and the specific requirements of multi-satellite joint observation applications with infrared hyperspectral sounding data, this paper introduces an efficient storage and indexing method for infrared hyperspectral sounding data within a distributed architecture for the first time. The proposed approach, built on the Kubernetes cloud platform, utilizes the Google S2 discrete grid spatial indexing algorithm to establish a grid-based hierarchical model for unified metadata-embedded documents. Additionally, it optimizes the rowkey design using the BPDS model, thereby enabling the distributed storage of data in HBase. The experimental results demonstrate that the query efficiency of the Google S2 grid-based embedded document model is superior to that of the traditional flat model, achieving a query time that is only 35.6% of the latter for a dataset of 5 million records. Additionally, this method exhibits better data distribution characteristics within the global grid compared to the H3 algorithm. Leveraging the BPDS model, the HBase distributed storage system adeptly balances the node load and counteracts the detrimental effects caused by the accumulation of time-series remote sensing images. This architecture significantly enhances both storage and query efficiency, thus laying a robust foundation for forthcoming distributed computing.

Keywords: infrared hyperspectral sounding data; HIRAS; distributed storage; kubernetes; HBase

check for updates

Citation: Li, H.; Gu, M.; Shi, G.; Hu, Y.; Xie, M. Distribution-Based Approach for Efficient Storage and Indexing of Massive Infrared Hyperspectral Sounding Data. *Remote Sens.* **2024**, *16*, 4088. https://doi.org/10.3390/rs16214088

Academic Editor: Salah Bourennane

Received: 23 September 2024
Revised: 28 October 2024
Accepted: 31 October 2024
Published: 1 November 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The application of infrared hyperspectral technology offers significant benefits in the retrieval of vertical atmospheric profiles, including temperature, humidity, and gas composition. Its contributions are pivotal for data assimilation in numerical weather prediction models, atmospheric composition retrieval, climate change research, and various scientific investigations [1]. It has become essential to acquire global meteorological observation data. With advancements in satellite technology, multiple satellites operating simultaneously can observe the Earth's atmosphere, producing multi-source and multi-angle data. This collaborative satellite observation approach enhances data coverage, timeliness, and overall quality and reliability. Over time, a considerable amount of infrared hyperspectral sounding data has been amassed. To effectively leverage these extensive datasets, users encounter various

limitations, constrained not only by traditional data storage and management methods but also by the resource and processing capacity limitations of personal computers [2]. These limitations not only significantly hinder efficient data analysis and processing, but also complicate efforts for researchers to fully leverage the potential value of the data [3].

In response to the growing complexity of task requirements and emerging challenges, the integration of infrared hyperspectral technology and cloud computing holds great promise for the efficient processing and extraction of valuable information from large-scale multi-satellite infrared hyperspectral sounding data. The effectiveness of cloud computing in managing remote sensing big data has been well established [4]. Over the past decade, remote sensing cloud computing platforms have experienced rapid development in the realm of traditional remote sensing imagery, with Google Earth Engine (GEE) [5] being the most notable example. It provides access to a vast array of remote sensing datasets, along with internet-based application programming interfaces (APIs) and web-based interactive development environments (IDEs). Scholars from both domestic and international backgrounds have utilized the GEE platform for research, achieving significant advancements in multiple areas of study [6]. Nevertheless, there are still some constraints on GEE, such as the limitation of data and the application scenarios, rendering it unsuitable for addressing all remote sensing challenges. Google has released the big data storage component, the Hadoop Distributed File System (HDFS), along with its active involvement in the open-source community, which has fostered the exploration of integrating cloud computing with remote sensing applications.

Traditional remote sensing imagery data are generally derived from multispectral imagery, which is commonly stored and indexed as single-band tiles within remote sensing cloud computing platforms. Hyperspectral imagery obtained from satellite platforms, such as the Gaofen-5 Advanced Hyperspectral Imager (GF-5/AHSI) [7], delivers extensive spectral and spatial information regarding ground objects, with predominant applications in Earth observation. For various application objectives, two primary processing methods exist: one primarily focused on spatial information and the other on spectral information. Currently, several distributed applications have emerged, including hyperspectral dimensionality reduction [8], spatial–spectral classification [9], endmember extraction [10], image unmixing [11], target detection [12], and anomaly detection [13].

However, in contrast to most hyperspectral imaging applications that necessitate the simultaneous integration of spatial and spectral information for Earth observation, infrared hyperspectral sounding data primarily serve to detect vertical atmospheric information. Their applications are predominantly reliant on spectral processing, exhibiting limited sensitivity to spatial partitioning. Currently, mainstream methods for processing infrared hyperspectral sounding data employ radiative transfer models, including both forward and inverse models such as the Line-By-Line Radiative Transfer Model (LBLRTM) developed by Atmospheric and Environmental Research (AER) in the United States, the Community Radiative Transfer Model (CRTM) developed by the Joint Center for Satellite Data Assimilation (JCSDA) in the United States, and the Rapid Radiative Transfer Model (RTTOV) developed by the European Centre for Medium-Range Weather Forecasts (ECMWF) in Europe, as well as various statistical regression techniques [14–16]. Radiative transfer models face several challenges, including complex computations and reliance on environmental conditions, which can result in limitations due to single-machine constraints, difficulties in scaling resources, and low computational efficiency. Containerization technology in cloud computing offers innovative solutions to address these challenges.

Currently, there are no documented instances of integrating infrared hyperspectral sounding data with distributed storage and computing. However, considerable research has focused on the storage and processing of remote sensing images on cloud platforms. Most remote sensing data are typically organized through various file management systems. Commonly utilized distributed storage systems include HDFS, Ceph, GlusterFS, Lustre, MooseFS, and GFS. These storage systems perceive remote sensing images as merely large unstructured binary files, disregarding their specific data structures and spatiotemporal

attributes [17]. This perspective poses challenges for supporting large-scale computational applications in subsequent data processing. As a result, it is frequently necessary to incorporate additional big data components, such as HBase, along with management strategies for a more rational organization and processing of remote sensing data. Jing [18] investigated image data storage by taking advantage of the characteristics of the HBase distributed database, resulting in commendable outcomes. Xu [19] employed the HDFS in conjunction with HBase to facilitate the storage and management of remote sensing data, integrating quadtree structures with the Hilbert curve to enhance the performance of data query and retrieval. Zhou [17] utilized the Ceph distributed storage platform to establish the Remote Sensing Information Management System (RSIMS), applying a distributed multi-level Hilbert grid for indexing that enhances spatiotemporal performance. Yang [20] analyzed the chunking approach and encoding specifications of Google S2, achieving layered storage of remote sensing data in HBase.

The design of the rowkey within the HBase distributed storage system is directly related to the efficiency of the storage indexing process. Currently, research is exploring the integration of ASCII [21], GeoSOT [22], Hilbert curve [23], and Google S2 [20] with metadata to design geographic location codes known as rowkeys. However, these studies primarily focus on the implications of data accumulation for node load and query efficiency. Yan [24] developed a pre-partitioning mechanism within the HBase architecture that facilitates the unified storage of metadata characterized by asymmetric spatiotemporal distribution, thereby mitigating retrieval efficiency bottlenecks resulting from load imbalance. Building on this research, Wang [25] introduced two strategies, BPS and PSS, aimed at eliminating load imbalance among nodes. Furthermore, attribute-based metadata retrieval and location-based spatial retrieval are critical considerations in remote sensing data management. In previous studies, a common approach has been to construct a secondary index on the column family metadata in HBase, thereby closely integrating data indexing with data storage operations. This integration may result in load hotspot issues during high-concurrency periods. Considering the low frequency of modifications to remote sensing data after loading, decoupling data indexing from data storage can significantly improve query performance [26]. Wang [27] and Cheng [28] employed the sharding architecture of MongoDB, along with its extensive querying capabilities, to manage raster data and metadata for large-scale distributed remote sensing datasets. The NoSQL-based object storage mechanism streamlines the management and retrieval of vast amounts of remote sensing data while mitigating the risk of single points of failure [29].

Within the framework of cloud computing, the storage of big data forms the foundation for distributed computation and efficient processing. In order to address the challenge of storage and indexing of extensive infrared hyperspectral sounding data and establish the foundation for subsequent distributed computing, we propose an efficient storage and indexing method within a distributed architecture. In this research, we commence by constructing a container-based distributed platform that is orchestrated and governed by Kubernetes to effectively manage diverse big data components with excellent scalability and disaster tolerance (Section 2.2). Concurrently, guided by an analysis of the attributes of infrared hyperspectral detection data, we employed Google S2's spatial indexing constraints for data organization and management. To address the issue of load balancing in storage, we introduced the balanced periodic distribution strategy (BPDS) model to optimize rowkey design, enabling the distributed storage of remote sensing data in HBase. Through enhancements to the data index structure, we ensured high performance in handling extensive volumes of infrared hyperspectral sounding data (Section 2.3). Subsequent to these developments, relevant experiments were conducted (Section 3).

2. Materials and Methods

2.1. Datasets

Currently, there are some main hyperspectral atmospheric sounding instruments operating in orbit worldwide: the Atmospheric Infrared Sounder (AIRS) [30] aboard the

Earth Observing System Aqua (EOS-Aqua) satellite in the United States, the Infrared Atmospheric Sounding Interferometer (IASI) [31] on the Metop-A/B/C satellites in Europe, and the Cross-track Infrared Sounder (CrIS) [32] onboard the Suomi National Polar-orbiting Partnership (SNPP) satellite in the United States. Additionally, the Infrared Sounder (IRS)-equipped Meteosat Third Generation Sounder (MTG-S) [33] geostationary satellite is expected to be launched in the coming years. China has developed hyperspectral infrared atmospheric detectors, referred to as the Hyperspectral Infrared Atmospheric Sounder (HIRAS) and the Geostationary Interferometric Infrared Sounder (GIIRS), which will be installed on the Fengyun (FY)-3D/E/F satellites and the new generation of geostationary meteorological satellites, FY-4A/B, respectively. The Table 1 below presents the currently widely used spaceborne atmospheric sounding infrared hyperspectral payloads.

