IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 8, 2020, accepted October 25, 2020, date of publication October 28, 2020, date of current version November 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3034466

Research on Data Sharing Architecture for Ecological Monitoring Using Iot Streaming Data

ADAN WU[1,2], (Member, IEEE), JIANWEN GUO[1], AND PENGFEI YANG[1,3]

[1]Key Laboratory of Remote Sensing of Gansu Province, Heihe Remote Sensing Experimental Research Station, Northwest Institute of Eco-Environment and Resources, Chinese Academy of Sciences, Lanzhou 730000, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
[3]College of Geography and Environmental Science, Northwest Normal University, Lanzhou 730030, China

Corresponding author: Jianwen Guo (guojw@lzb.ac.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFC0500105, in part by the National Natural Science Foundation of China under Grant 41801270, and in part by the Foundation for Excellent Youth Scholars of the Northwest Institute of Eco-Environment and Resources, Chinese Academy of Sciences under Grant Y851D41.

ABSTRACT The rapid development of Internet of Things (IoT) technology and the widespread deployment of various sensors around the world have produced a large number of data streams. Thus, current computing systems face the challenge of quickly receiving and managing these large-scale streaming data. This study builds an efficient distributed database based on Greenplum (GP) and focuses on solving the problem of the low efficiency of structured data queries for observed ecological data collected from fragile areas in Northwest China's desert oasis. First, a distributed database is designed and deployed at the physical storage structure level. A database table structure is then established based on the characteristics of the streaming data. On this basis, the data storage strategy is optimized at the data table level. Additionally, the query efficiency of the distributed database is compared with the query efficiency of traditional standalone databases. The results show that the distributed database significantly improves the data query efficiency. The greater the amount of data stored, the better the improvement in efficiency. Finally, based on the optimized distributed database, we develop a data sharing system for streaming data from ecologically fragile areas in the desert oasis in Northwest China, which provides a new approach for the efficient sharing of massive amounts of IoT streaming data for ecological monitoring. Our storage system is still currently working normally, which is highly important to both data managers and users.

INDEX TERMS Ecological monitoring, IoT, greenplum, performance optimization, data sharing.

I. INTRODUCTION

Ecosystems are important parts of the Earth's framework and form the core of its most active biosphere. Since the 1980s, many ecosystem observation and research networks have been established, including regional monitoring networks such as the Global Environmental Monitoring System (GEMS) [1], the Global Terrestrial Observing System (GTOS) [2], the International Long Term Ecological Research Network (ILTER) [3], the Global Flux Observation Network (FLUXNET) [4] and the International Biological Diversity Observation Network (GEO BON) [5], and national networks such as the US Long Term Ecological Research Network (US-LTER) [6], the United Kingdom Environmental Change Monitoring Network (ECN) [7] and the Chinese Ecosystem Research Network (CERN) [8]. All the different types of ecosystems around the world play important roles in monitoring and studying regional ecological, environmental and resource issues.

The rise of the Internet of Things (IoT) [9], [10], [11] has provided strong technical support for improving the level of global ecological monitoring. IoT is ''a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies'' [12]. This new technology substantially reduces the cost of environmental monitoring system deployment and maintenance and effectively improves the

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Hua Chen.

ecological and economic benefits of ecological environmental monitoring [13], [14]. However, it also presents a new challenge for data management: large-scale data are continuously generated over short timespans. According to a published report, the total amount of global Earth monitoring data is increasing exponentially each year, and the International Data Corporation (IDC) predicts that global data might reach approximately 163 ZB by 2025. For example, for the Heihe river basin, four large-scale comprehensive observational studies, including the Heihe Basin Field Experiment (HEIFE) [15], Watershed Allied Telemetry Experimental Research (WATER) [16], a prototype watershed observing system [17], and Heihe Watershed Allied Telemetry Experimental Research (HiWATER) [18], have been carried out in the past 20 years, and nine observation areas and 362 observation points have been established in the upper, middle and lower reaches of the Heihe River, including 385 data collectors and 2145 sensors, since 2012. Thus, 10 million observation records are generated daily (approximately 455 MB/day, 159 GB/year). Except for image and video data used to monitor animals and plants, all other data from environmental monitoring networks are structured data characterized by high speed, infinity and uncertainty. These large, continuous, fast, and time-varying data are called streaming data [19], [20], [21] and possess big data features (volume, velocity, variety, value and veracity) [22].