Table 1. Retrieval of the dataset for each gas composition.

Satellite	Payload	Optical Splitting Mode	Wavenumber (cm^{-1})	Spectral Resolution (cm^{-1})	Channel (Apodized)	Space Resolution (km)
EOS-Aqua	AIRS	Diffrangible	659–1136 1212–1612 2169–2673	L:0.55M:1.2S:2.0	2378	13
MetOp-A/B/C	IASI	Interferometric	645–1210 1210–2100 2100–2700	0.25	8461	12
S-NPP	CrIS	Interferometric	650–1095 1210–1750 2155–2550	0.625	2211	14
MTG-S	IRS	Interferometric	700–1210 1600–2200	0.625	1740	4
FY-3D	HIRAS	Interferometric	648.75–1134 1208.5–1749.375 2153.75–2549.375	0.625	2275	16
FY-3E/F	HIRAS-II	Interferometric	650–1168.125 1168.75–1920 1920.625–2550	0.625	3041	14
FY-4A	GIIRS	Interferometric	700–1130 1650–2250	0.625	1650	16
FY-4B	GIIRS-II	Interferometric	680–1130 1650–2250	0.625	1681	12

In this study, we utilized satellite data from the Hyperspectral Infrared Atmospheric Sounder (HIRAS) and HIRAS-II instruments on the FY-3D/E/F satellites (http://satellite.nsmc.org.cn/portalsite/default.aspx, accessed on 12 September 2024) and conducted subsequent analyses based on these data. It is noteworthy that while the methods discussed herein have general applicability for processing infrared hyperspectral sounding data, the specific characteristics of each satellite product must be considered, as there are variations in their preprocessing. A comprehensive understanding of the operational characteristics, scanning methods, and data features of the instruments is essential for effective satellite data processing and subsequent applications, such as the retrieval of key atmospheric parameters.

For example, the HIRAS data analyzed in this study indicate that both HIRAS and HIRAS-II have a scanning field-of-view range of $\pm 50.4^{\circ}$, with a stepping angle of 3.6° for adjacent fields of view [34]. The HIRAS instrument employs a 2×2 detector array to observe the same Earth scene across three bands, where the combined observations of four target regions from each detector during a single observation form one field of view (FOV). In contrast, HIRAS-II uses a 3×3 detector array to create its field of view. The Figure 1 illustrates the cross-track scanning diagrams for the two instruments [35]. During the instrument's operational phase, it is essential to conduct tests related to detector sensitivity. Zhang [36] indicated that the sensitivity of FOV1 for FY-3E/HIRAS-II is notably lower than others, highlighting a potential concern. Considering the need to assess the

long-term operational stability of the instrument and utilize high-sensitivity detector data for retrieval, recording the sequence of the fields of view (FOVs) is crucial. This practice not only enhances our understanding of the temporal characteristics of sensor performance but also serves as a basis for effective data post-processing and algorithm optimization. By systematically documenting the sequence of FOVs, researchers can analyze sensitivity variations under different observation conditions, leading to the development of appropriate calibration and correction strategies that improve the reliability and accuracy of the data.

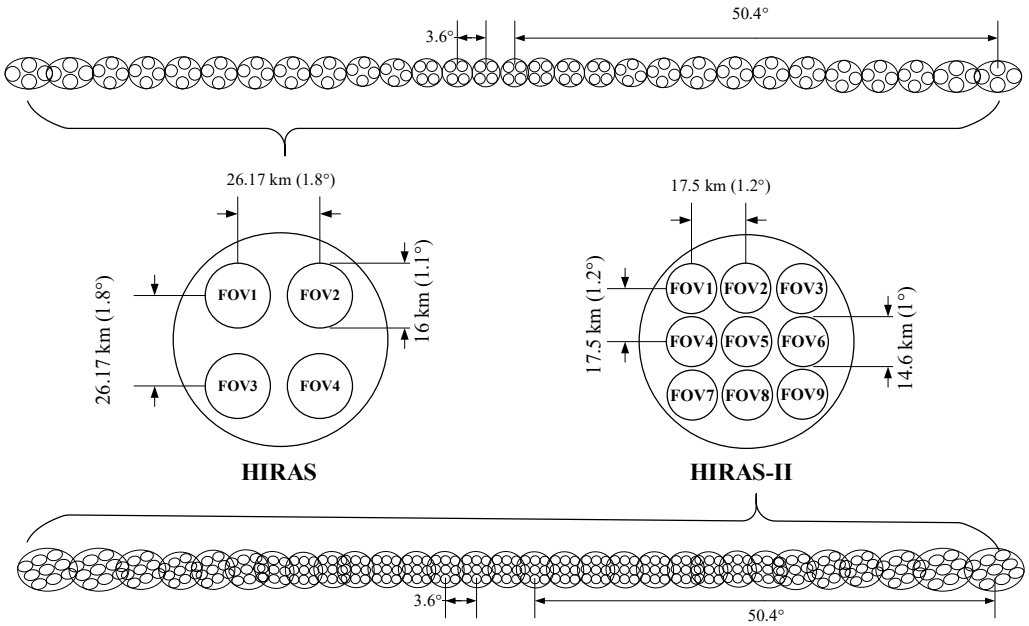


Figure 1. Illustration of ground field-of-view arrangement for the HIRAS series detector [36,37].

2.2. System Architecture Overview

Before introducing the architecture used in this paper, we will briefly outline the relevant cloud computing technologies.

(1) Docker: This is a lightweight container with an independent dependency environment, enabling rapid deployment and portability of applications, providing a flexible and efficient deployment method. This makes the application environment easier to manage and more portable in the cloud.

(2) HDFS (Hadoop Distributed File System): This is a distributed file system within the Apache Hadoop ecosystem, used for storing and processing large-scale datasets. Its significance in cloud computing resides in its provision of capabilities for large-scale data processing, elastic scalability, and data reliability, thereby establishing a dependable groundwork for the storage and processing of large-scale data in cloud environments.

(3) HBase: HBase is a distributed, column-oriented NoSQL database based on Hadoop. Within the realm of cloud computing, its importance is underscored by its provision of dependable large-scale data storage and real-time accessibility, alongside its support for elastic scalability and high-reliability features.

(4) Kubernetes: Kubernetes is an open-source container orchestration platform used for automating the deployment, scaling, and management of containerized applications. In cloud computing, Kubernetes' significance is reflected in simplifying deployment and management, elastic scalability, and cross-platform support, providing an efficient environment for running containerized applications.

The architecture diagram depicting the efficient storage and indexing of massive infrared hyperspectral sounding data based on distributed systems, as proposed in this paper, is illustrated in Figure 2 below. The system comprises three components: data source access and processing (Part 1), Kubernetes cluster management (Part 2), and data storage in scalable and extensible containers (Part 3).

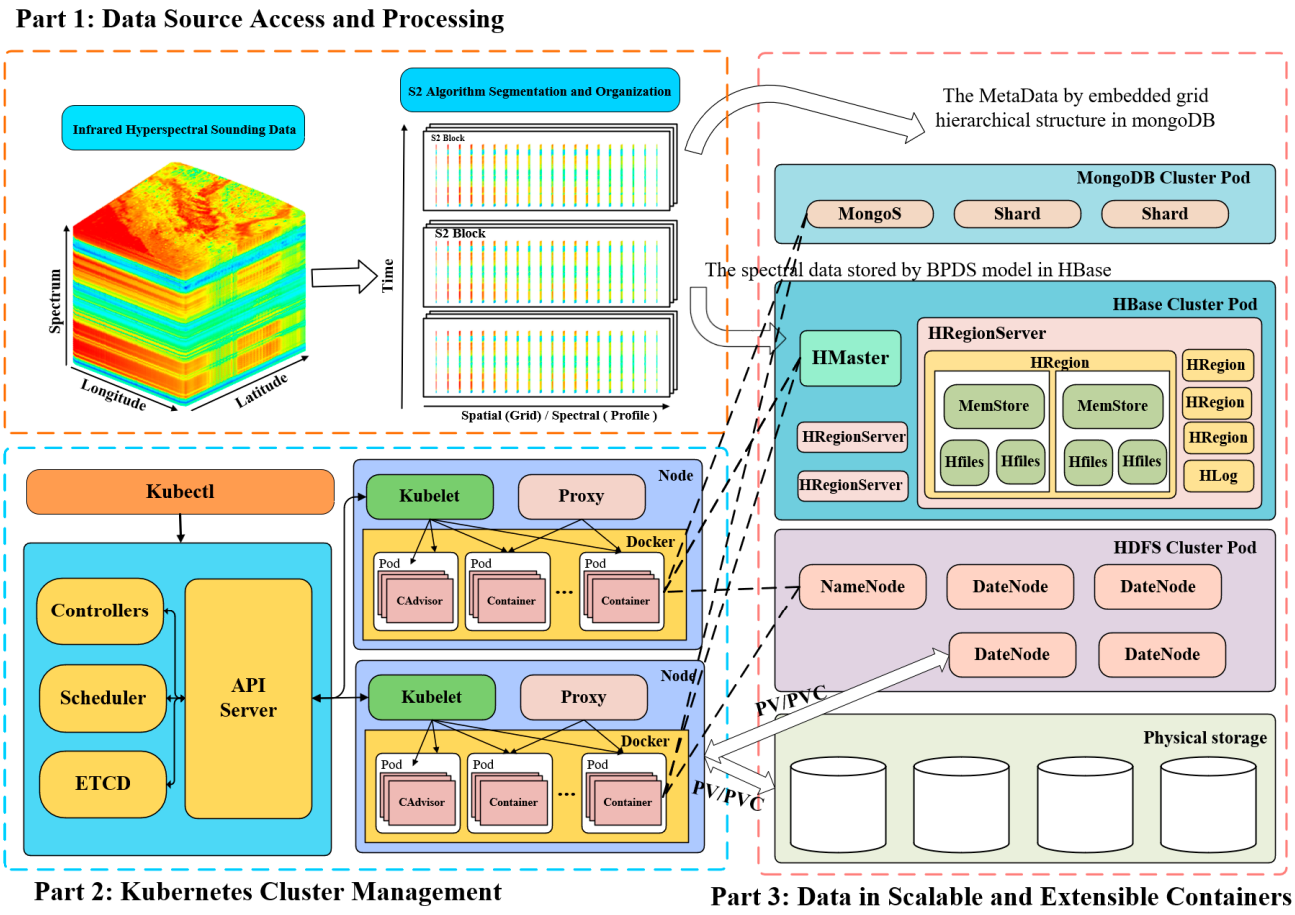


Figure 2. The system architecture for the efficient storage and indexing of hyperspectral infrared sounding data.