A database can be a powerful tool for providing efficient, convenient, and safe multi-user storage of and access to massive amounts of streaming data, but traditional databases such as MySQL, SQL Server, and Access are online transactional processing (OLTP) relational databases, which are designed for insertion, modification or deletion of small numbers of records in the databases. Therefore, using traditional data storage methods to manage these streaming data has several drawbacks:

(1) Storage bottleneck: Streaming data are massive; however, traditional relational databases generally use one server to store data and have poor scalability.

(2) Poor query efficiency: The performance of traditional relational database systems decreases sharply as the data size increases.

(3) Poor visualization: In the face of such massive data, traditional relational databases have difficulty achieving rapid mining and visualization [23].

To overcome the problems above that arise with large-scale data housed in a single database, we turn to distributed databases. In this paper, we adopt a geoscience field (IoT ecological monitoring) as an example to study a new, open source data management architecture to achieve massive data sharing and services in the context of earth science big data. The focus of this study is to solve the problem of the poor efficiency of structured data queries through optimization on Greenplum (GP). The main contribution of our work is a data sharing architecture for ecological monitoring-oriented IoT streaming data. The system has the following merits:

(1) Optimization strategy

We introduce GP into the application of ecological monitoring IoT and optimize the query efficiency from the database physical structure and database table structure according to the characteristics of ecological monitoring streaming data. This strategy solves the problem of data query bottlenecks in the case of a limited number of servers. Moreover, with the rapid growth of streaming data, our server and table structure can be dynamically expanded, which does not affect ongoing operations. Finally, this strategy is general and can be used for other application scenarios such as hospitals, education, banks, and traffic where massive amounts of structured data must be shared.

(2) Automatic data parsing and warehousing middleware

We develop an automatic data parsing and warehousing middleware for multi-source heterogeneous data. This middleware can automatically collect streaming data from different types of remote wireless sensor network (WSN) [24] observation instruments and store the data in GP after normalization with no need for any human intervention.

(3) Data-sharing mode

An improved strategy to share data based on cloud disk is proposed, where the query results are directly generated into files in the master node of GP. These data can then be download by users through cloud disk service, which avoids the problem of a long waiting time.

This paper is organized as follows. Section II discusses related work. Section III presents the design, deployment and optimization of a distributed database. Section IV discusses an application example of a data sharing system for IoT. Finally, a brief conclusion and prospects are given in section V.

II. RELATED WORKS

A. IoT APPLICATIONS

IoT is expected to be a major producer of big data. A timely analysis of big data acquired from IoT and other sources to enable highly efficient, reliable and accurate decision making and management of ubiquitous environments is an ambitious challenge. To break through this bottleneck, some approaches and flexible architectures have been developed for time-critical applications. Štefanič *et al.* presented a new concept for engineering complex adaptable cloud systems with time-critical constraints: the application-infrastructure co-programming model. They described the architecture, design and implementation of the system components and how such tools were applied to three time-critical real-world use cases [25]. Koulouzis *et al.* presented a microservice-based infrastructure optimization suite, the Dynamic Real-Time Infrastructure Planner (DRIP), which was used to construct virtual infrastructures for research applications on demand. Their results clearly showed the value of integrated systems such as DRIP for dynamic optimization of data services in research support environments and how they might be used for a number of similar applications involving distributed services and large, dynamic data sets with further investigation and

TABLE 1. Comparison of mainstream distributed databases. "—" indicates nonsupport, "•" indicates support.

Database Features	GP	Vertica	Sybase IQ	Teradata Aster	Oracle Exadata
Shared-nothing architecture	•	•	•	•	—
Data are stored by columns	•	•	•	•	•
Data are stored by rows	•	—	—	•	—
Table partition	•	—	•	•	—
Table index	•	—	•	•	—
Structured data	•	•	•	•	•
Online system expansion	•	—	—	—	•
Linear performance expansion	•	•	—	•	—
Open source	•	—	—	—	—

development [26]. Yang *et al.* proposed a lightweight distributed access control system with an efficient keyword search function to secure the data management in health IoT in 2019 [27]. Alelaiwi evaluated available IoT databases in an edge/cloud platform by applying the analytic hierarchy process (AHP) and suggested a suitable approach for developing a database application. They found that FileMaker was the best choice because it offers the best usability, portability, and supportability for IoT scenarios [28].

In addition, some researchers have attempted to determine how to effectively collect sufficient data while not increasing the amount of redundancy in big data applications. To accomplish this task, Liu *et al.* proposed a matrix-completion-based sampling points selection joint intelligent unmanned aerial vehicle (UAV) trajectory optimization (SPS-IUTO) scheme for data acquisition [29]. Huang *et al.* proposed a novel baseline-data-based verifiable trust evaluation (BD-VTE) scheme to guarantee security at a low cost that has been used for effective trust evaluation, reasonable incentivization and path adjustment. Through the establishment of this data collection scheme, the cost of data collection becomes more reasonable, the data collection rate and accuracy are further improved, and the security is more effectively guaranteed [30]. Ren *et al.* proposed a data relay mule-based collection scheme (DRMCS) to enhance the data collection rate in a vehicular network for opportunistic communication [31].

B. DISTRIBUTED DATABASE

A distributed database [32] refers to the fusion of a database and network, which can effectively address the performance problems caused by high concurrent user access. In the context of the current big data era, distributed databases have gradually replaced centralized databases and have become the mainstream data analysis system for meeting the needs of large-scale data processing and analysis [33], [34], [35]. At the same time, the growth of large-scale data reduces the query efficiency of distributed databases. Thus, improving query efficiency is a popular research topic and a challenging problem in the distributed database field. Li *et al.* designed a distributed storage system-based massively parallel processing (MPP) architecture to solve the shortcomings of low concurrency and poor scalability and accelerate the query of the entire project to improve project performance. They concluded that with its good scalability and MPP advantage, the system can be used to solve the mass data storage problem [36]. Yang *et al.* proposed a novel system with distributed secure data management and keyword search for health IoT. Since patients are usually managed by diverse medical institutions, the proposed system enables distributed access control to protected health information (PHI) among different medical domains [37].

At present, the mainstream distributed databases include GP (https://greenplum.org/), Vertica (http://www.vertica china.com/), Sybase IQ (http://infocenter.sybase.com/help/index.jsp), Teradata Aster (https://www.tableau.com/) and Exadata (https://www.oracle.com/index.html). We compare and summarize the main features of these databases as presented in Table 1. Considering the cost performance factor of each database shown in Table 1, we find that GP is a strong fit to provide a suitable solution to address the poor data sharing efficiency of streaming data produced by IoT through its attractive features, such as a shared-nothing architecture, online system expansion and an open source design. At present, several theoretical and practical approaches in the literature propose data processing and query optimization based on GP.

Waas *et al.* discussed trends and challenges for data warehousing beyond conventional application areas. In particular, they discussed how a massively parallel system such as the GP database can be used for MapReduce-like data processing [38].

Rajput *et al.* performed a comparative study of two databases: GP and Oracle Exadata. They focused on the efficiency, complexity and capacity of both databases and provided insight into their advantages and disadvantages but did not draw any conclusion regarding the superior database [39].

Antova *et al.* presented optimization techniques for queries over partitioned tables as implemented in the GP database. Through several experiments, they demonstrated that the resulting query plans distinctly outperform conventional query plans in a variety of scenarios [40].