In the first part, the system acquires the accessed infrared hyperspectral sounding data and processes them using specific methods. The processed remote sensing metadata are stored in a distributed MongoDB cluster, while the spectral data are stored in an HBase distributed cluster. Detailed explanations of the specific processing methods will be provided in subsequent sections.

The second part focuses on Kubernetes cluster management. In the Kubernetes environment, pods serve as the smallest deployable units, capable of dynamically scaling up or down based on resource demands to optimize the utilization of computing resources. Each pod contains at least one docker container hosting the application. The Kube-apiserver, Kube-controller, Kube-scheduler, and etcd are responsible for providing API services, running controllers, scheduling pods, and maintaining cluster states, respectively. Node components consist of Kubelet, Kube-proxy, and Container, which, respectively, manage pods and containers on nodes, maintain network rules, and run container software. Additionally, network plugins and storage plugins are integral parts of Kubernetes, responsible for providing network communication capabilities and persistent storage capabilities for pods, thus enabling efficient deployment, management, and scalability of containerized applications.

The third component addresses data storage within flexible and scalable containers. This section includes services such as HDFS, HBase, and MongoDB, all deployed in a containerized architecture. This design enhances both the flexibility and scalability of the system while optimizing resource management and utilization. Data are mounted to physical storage via Kubernetes' Persistent Volumes (PVs), utilizing Kubernetes' key feature of automatically managing container distribution across the node cluster to ensure high availability and fault tolerance. Additionally, the system incorporates self-healing

capabilities, automatically restarting failed containers and replacing them as necessary to maintain the desired operational state.

This big data storage architecture operates such that all nodes engaged in data ingestion and index reading serve as both computing and storage nodes, effectively eliminating the need for data movement between computing nodes which enhances efficiency. In the event of a requirement for large-scale computing nodes in the future, supplementary nodes can be elastically provisioned, enabling the segregation of storage and computation. This segregation facilitates adaptability to a broader array of application scenarios.

2.3. Organization and Storage of Infrared Hyperspectral Sounding Data

In the field of infrared hyperspectral atmospheric sounding, data representation is a significant issue of interest for researchers. Although infrared hyperspectral atmospheric sounding data are distinct from traditional image data and are challenging to present intuitively, there are similarities in the representation methods of both types of data. Both infrared hyperspectral atmospheric sounding data and remote sensing image data rely on two main components: metadata and digital matrix information. Metadata are essential for recording important attributes such as data status, source, and acquisition time, which aids in subsequent processing and analysis. During data collection, signals pass through a complex optical path structure; after undergoing Fourier interferometry or grating spectroscopy and calibration, spectral intensity values at different wavelengths are obtained, forming a data matrix. The sensitivity of different fields of view (FOVs) significantly influences the quality of subsequent data, and this is of particular concern to researchers, especially in comparison to area detectors used in imaging.

In contrast to traditional image data, which typically exhibit strong block characteristics and maintain spatial structural integrity, infrared hyperspectral data possess distinct intrinsic properties despite sharing a similar three-dimensional structure. The primary focus of infrared hyperspectral data lies in its spectral information, where pixel-level hyperspectral information serves as the fundamental unit. Moreover, the data demonstrate relative dispersion during processing, displaying minimal reliance on the values of spatially adjacent pixels. Additionally, strong infrared radiation from clouds often significantly interferes with certain spectral curves frequently requiring the exclusion of spectra from these pixels. The mechanisms for sounding vertical information above the clouds are inadequately clear, creating challenges for data processing under intricate atmospheric conditions. As a result, researchers typically focus on analyzing the spectra of clear sky pixels when applying forward and inverse radiation transfer models and statistical regression methods. However, there is still some correlation between nearby sounding points, suggesting avenues for further research.

Following the analysis of the characteristics of infrared hyperspectral sounding data discussed earlier, this paper considers single-pixel data as the smallest storage unit in Hbase. It employs an embedded grid hierarchical structure based on the Google S2 algorithm for metadata indexing. This structure enables the unified management of dispersed small cells (the sounding points) by employing larger cells. The embedded documents thoroughly document information essential for filtering and displaying sounding points. Further elaboration on specific details will be provided in the subsequent sections.

2.4. Spatial Index Constraint Based on Google S2

This study employs the Google S2 algorithm, a global geographic grid, to manage pixel points and generate blocks. Furthermore, it constructs a storage and organization model based on S2 encoding. The Google S2 algorithm is a grid encoding technique for geospatial neighborhoods that uses the Hilbert curve for multidimensional spatial point indexing [38]. The algorithm executes a series of transformations and projections based on longitude and latitude, ultimately mapping the data onto the six faces of a cube to ensure that the cell sizes on each face are approximately equal through calibration. Each division is then subdivided into four subspaces, connecting points in space via a continuous Hilbert curve,

which facilitates efficient spatial indexing. This process reduces the dimensionality of the two-dimensional coordinates on the plane and encodes them into CellID grid encoding [39]. The detailed algorithm is presented in Table 2. This algorithm ensures that the partitioning units align consistently with the spherical surface. Given the advantageous properties of the Hilbert curve, the resulting CellID from units exhibiting high-dimensional spatial proximity retains similar proximity characteristics. Consequently, this method is suitable for the global storage and analysis of spatial data, facilitating efficient data management and retrieval.

Table 2. Google S2 algorithm.

Google S2 Algorithm

Input: <*Longitude; Latitude; Level*>
Output: *CellID of S2*
//Step1: Convert Longitude and Latitude into Radians.
< *Longitude; Latitude*>→*S(Lon, Lat)*
//Step2: Transform Radians into Three-Dimensional Coordinates.
S(Lon, Lat) → *f (x, y, z)*
//Step3: Project Earth's Surface onto Hexahedral Planes.
f(x, y, z)→*g (face, s, t)*
//Step4: Apply Spherical Rectangular Projection Correction.
g (face, s, t)→*h(face, s, t)*
//Step5: Map Points onto a Grid of 30 Layers of Small Squares
h (face, s, t)→*H(face, s, t)*
//Step6: Encode Coordinates Using Hilbert Curve to Generate CellID.
H(face, s, t) →*CellID*
Return *CellID*

The selection of an appropriate S2 level must be based on the dataset's spatial resolution. Table 3 shows the size and number of tiles for some of the different S2 levels, where k stands for thousands and m for millions. We have revised Table 1 shows that the spatial resolution of most infrared hyperspectral sounding points lies between 12 and 16 km. Typically, the coding level should closely correspond to the target resolution to ensure the effectiveness and accuracy of spatial encoding. Setting the S2 level to 10 guarantees complete coverage of the cell space by the corresponding pixels, resulting in a unique S2 ID critical for spatial numerical calculations. However, if the hierarchical structure based on resolution is only used for encoding, we just end up with a bunch of points. Although these points contain some spatial information, their organization and management are insufficient. Given that high-level S2 units can fully encompass low-level S2 units, we can consider high-level S2 units, such as those at level 5, as unified blocks for management purposes. By managing these blocks, the set of points contained within each S2 unit can be effectively organized and overseen. As S2 units represent defined geographic areas, mapping can be achieved using a search filter based on unit addresses, negating the necessity for prior knowledge of S2 data unit distribution.

Table 3. Size and number of tiles of different S2 levels (partial).

S2 Level	**Cell Area/km^2**			**Random Cell/km**		**Number of Cells**
	Min	**Max**	**Avg**	**Min**	**Max**	
4	231,524	413,918	332,074	432	609	1536
5	53,798	104,297	83,018	210	298	6 k
6	12,948	26,113	20,754	108	151	24 k
7	3175	6529	5188	54	76	98 k
8	786	1632	1297	27	38	393 k
9	195	408	324	14	19	1573 k
10	48.78	102	81	9	9	6 m

2.5. Hbase and HDFS Based on Container Management

In the contemporary big data landscape, the efficiency of data storage and processing is essential for effective data management and analysis. HBase and HDFS, as pivotal components of the Apache Hadoop framework, deliver robust distributed storage and efficient querying capabilities [21]. However, with the advancement of cloud computing and containerization technologies, integrating HBase and HDFS with Kubernetes (K8s) for managing their containerized deployment presents distinct advantages, such as improved scalability and resource management [40]. As a container orchestration platform, Kubernetes facilitates enhanced resource management and scheduling efficiency, particularly by optimizing the allocation of computing resources. Containerizing HBase and HDFS enables these services to be quickly initiated and terminated, thereby significantly increasing system flexibility and responsiveness. Additionally, Kubernetes provides auto-scaling capabilities, which allow HBase and HDFS to dynamically modify their resource configurations based on workload variations, thereby optimizing overall performance [41]. In a containerized environment, the nodes of HBase and HDFS can be dynamically scaled up or down. In the case of service failures, specific services can be swiftly restored, enabling the system to adapt in real time to operational changes and ensuring high availability of data querying and storage. This capability significantly reduces response times and enhances data access speeds when processing large volumes of data.