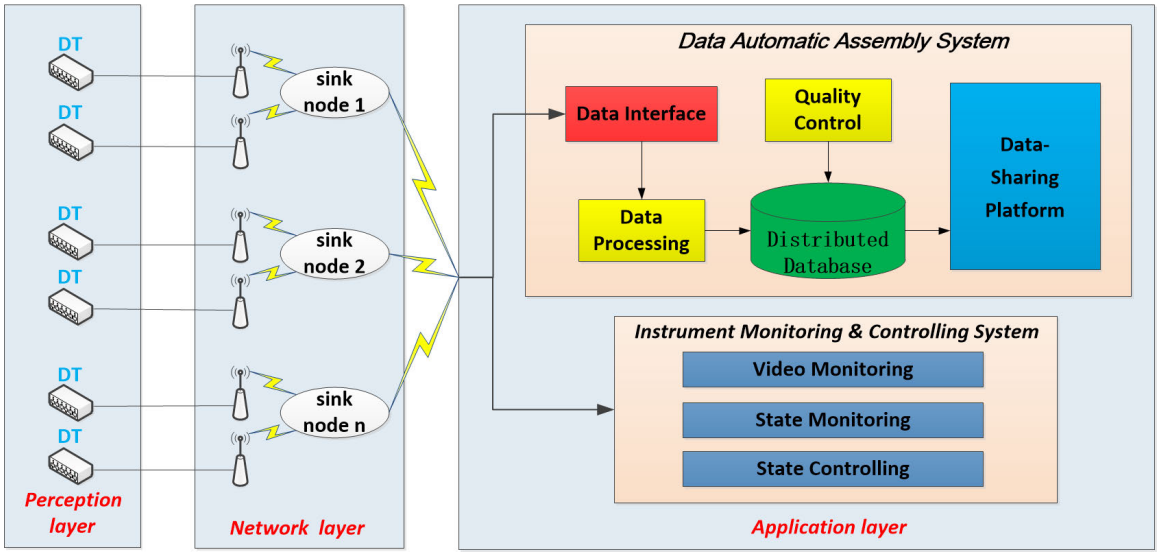

FIGURE 1. The architecture of the ecosystem monitoring IoT.

Zou *et al.* proposed a migration scheme that comprehensively considered user service and the efficiency of migration in the GP database cluster. They conducted experiments on the GP database, the results of which showed the effectiveness of the proposed migration scheme [41].

Raghavan *et al.* proposed a GP database platform extension framework (PXF) that supports parallel high-throughput data access and federated queries across heterogeneous data sources, enabling users to efficiently query large datasets from multiple external sources, without requiring those datasets to be loaded into GP [42].

Patel *et al.* provided a brief overview of GP for Kubernetes's architecture and discussed its implementation. They also demonstrated a full lifecycle of managing a cluster from birth to retirement, including scale-up and self-healing all with minimal database administrator inputs [43].

Albertini *et al.* covered the PXF to fetch and transform images from GP and exploit graphics processing unit (GPU) cycles to run Apache MADlib deep learning methods [44].

According to existing studies on GP, many optimization strategies for GP have been proposed by researchers, but different researchers' schemes focus on different factors and can be applied to different application scenarios, such as banks, hospitals, traffic and social media big data. However, no GP database application cases in the field of ecological monitoring IoT are available.

III. SYSTEM OVERVIEW

A. THE ARCHITECTURE OF THE ECOSYSTEM MONITORING IOT

The IoT ecosystem monitoring architecture [45], [46] contains three layers: a perceptual layer, a network layer, and an application layer. The perceptual layer is responsible for perceiving and acquiring various ecological monitoring data. The network layer is responsible transmitting the acquired data from the perceptual layer, mainly through a variety of wireless transmission methods. The application layer lies at the top level of the information system and includes automatic data collection and normalization, data storage and management, data sharing, data analysis, and other application functions (Fig. 1).

The main goal of this paper is to use an improved distributed database technology to increase the efficiency of data retrieval (the green and light blue parts in Fig. 1)

B. GP ARCHITECTURE

GP is a distributed data warehouse based on PostgreSQL that distributes tasks to multiple node server hosts for transaction management and processing. In terms of user experience, GP is similar to traditional databases, but essential differences exist in its task processing. The GP data warehouse is a database software system based on MPP and a completely shared-nothing architecture [47], [48]. GP is composed of three main parts: a master node, segment nodes, and an interconnect (Fig. 2). The master node is the entry point to the GP database system. It accepts client-side connections and SQL statements and distributes workloads to other database instances (the segment instances). The segment node is a standalone PostgreSQL database, and each segment stores and processes a portion of the data. The GP interconnect is responsible for communication between the master node and the segment node.

C. CLUSTER DEPLOYMENT

In this study, we use four high-performance computers to build a GP distributed database to improve the query efficiency of ecological monitoring IoT data (Fig. 3). The number of servers can be dynamically expanded based on storage

TABLE 2. Hardware Configuration.

No.	Brand	Model	CPU core	Thread	RAM	Disk	Purpose
1	DELL	R940	28	56	64GB	600GB	1 master node and 1 standby master node are deployed in this computer
2	DELL	R940	56	112	64GB	9TB	14 segment nodes and 14 standby segment nodes are deployed in this computer
3	DELL	R940	56	112	64GB	9TB	14 segment nodes and 14 standby segment nodes are deployed in this computer
4	DELL	R940	56	112	64GB	9TB	14 segment nodes and 14 standby segment nodes are deployed in this computer

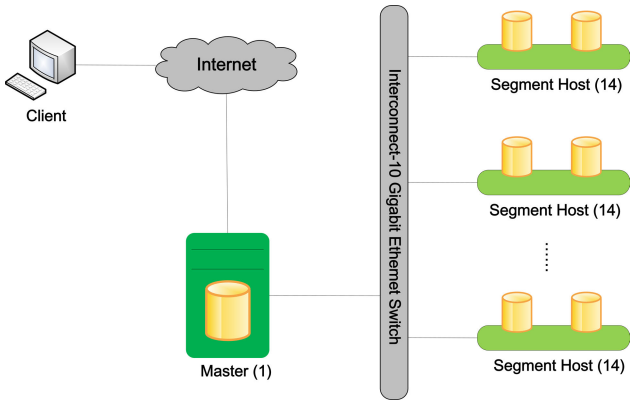

FIGURE 2. GP architecture.

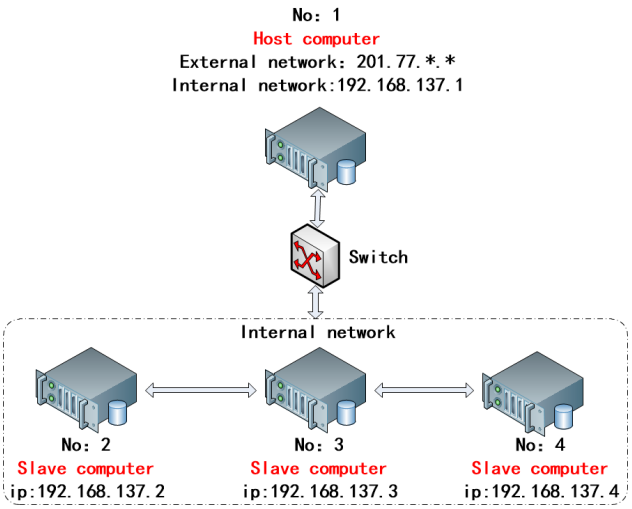

FIGURE 3. Distributed Server Cluster.

requirements. All the data are stored on three slave computers based on a random distribution strategy, and these four computers communicate with each other through the intranet. In addition, external network service is available through the host computer. The hardware configuration is shown in Table 2. The software versions are as follows:

Operating system version: CentOS7.5 (Linux version 3.10.0862.14.4.el7.x86_64)

Greenplum version: GP Database 6.0.0alpha.0+dev.7321 .gb7ce9c4

PostgreSQL version: PostgreSQL 9.1 beta 2

We deploy one master node on the host computer and 42 segment nodes on three slave computers (14 PostgreSQL instances are deployed on each slave computer based on memory impact); that is, when we perform an SQL query, our cluster divides the SQL statement into 42 subtasks performed in different segment instances (each instance monopolizes a CPU core). Namely, the master node dispatches parallel query plans to 42 segments, as shown in Fig. 2. Each segment is responsible for executing local database operations on its own set of data query plans. Most database operations such as table scans, joins, aggregations, and sorts are executed across all segments in parallel. Each operation is performed on a segment database independent of the data stored in the other segment databases.

Query plans are read from bottom to top. In this study, four main steps exist in query plans:

(1) Each segment server is sequentially scanned to access the rows.

(2) Each segment server is aggregated to produce a count of the number of rows from that segment.

(3) Count values are gathered to a single location.

(4) The counts from each segment are aggregated to produce the final result on the master node.