This research utilizes HBase, a column-oriented database, for data storage, effectively overcoming the shortcomings of traditional relational databases in managing large-scale data, while also reducing the memory wastage prevalent in certain non-relational databases during instances of data loss. HBase supports millisecond-level response times through the use of rowkeys, a unique identifier for data rows, and efficiently integrates with distributed computing engines such as Spark, thus providing significant advantages for the subsequent development and deployment of distributed applications [42]. The design of the HBase database's storage table structure is essential, as it not only affects storage efficiency but also plays a critical role in optimizing data retrieval.

2.5.1. Rowkey Design Based on the BPDS Model

In the design of a data storage system based on HBase, the distribution of rowkeys plays a crucial role in determining the system's performance and achieving load balancing [43]. Generating rowkeys in ascending order of timestamps results in rowkeys with similar prefixes being created within the same time frame. Consequently, the HBase system tends to assign rowkeys sharing common prefixes to a single region server. This automatic partitioning mechanism may lead to uneven distribution of data across region servers, with some servers shouldering heavy loads while others store comparatively less data, thereby failing to achieve optimal load balancing. The occurrence of load imbalance significantly impacts the overall performance and availability of the system. In particular, when processing remote sensing data over extended time intervals, overloaded region servers may be required to handle numerous requests, which can lead to performance degradation and increased response times, ultimately resulting in system unavailability. Conversely, region servers with lighter loads may not fully utilize their resources, leading to inefficiencies and overall diminished system performance [44].

This paper proposes a balanced periodic distribution strategy (BPDS) to mitigate the issue of load imbalance, employing partition codes and periodic codes to optimize the generation of rowkeys. In the context of storing infrared hyperspectral sounding data collected during natural temporal observations, partition codes and periodic codes are employed in combination to ensure that the generated rowkey exhibits strong randomness and uniform distribution characteristics. The proposed algorithm computes grid encodings for the latitude and longitude of various sounding points in infrared hyperspectral imaging,

utilizing the S2 algorithm and mapping them to a defined number of partitions, labeled as N. The formula for calculating the partition code is presented as follows:

$$partitionCode = GridID \bmod N \tag{1}$$

The S2 algorithm, designed for geospatial data, demonstrates robust spatial distribution properties and high resolution, enabling effective mapping of geographic location information into partitioned spaces. Data located within the same geographic area are assigned to the same region, which improves data access locality and enhances reading efficiency. Given the natural temporal growth of remote sensing data and the increasing demand for long-term observation of specific areas, this paper introduces a time code based on the existing partition code to optimize the data's organization and storage structure. The implementation of the time code facilitates effective partitioning of the data across both spatial and temporal dimensions, thereby improving its manageability and accessibility. The design of the time code considers the acquisition density and specific observational requirements of time-series remote sensing imagery. Selecting an appropriate data aggregation period is essential based on the characteristics of time-series remote sensing imagery and their spatial dimensions. For example, remote sensing data collected within the same week can be stored within the same region, facilitating rapid and effective access and processing of relevant data during time-series analysis. The formula for calculating the aggregation period code is presented in the following equation, where T_0 represents a reference time for the system, T_1 denotes the observation time of the data, and C signifies the interval period:

$$periodCode = \frac{T_1 - T_0}{C} \tag{2}$$

The design diagram of the rowkey presented in this study is illustrated in Figure 3 below. This rowkey comprises a total length of 20 bytes and consists of two parts. The first part represents the region, calculated through a data pre-partitioning algorithm that considers both spatial and temporal dimensions. The second part comprises the metadata of the remote sensing data, including GridID, SatID, PlayLoadID, level, time, and SequenceID, which are essential for data identification and retrieval.

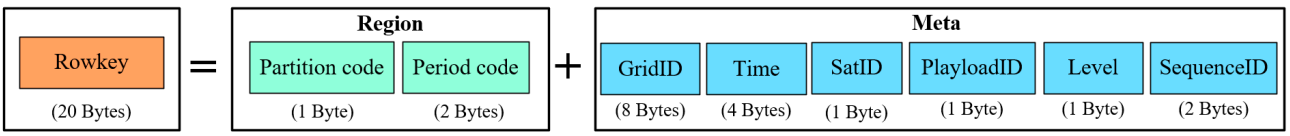


Figure 3. A schematic diagram of the design structure of rowkeys.

2.5.2. The Design of Column Family

Remote sensing data, particularly infrared hyperspectral data, typically appear as high-dimensional pixel matrices. The characteristics of such data impose specific requirements for their storage and processing. To enhance the read and write capabilities of HBase when handling large-scale and high-dimensional remote sensing data, this study defines HBase's column families using the MOB (Massive Objects Storage) paradigm. MOB facilitates the storage of extremely large objects, significantly optimizing the storage and continuous retrieval of infrared hyperspectral data. Moreover, this study utilizes the efficient serialization framework Avro [45] to serialize the data from various remote sensing channels. Avro is a data serialization framework that utilizes a compact format to encode various data types, significantly reducing data storage space and improving storage efficiency. This compactness also decreases resource overhead during data retrieval, enhancing overall performance.

Moreover, the diversity and complexity of remote sensing data necessitate more detailed strategies for data storage and management. According to the storage mechanism of HBase, all column data within a column family are stored contiguously. The infrared hyperspectral band data are typically read in their entirety to facilitate analysis, and based

on this, channel selection can be conducted, enabling the organization of these band data within a unified PixelCF column family as shown in Figure 4. For example, in the case of HIRAS data, the columns of PixelCF can be configured to represent data from different bands, such as MW1, MW2, and LW. These bands represent distinct spectral ranges, each with specific physical characteristics and application contexts. Storing these band data within a single column family not only facilitates batch processing but also improves the efficiency of data retrieval. Furthermore, the MateCF column contains data fields that can be filtered in MongoDB, including various detection angles, detector types, and key attribute information. These fields can be used for both filtering and querying based on specific requirements, as well as for calculations, thus enhancing the flexibility of data utilization.

Rowkey	Column Family:PixelCF				Column Family:MateCF			
	Layer1	Layer2	Layer3	•••	Sat	Date	property1	•••

Figure 4. The table structure of infrared hyperspectral sounding data in HBase.

This paper presents a design for the storage and management of remote sensing data by introducing the Avro serialization framework in conjunction with HBase's storage mechanism, thus facilitating the effective processing of complex data structures. The definition of MOB columns enhances the system's ability to manage large-scale remote sensing data. Moreover, effective column family partitioning and robust metadata management strategies further enhance the efficiency of data retrieval and utilization. This approach not only optimizes the storage mechanism but also achieves effective data compression, resulting in improved efficiency during both storage and transmission, and provides a viable solution for data processing in real-world applications.

2.6. Grid-Oriented MongoDB Structure Design

Although HBase lacks inherent support for secondary indexing, similar capabilities can be realized through the use of additional column families. In the context of row data where pixels serve as the basic unit, the efficiency remains low even with index construction due to the large volume of data. The metadata formats and content from various satellites differ significantly, necessitating a flexible data storage format. MongoDB is a distributed file storage database that lies between relational and non-relational databases. It utilizes bson as its data model, allowing for a flexible data structure capable of accommodating complex data types.

This study leverages the object-oriented features of MongoDB to store data from the same grid at the same time dimension as a data block within a single document. By employing a GeoJSON format for nested documents in accordance with OGC standards to store the metadata for each pixel point, the approach can significantly reduce the number of documents in a MongoDB collection. During area searches, the filtered document count will be smaller, thereby resulting in a significant improvement in query performance. When the block is defined in both temporal and spatial dimensions, a selection process is conducted based on various attributes of the pixels within that block, including sensor type, angle information, whether it is a clear sky point, and the distribution of ground objects, to obtain a set of rowkeys that meet the criteria for the pixel data. Moreover, the attributes of the metadata for point data can dynamically add columns without prior definition, demonstrating considerable flexibility.

The specific processing workflow and database organization are illustrated in the accompanying Figure 5. The figure on the left illustrates the data processing flow, involving the extraction of fundamental metadata for each pixel from the infrared hyperspectral sounding data and their aggregation based on temporal and S2-encoded spatial information. In the middle section of the figure, the database storage format is depicted, where each storage block contains GeoJSON formatted metadata for multiple pixels. This information

can be utilized for various filtering purposes, such as querying and visualizing sampled data for the brightness temperature (BT) column on a map. On the right side of the figure, the MongoDB cluster structure is shown, consisting of the 'Mongos' routing process responsible for handling client requests and directing them to the appropriate shard within the MongoDB sharding architecture. Mongos is responsible for routing client requests to the appropriate shard, while the Replica Set ensures data redundancy and persistence through data replication and automatic failover mechanisms.

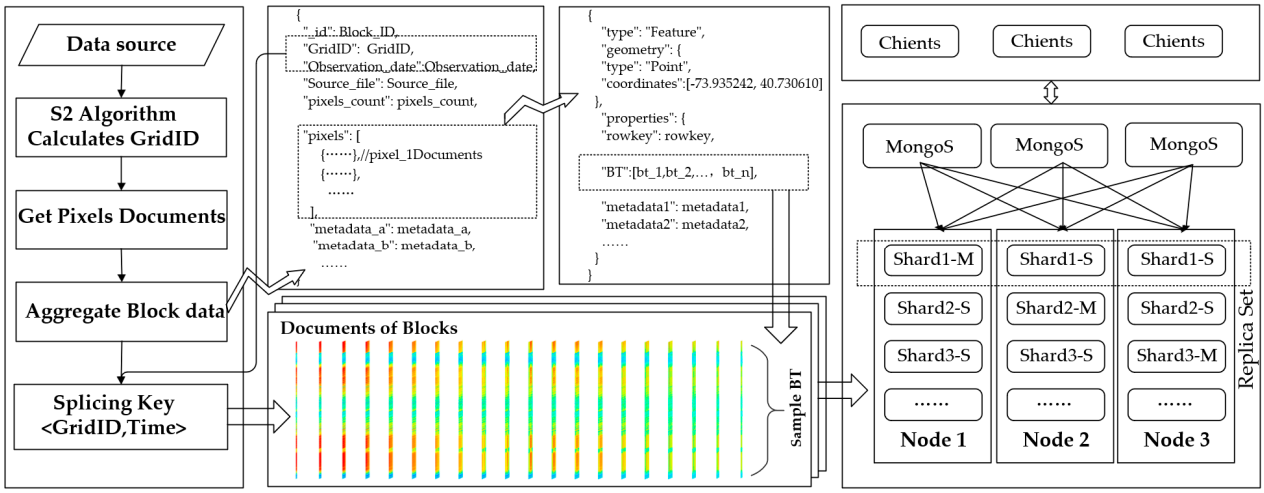


Figure 5. Collections on MongoDB cluster.

3. Results

3.1. Grid-Based Retrieval Efficiency Test

This study analyzes the performance of three geospatial indexing methods: the latitude- and longitude-based GeoHash indexing method, the Google S2 grid indexing method, and the H3 grid indexing method. The GeoHash algorithm [46] implements spatial indexing by dividing the Earth's surface into latitude- and longitude-based units, each uniquely encoded. In conjunction, the 2dsphere indexing method used in MongoDB leverages B+ trees for efficient management and retrieval of GeoHash encodings, significantly enhancing the performance of spatial queries. The H3 method [47] conducts spatial indexing by dividing the Earth's surface into hexagonal grids that are recursively subdivided, with each grid assigned a unique hierarchical code. In contrast, Google S2 divides the Earth's surface into various polygonal grids, utilizing a similar encoding strategy for indexing and querying. In this study, both Google S2 and H3 methods utilized a precomputation hierarchical coding strategy, calculating the encoding for each geographical unit at its respective level prior to data querying. These encodings facilitate indexing and querying within the database, thereby optimizing the spatial data retrieval process.

Figure 6, as follows, illustrates the relationships among various spatial indexing encodings at resolutions exceeding one meter. This figure is depicted in a logarithmic coordinate system. It can be observed from the figure that the GeoHash method possesses a limited number of hierarchical levels, with significant variations in resolution between adjacent levels, whereas the Google S2 encoding demonstrates minimal variation across different scales. This indicates that the utilization of Google S2 encoding facilitates more straightforward joint analysis and processing with various data sources, including tasks such as data mining.

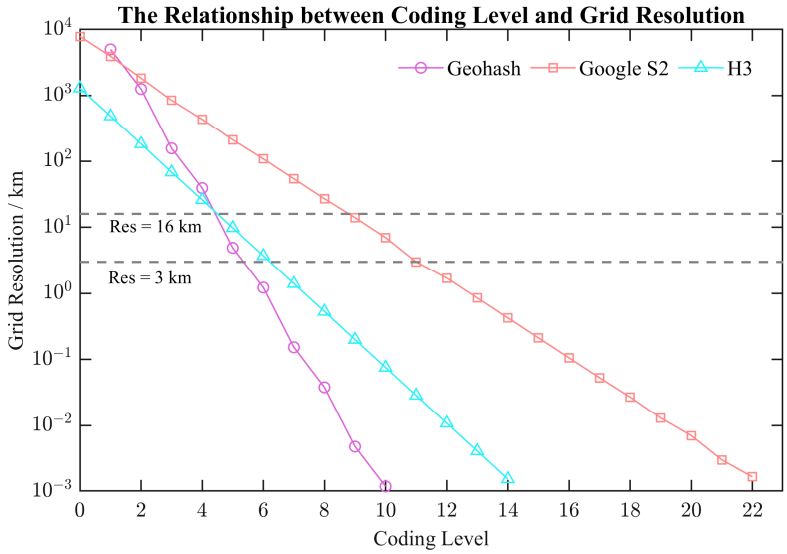


Figure 6. The relationship between coding level and grid resolution.

During the client-side testing phase, we conducted a comprehensive performance evaluation of these three indexing methods. To ensure result accuracy, we performed indexing and loading operations on the designated columns of data located within randomly generated polygonal regions using the same dataset, while maintaining consistency in other relevant conditions. The experiments documented the indexing and loading times for various methods across different data volumes, and these results are illustrated in Figure 7. For data volumes not exceeding 100 k, all three indexing encoding methods exhibit strong performance. However, the GeoHash method requires over 1 s when loading a data volume of 200 k, whereas the Google S2 and H3 methods exceed 1 s only when loading 500 k. As data volume escalates, the time overhead of GeoHash relative to the pre-computed grid-based Google S2 and H3 methods reveals a significant disparity; specifically, at a data volume of 5000 k, the time overhead for Google S2 and H3 is only 35.6% and 38.1% of that of GeoHash. Thus, in the indexing of data points within polygonal regions, pre-calculating the grid encoding at the specified level for that area and then indexing the corresponding data in the database is more efficient than directly using the 2dsphere index, particularly under conditions of high data volume.

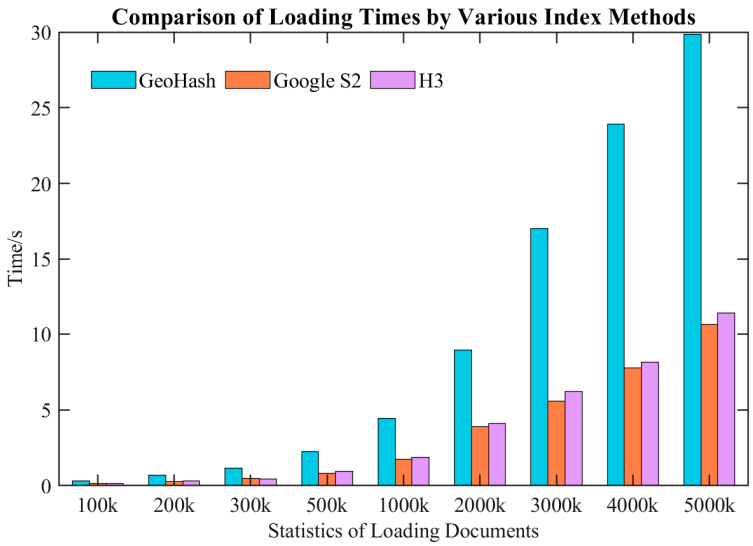


Figure 7. Comparison of loading times by various index methods.

The MongoDB document model stores data in BSON format, facilitating quick retrieval with minimal access time. Compared to relational models that utilize two-dimensional tables, it offers significant advantages in I/O performance, including faster speeds. Building on this foundation, further optimization involves defining pixels within the same grid at the same time as a single block and using blocks that contain similar information for centralized management. This approach reduces the number of indexes and optimizes read efficiency. After empirical testing, the Google S2 spatial grid encoding level was established as five for classifying blocks, while the H3 spatial grid encoding level was set to two. By calculating and retrieving the grids that cover polygonal areas, tests were conducted on both grid-based embedded document models and grid-based flat document models, with the results shown in Figure 8 below.

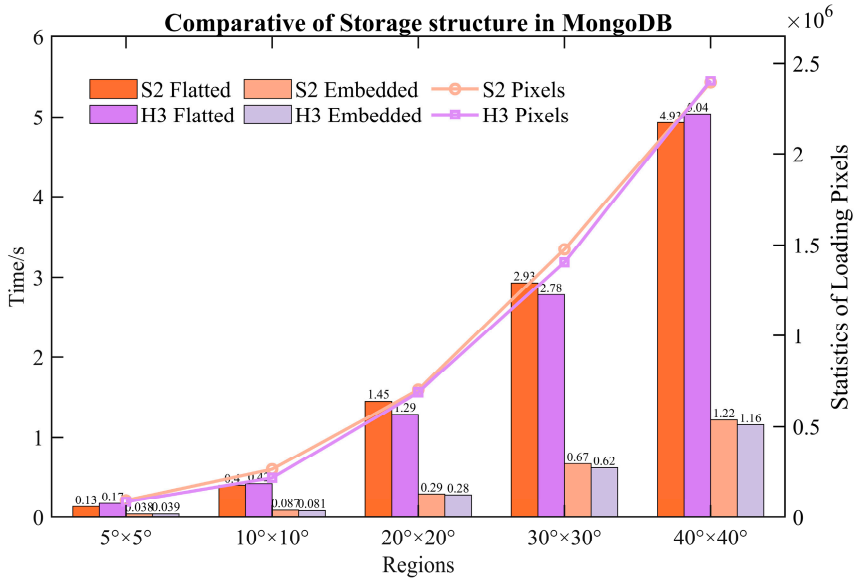


Figure 8. Comparison of storage structure in MongoDB.

Within the same storage architecture, the differences in data indexing and loading times between Google S2 and H3 spatial grid encodings are minimal. The grid-based embedded document model has a loading time overhead for data indexing that does not exceed 25% of that of the grid-based flat document model, indicating a significant enhancement in efficiency. Therefore, in managing highly correlated spatial datasets, the grid-based embedded document model facilitates the centralized storage of related data, thereby minimizing the number of I/O operations required during queries. This study offers critical theoretical insights and practical recommendations for the application of MongoDB in geospatial data management.

3.2. Grid-Based Spatial Region Coverage Frequency Analysis

For this experiment, a one-month dataset of real samples from FY3E/HIRAS-II was utilized. Using both Google S2 and H3 spatial grid encodings, a grid-based embedded indexing model was constructed based on prior research. The frequencies of various regions across the globe were subsequently subjected to statistical analysis. The corresponding findings are illustrated in Figures 9 and 10.

Given that the FY3E satellite is a polar-orbiting satellite, its revisit frequency for the same area in mid- to low-latitude regions is twice daily. In contrast, higher-latitude regions experience multiple revisits and overlaps, leading to a relatively greater number of grid data points. Statistical analysis reveals that the proportion of grids containing over 20,000 pixels in both grid encoding models does not exceed 12% of the total number of grids globally. Even in highly revisited areas, the concentration of very large pixel counts is not predominant, and these data are rarely utilized for research purposes. Furthermore,

the result graphs for both models indicate that in mid- to low-latitude regions, the spatial area coverage based on S2 grids exhibits greater uniformity globally compared to that of H3 grids.

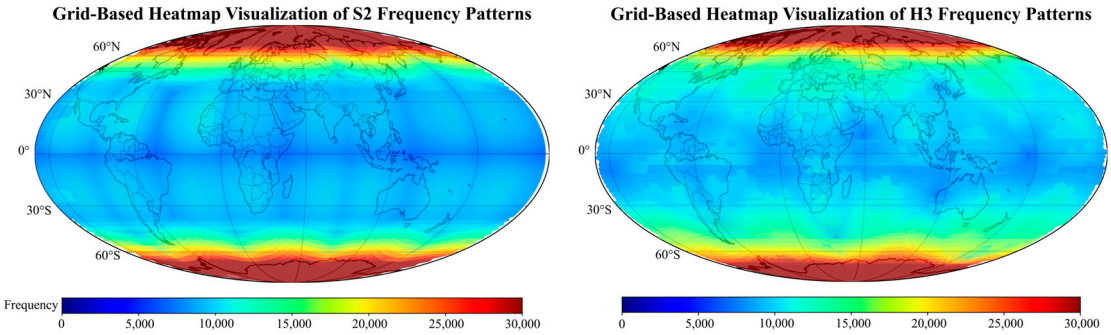


Figure 9. Grid-based heatmap visualization of different methods: (**a**) Google S2 algorithm; (**b**) H3 algorithm.

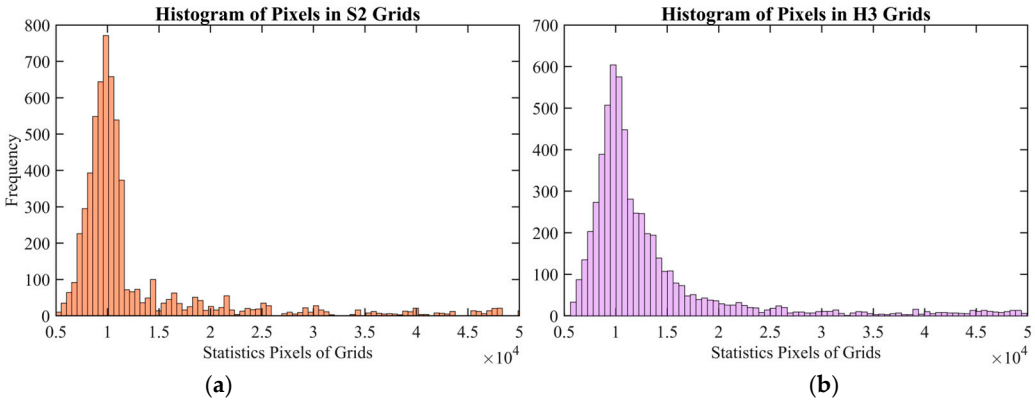


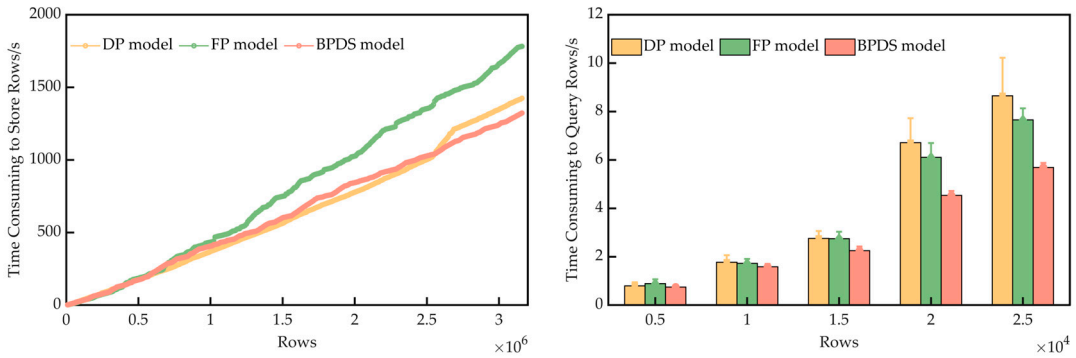
Figure 10. The histogram of pixels of grids in different methods: (**a**) Google S2 algorithm; (**b**) H3 algorithm.

3.3. Analysis of Storage and Read Performance Testing for the BPDS Model

In this investigation, we conducted a comparative evaluation of three storage models at various scales, employing query time and storage time as performance indicators. To analyze data skewness, we compared the number of stored files in diverse regions. The figure below showcases the experimental results for storage time and query time, taking into account three partitioning strategies: the default partition model (DP model), the fixed partition model (FP model), and the balanced periodic distribution strategy model (BPDS model).

The analysis of Figure 11a reveals that the DP model demonstrates the fastest data storage speed for small data volumes. However, as the data volume gradually increases and reaches a specific threshold, the system triggers background services for region splitting. This process has a detrimental effect on subsequent data storage speed, causing delays in storage operations. In contrast, the FP model, due to discrepancies with the actual data distribution, exhibits the longest average storage time. This model stores data in predefined fixed regions, lacking the flexibility to adapt to dynamic data changes, resulting in a gradual increase in storage time as the data volume grows. The BPDS model employed in this study exhibits a consistent storage time characteristic by partitioning data based on spatial and temporal distribution. This model effectively divides the dataset into multiple regions according to its specific characteristics, enabling improved data management and storage. As the data volume increases, the BPDS model maintains a low storage time, thereby demonstrating optimal performance. The key advantage of the BPDS model lies in its sensitivity to the distribution of data. By considering the spatial and temporal features of

the dataset, this model can better adapt to the storage requirements and efficiently manage the dataset.

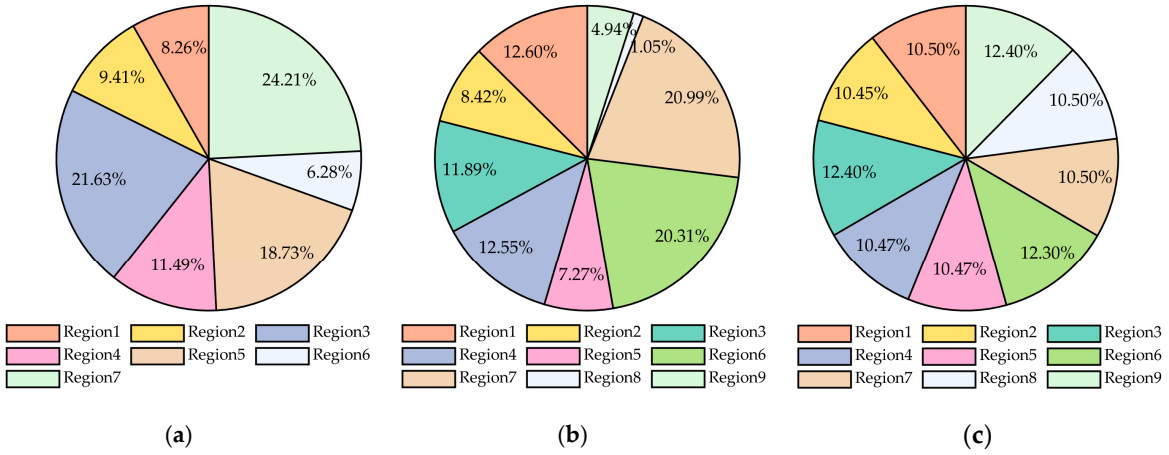


(**a**) (**b**)

Figure 11. Query time and storage time are utilized as comparative metrics to assess performance by different methods: (**a**) storage time and (**b**) query time.

In Figure 11b, we compare the query efficiency of three storage models for different data volumes. For the DP model, the data distribution is solely managed by the region server. While this dynamic management approach can effectively handle changes when the data volume is small, it results in a significant increase in query time as the number of pixel index queries within the region grows. This is primarily attributed to the continuous adjustment of the region server's internal structure to accommodate data growth and the lack of control over the uniform distribution of data across different regions, leading to prolonged query response times. Furthermore, the DP model demonstrates a significant variance in response time, as indicated by the large mean square deviation observed during the query process. This disparity in query performance can adversely affect the overall system efficiency, particularly in scenarios with high-concurrency access, potentially leading to performance bottlenecks. Comparatively, the FP model while exhibiting subpar storage efficiency, outperforms the DP model in terms of query efficiency. This is attributed to the FP model's consideration of specific data distribution characteristics during design, ensuring relatively consistent query performance. On the other hand, the storage model implementing the BPDS model showcases superior query efficiency and reduced query time variance. By dynamically partitioning data based on spatial and temporal attributes, this model adeptly adapts to evolving data distributions, optimizes query pathways, and diminishes query delays. The BPDS model's enhanced flexibility and adaptability notably boost the overall query performance of the system, particularly in managing high-concurrency queries.

Furthermore, the analysis of Figure 12 reveals that regions employing the BPDS model exhibit minimal to no data skewness. In contrast, both the FP model and the model demonstrate varying degrees of data skewness, impacting data storage distribution and query efficiency. In foreseeable scenarios, the BPDS model's partitioning based on spatial and temporal data distributions ensures a predictable and stable number and distribution of regions as time and data volume increase. This strategy evenly distributes data storage and query workload across each node, effectively optimizing the utilization of the cluster's storage and computational resources.



(**a**) (**b**) (**c**)

Figure 12. Distribution of regions in different models: (**a**) DP model, (**b**) FP model, and (**c**) BPDS model.

4. Discussion

In recent years, with the successive deployment of the Fengyun series polar-orbiting satellites into space and their commencement of observations, the HIRAS and HIRAS-II instruments onboard FY-3D/E/F have established an initial global atmospheric sounding network. The utilization of a coordinated multi-satellite observation mode is expected to enhance the coverage and timeliness of data acquisition, thereby further improving data quality and reliability. Future advancements will progressively transition from single-satellite data processing to collaborative observation and processing using multiple satellites. Currently, the processing methods predominantly rely on conventional techniques, while the infrared hyperspectral sounding data holds abundant information that remains largely untapped, presenting substantial potential.

In order to cope with the integrated needs of storage, processing and computing of large-scale data in the future, it is necessary to consider a more flexible, efficient and viable way. Therefore, this paper introduces an efficient distributed storage and indexing method for infrared hyperspectral sounding data, based on satellite data from the Fengyun series polar-orbiting satellites' infrared hyperspectral atmospheric detectors FY-3D/E/F HIRAS and HIRAS-II. Analysis of the data structure reveals the distinctive nature of infrared hyperspectral sounding data—they exhibit greater discreteness and flexibility compared to traditional image-based remote sensing data. Notably, the spatial correlation in current processing methods is not relatively high, so it can be considered as a method to organize geospatial information elements by their midpoints. This study utilizes the Google S2 model, a global discrete grid system, for spatial constraint, and introduces a grid-based unified metadata-embedded document model. By utilizing different types of metadata of embedded documents, hierarchical indexing enables the selection of suitable data. Since the S2 cell representation is well suited for geographic space, mapping geographic space to S2 cells allows for efficient searching using search filters composed of cell addresses. The results demonstrate that the pre-computed grid-based Google S2 method incurs a time cost of only 35.6% compared to the 2D shape-based GeoHash method with B-tree indexing for a data volume of 5000k. Furthermore, the grid-based embedded document model exhibits a loading time cost for data indexing that does not exceed 25% of the grid-based flat document model. Additionally, the Google S2 grid model demonstrates a more uniform distribution across the global grid compared to the H3 method. This paper separates data filtering retrieval from data computation retrieval, highlighting the effectiveness of the grid-based unified metadata-embedded document hierarchical indexing model based on the Google S2 grid for distributed indexing of infrared hyperspectral data.

This paper deploys the HBase and HDFS components of big data distributed storage in containers and utilizes the Kubernetes platform for unified orchestration and management. This deployment approach simplifies scaling and expansion between nodes, enhances

resource utilization efficiency, and increases the flexibility of service deployment. It also enables swift recovery from system interruptions caused by service failures, ensuring real-time adaptation to system changes and maintaining high availability for data queries and storage. Additionally, the framework support provided for future cloud computing service development and deployment enables the implementation of container-based solutions for both the radiative transfer model of infrared hyperspectral data and relevant distributed algorithms. Furthermore, Apache Spark natively supports reading data from HBase and enables distributed computations, providing an efficient solution for data processing. In this experiment, for the distributed storage of infrared hyperspectral sounding data based on all-day time, the BPDS model employed in this paper exhibits a consistent storage time characteristic by partitioning data according to their spatial and temporal distribution. This model effectively divides the data into multiple regions based on their characteristics, facilitating efficient data management and storage. With the gradual accumulation of time and data, the number and distribution of regions are also predictable and stable. The data storage and query pressure are evenly distributed on each node, making full use of the storage and computing resources of the cluster.

In this research, a high-efficiency distributed storage and indexing method for infrared hyperspectral sounding data was proposed based on the satellite data from HIRAS and HIRAS-II on the Fengyun series polar-orbiting satellites FY-3D/E/F. Some initial progress was made through preliminary experiments. However, the current experiment presents several shortcomings and areas for future improvement and research: addressing the method for conducting joint screening of multiple sources of infrared hyperspectral data using this distributed data storage engine and effectively leveraging cloud computing capabilities for distributed data processing to uncover additional latent information.

5. Conclusions

Infrared hyperspectral sounding data hold significant application value in atmospheric remote sensing, but their data scale is huge and complex, and the traditional storage and indexing methods have been unable to meet the demand. The adoption of cloud computing technology effectively addresses the rapid growth of data volumes and the increasingly complex requirements of applications. This paper presents a novel distributed storage and indexing method for efficiently managing infrared hyperspectral sounding data on the cloud. This approach effectively addresses the challenges of data storage and indexing arising from the accumulation of original data, thereby facilitating subsequent distributed computing. This method leverages cutting-edge cloud computing technology by integrating big data components into containerized services to establish distributed cluster services. The unified management is achieved through the cloud-native platform, Kubernetes. By adopting the cloud-native platform and employing the discrete grid spatial indexing algorithm, rapid data storage and efficient retrieval are accomplished, enhancing the effectiveness and accuracy of data screening. In addition, for the distributed storage of infrared hyperspectral sounding data based on all-day time, in the face of the problem of load balancing caused by the accumulation of time and data, the design of rowkeys is optimized by the BPDS model, which can effectively balance the load of nodes in HBase. The research outcomes of this method hold significant implications for the management and application of infrared hyperspectral sounding data. Future studies can further explore the applicability of this method in the storage and indexing of other data types, in conjunction with distributed computing techniques, to further enhance data processing capabilities and efficiency, thereby driving advancements and innovation in the field of infrared remote sensing big data.

Author Contributions: Conceptualization, H.L., Y.H. and M.G.; methodology, H.L.; software, H.L. and G.S.; validation, H.L., G.S. and M.X.; formal analysis, H.L.; investigation, H.L.; resources, H.L. and M.G.; data curation, M.G. and Y.H.; writing—original draft preparation, H.L.; writing—review and editing, H.L., G.S. and M.X.; visualization, M.G.; supervision, M.G. and Y.H.; project administration,

M.G.; funding acquisition, M.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key R&D Program of China, Engineering integration technology of infrared base load, under grant number 2023YFB3905401.

Data Availability Statement: All data generated or analyzed during this study are included in this article.

Acknowledgments: We would like to thank the NSMC (National Satellite Meteorological Center) for sharing FY3D/EF with HIRAS/HIRAS-II data. We would also like to thank Apahe for its open-source software. Additionally, we also would like to thank Mingjian Gu and Yong Hu of the Shanghai Institute of Technical Physics, Chinese Academy of Sciences, for equipment and technical guidance.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Dong, C.H.; Li, J.; Zhang, P. *Principle and Application of Satellite Hyperspectral Infrared Atmospheric Remote Sensing*; Science Press: Beijing, China, 2013.
2. Zhu, J.; Shi, Q.; Chen, F.; Shi, X.; Dong, Z.; Qin, Q. Research status and development trends of remote sensing big data. *J. Image Graph.* **2016**, *21*, 1425–1439.
3. Zhang, S.; Xue, Y.; Zhou, X.; Zhang, X.; Liu, W.; Li, K.; Liu, R. State of the Art High-Performance and High-Throughput Computing for Remote Sensing Big Data. *IEEE Geosci. Remote Sens. Mag.* **2022**, *10*, 125–149. [CrossRef]
4. Wu, Z.; Sun, J.; Zhang, Y.; Wei, Z.; Chanussot, J. Recent Developments in Parallel and Distributed Computing for Remotely Sensed Big Data Processing. *Proc. IEEE* **2021**, *109*, 1282–1305. [CrossRef]
5. Amani, M.; Ghorbanian, A.; Ahmadi, S.A.; Kakooei, M.; Brisco, B. Google Earth Engine Cloud Computing Platform for Remote Sensing Big Data Applications: A Comprehensive Review. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2020**, *13*, 5326–5350. [CrossRef]
6. Tamiminia, H.; Salehi, B.; Mahdianpari, M.; Quackenbush, L.; Adeli, S.; Brisco, B. Google Earth Engine for Geo-Big Data Applications: A Meta-Analysis and Systematic Review. *ISPRS J. Photogramm. Remote Sens.* **2020**, *164*, 152–170. [CrossRef]
7. Liu, Y.N.; Sun, D.X.; Hu, X.N.; Liu, S.F.; Cao, K.Q.; Chai, M.Y.; Liao, Q.J.; Zuo, Z.Q.; Hao, Z.Y.; Duan, W.B.; et al. Development of Visible and Short-Wave Infrared Hyperspectral Imager Onboard GF-5 Satellite. *J. Remote Sens.* **2020**, *24*, 333–344. [CrossRef]
8. Wu, Z.; Li, Y.; Plaza, A.; Li, J.; Xiao, F.; Wei, Z. Parallel and Distributed Dimensionality Reduction of Hyperspectral Data on Cloud Computing Architectures. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2016**, *9*, 2270–2278. [CrossRef]
9. Bouzidi, S. Parallel and Distributed Implementation on SPARK of a Spectral-Spatial Classifier for Hyperspectral Images. *J. Appl. Remote Sens.* **2019**, *13*, 034501. [CrossRef]
10. Ayma Quirita, V.A.; da Costa, G.A.O.P.; Beltrán, C. A Distributed N-FINDR Cloud Computing-Based Solution for Endmembers Extraction on Large-Scale Hyperspectral Remote Sensing Data. *Remote Sens.* **2022**, *14*, 2153. [CrossRef]
11. Zheng, P.; Wu, Z.; Sun, J.; Zhang, Y.; Zhu, Y.; Shen, Y.; Yang, J.; Wei, Z.; Plaza, A. A Parallel Unmixing-Based Content Retrieval System for Distributed Hyperspectral Imagery Repository on Cloud Computing Platforms. *Remote Sens.* **2021**, *13*, 176. [CrossRef]
12. Li, Y.; Qin, H.; Xie, W. HTDFormer: Hyperspectral Target Detection Based on Transformer with Distributed Learning. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 1–15. [CrossRef]
13. Du, Q.; Tang, B.; Xie, W.; Li, W. Parallel and Distributed Computing for Anomaly Detection From Hyperspectral Remote Sensing Imagery. *Proc. IEEE* **2021**, *109*, 1306–1319. [CrossRef]
14. Li, H.; Gu, M.; Zhang, C.; Xie, M.; Yang, T.; Hu, Y. Retrieving Atmospheric Gas Profiles Using FY-3E/HIRAS-II Infrared Hyperspectral Data by Neural Network Approach. *Remote Sens.* **2023**, *15*, 2931. [CrossRef]
15. Xie, M.; Gu, M.; Hu, Y.; Huang, P.; Zhang, C.; Yang, T.; Yang, C. A Study on the Retrieval of Ozone Profiles Using FY-3D/HIRAS Infrared Hyperspectral Data. *Remote Sens.* **2023**, *15*, 1009. [CrossRef]
16. Zhang, C.; Gu, M.; Hu, Y.; Huang, P.; Yang, T.; Huang, S.; Yang, C.; Shao, C. A Study on the Retrieval of Temperature and Humidity Profiles Based on FY-3D/HIRAS Infrared Hyperspectral Data. *Remote Sens.* **2021**, *13*, 2157. [CrossRef]
17. Zhou, X.; Wang, X.; Zhou, Y.; Lin, Q.; Zhao, J.; Meng, X. RSIMS: Large-Scale Heterogeneous Remote Sensing Images Management System. *Remote Sens.* **2021**, *13*, 1815. [CrossRef]
18. Jing, W.; Tian, D. An Improved Distributed Storage and Query for Remote Sensing Data. *Procedia Comput. Sci.* **2018**, *129*, 238–247. [CrossRef]
19. Xu, C.; Du, X.; Yan, Z.; Fan, X. ScienceEarth: A Big Data Platform for Remote Sensing Data Processing. *Remote Sens.* **2020**, *12*, 607. [CrossRef]
20. Yang, B.; Wang, R.; Zhang, W.; Wu, C.; Wang, X.; Meng, L. A Blocky and Layered Management Schema for Remote Sensing Data. *IEEE Access* **2020**, *8*, 99254–99272. [CrossRef]
21. Xiong, Q.; Zhang, X.; Liu, W.; Ye, S.; Du, Z.; Liu, D.; Zhu, D.; Liu, Z.; Yao, X. An Efficient Row Key Encoding Method with ASCII Code for Storing Geospatial Big Data in HBase. *ISPRS Int. J. Geo-Inf.* **2020**, *9*, 625. [CrossRef]
22. Wang, L.; Cheng, C.; Wu, S.; Wu, F.; Teng, W. Massive Remote Sensing Image Data Management Based on HBase and GeoSOT. In Proceedings of the 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Milan, Italy, 26–31 July 2015.

23. Wang, X.; Sun, Y.; Sun, Q.; Lin, W.; Wang, J.Z.; Li, W. HCIndex: A Hilbert-Curve-Based Clustering Index for Efficient Multi-Dimensional Queries for Cloud Storage Systems. *Clust. Comput.-J. Netw. Softw. Tools Appl.* **2023**, *26*, 2011–2025. [CrossRef]
24. Yan, J.; Liu, Y.; Wang, L.; Wang, Z.; Huang, X.; Liu, H. 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 Obs. Remote Sens.* **2021**, *14*, 9350–9363. [CrossRef]
25. Wang, X.; Wang, R.; Zhan, W.; Yang, B.; Li, L.; Chen, F.; Meng, L. A Storage Method for Remote Sensing Images Based on Google S2. *IEEE Access* **2020**, *8*, 74943–74956. [CrossRef]
26. Xu, C.; Du, X.; Fan, X.; Giuliani, G.; Hu, Z.; Wang, W.; Liu, J.; Wang, T.; Yan, Z.; Zhu, J.; et al. Cloud-Based Storage and Computing for Remote Sensing Big Data: A Technical Review. *Int. J. Digit. Earth* **2022**, *15*, 1417–1445. [CrossRef]
27. Wang, S.; Li, G.; Yao, X.; Zeng, Y.; Pang, L.; Zhang, L. A Distributed Storage and Access Approach for Massive Remote Sensing Data in MongoDB. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 533. [CrossRef]
28. Cheng, Y.; Zhou, K.; Wang, J.; Yan, J. Big Earth Observation Data Integration in Remote Sensing Based on a Distributed Spatial Framework. *Remote Sens.* **2020**, *12*, 972. [CrossRef]
29. Yan, J.; Ma, Y.; Wang, L.; Choo, K.-K.R.; Jie, W. A Cloud-Based Remote Sensing Data Production System. *Future Gener. Comput. Syst.* **2018**, *86*, 1154–1166. [CrossRef]
30. Xu, C. A Weighted Linear Regression Model for Precipitation Spatial Interpolation in Altiplano and Mountain Area: A Weighted Linear Regression Model for Precipitation Spatial Interpolation in Altiplano and Mountain Area. *Geo-Inf. Sci.* **2008**, *10*, 14–19. [CrossRef]
31. George, M.; Clerbaux, C.; Hurtmans, D.; Turquety, S.; Coheur, P.-F.; Pommier, M.; Hadji-Lazaro, J.; Edwards, D.P.; Worden, H.; Luo, M.; et al. Carbon Monoxide Distributions from the IASI/METOP Mission: Evaluation with Other Space-Borne Remote Sensors. *Atmos. Chem. Phys.* **2009**, *9*, 8317–8330. [CrossRef]
32. Zhang, X.; Zhang, Y.; Bai, L.; Tao, J.; Chen, L.; Zou, M.; Han, Z.; Wang, Z. Retrieval of Carbon Dioxide Using Cross-Track Infrared Sounder (CrIS) on S-NPP. *Remote Sens.* **2021**, *13*, 1163. [CrossRef]
33. Dussarrat, P.; Theodore, B.; Coppens, D.; Standfuss, C.; Tournier, B. Correction of Calibration Ringing in the Context of the MTG-S IRS Instruments. *arXiv* **2023**. [CrossRef]
34. Yang, T.H.; Hu, X.Q.; Xu, H.L.; Wu, C.Q.; Qi, C.L.; Gu, M.J. Radiation Calibration Accuracy Assessment of FY-3D Hyperspectral Infrared Atmospheric Sounder Based on Inter-Comparison. *Acta Opt. Sin.* **2019**, *39*, 377–387.
35. Yang, T.; Gu, M.; Shao, C.; Wu, C.; Qi, C.; Hu, X. Nonlinearity correction of FY-3E HIRAS-II in pre-launch thermal vacuum calibration tests. *J. Infrared Millim. Waves* **2022**, *41*, 597–607.
36. Zhang, C.; Qi, C.; Yang, T.; Gu, M.; Zhang, P.; Lee, L.; Xie, M.; Hu, X. Evaluation of FY-3E/HIRAS-II Radiometric Calibration Accuracy Based on OMB Analysis. *Remote Sens.* **2022**, *14*, 3222. [CrossRef]
37. Lu, L.; Qi, C.; Zhang, P.; Hu, X.; Gu, M. Noise Estimation of Hyper-Spectral Infrared Atmospheric Sounder Observations Using Principal Component Analysis. *Acta Photonica Sin.* **2019**, *48*, 154–162.
38. Jiang, B.; Zhou, W. Comparative Analysis of GeoHash, Google S2 and Uber H3 as Global Geographic Grid Coding Methods. *Geogr. Geo-Inf. Sci.* **2024**, *40*, 19–28.
39. Jiang, B.; Zhou, W.; Han, H. Storage and Management of Ship Position Based on Geographic Grid Coding and Its Efficiency Analysis in Neighborhood Search—A Case Study of Shipwreck Rescue and Google S2. *Appl. Sci.* **2024**, *14*, 1115. [CrossRef]
40. Cardas, C.; Aldana-Martin, J.F.; Burgueno-Romero, A.M.; Nebro, A.J.; Mateos, J.M.; Sanchez, J.J. On the Performance of SQL Scalable Systems on Kubernetes: A Comparative Study. *Clust. Comput.-J. Netw. Softw. Tools Appl.* **2023**, *26*, 1935–1947. [CrossRef]
41. Poniszewska-Maranda, A.; Czechowska, E. Kubernetes Cluster for Automating Software Production Environment. *Sensors* **2021**, *21*, 1910. [CrossRef]
42. Hu, F.; Yang, C.; Jiang, Y.; Li, Y.; Song, W.; Duffy, D.Q.; Schnase, J.L.; Lee, T. A Hierarchical Indexing Strategy for Optimizing Apache Spark with HDFS to Efficiently Query Big Geospatial Raster Data. *Int. J. Digit. Earth* **2020**, *13*, 410–428. [CrossRef]
43. Sun, Y. Research on HBase Multi-Dimensional Index Construction and Query Optimization Based on Hilbert Space Curve. Master's Thesis, Beijing Forestry University, Beijing, China, 2022.
44. Jiang, Y. Research on Spatiotemporal Big Data Storage and Indexing Based on HBase Database. Master's Thesis, Kunming University of Science and Technology, Kunming, China, 2022.
45. Bhimji, W.; Bristow, T.; Washbrook, A. HEPDOOP: High-Energy Physics Analysis Using Hadoop. *J. Phys. Conf. Ser.* **2014**, *513*, 22004. [CrossRef]
46. Zhou, C.; Lu, H.; Xiang, Y.; Wu, J.; Wang, F. GeohashTile: Vector Geographic Data Display Method Based on Geohash. *ISPRS Int. J. Geo-Inf.* **2020**, *9*, 418. [CrossRef]
47. Woźniak, S.; Szymański, P. Hex2vec: Context-Aware Embedding H3 Hexagons with OpenStreetMap Tags. In Proceedings of the 4th ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery, Beijing, China, 2–5 November 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 61–71.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.