D. DATABASE TABLE DESIGN

Database design refers to constructing an optimal database model for a given application environment, establishing a database and its application system, storing data effectively, and meeting user information and data processing requirements. For the data collected by the IoT ecological monitoring system, we must consider the following important elements and their relationships.

Different projects have different wireless sensor networks. Each observation network consists of multiple observation areas, and each observation area can contain multiple observation stations. Each observation station can be equipped with multiple data loggers, and each data logger can be connected to multiple sensors. Each sensor can acquire multiple variables.

As the above requirements show, the database must store not only basic information about the observation area, observation station, and data logger and sensor but also the observed values. According to the specific application requirements, we design a database table structure. Fig. 4 depicts the relationships between the main tables, and Table 3 describes their fields and uses.

E. OPTIMIZATION STRATEGY

According to the design in the previous section, data can be distributed to 42 different segment hosts in our MPP

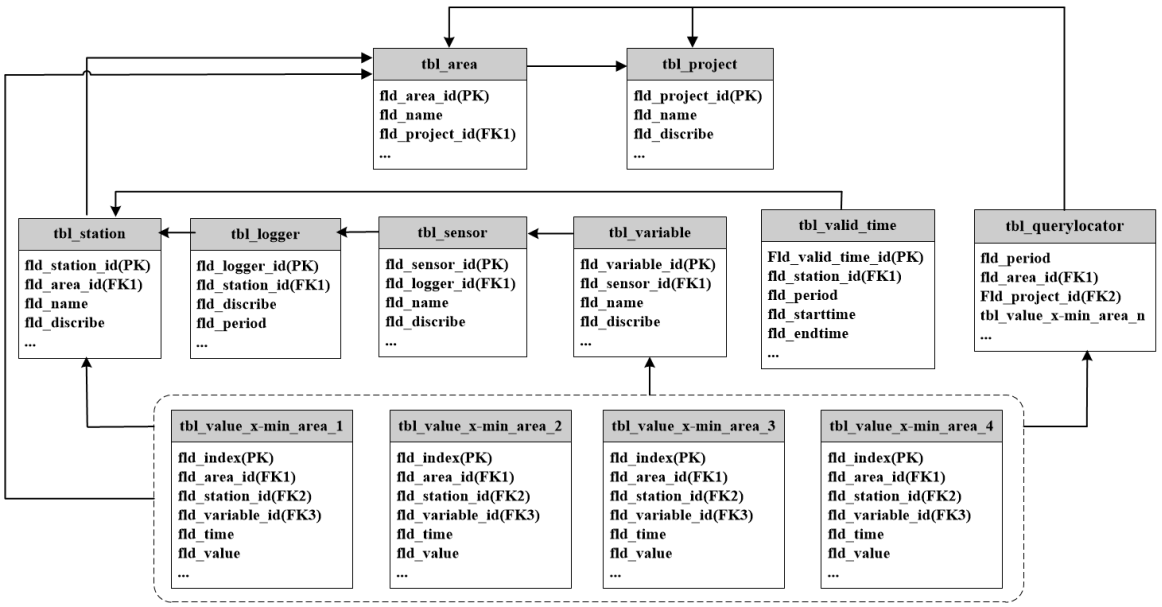

FIGURE 4. Relationships in the main database table.

TABLE 3. The functions of the main tables.

Table name	Description of data table function
Tbl_project	Different experiments belong to different projects, and this table is used to store relevant project information such as name, time, unit and so on.
Tbl_area	For ecological monitoring IoT, different electronic devices are installed in different research areas; thus, the information is very important.This table holds all information about relevant research areas and is associated with the project information.
Tbl_station	An observation station is set up for a sample observation of some item. The user always finds data by station name.
Tbl_logger	This table is mainly used to save transmission instruments .
Tbl_sensor	A sensor is a device that detects or measures a physical property and records, denotes, or otherwise responds to it.
Tbl_variable	Environmental variabilities such as temperature, humidity, soil moisture, wind speed and direction data are stored in this table.
Tbl_valid_time	The period during which the observation station collected data is recorded in this table.
Tbl_value	All the observed data are stored in the table named tbl_value, which is the most important and most frequently used table in the database.To reduce the scope of data retrieval, when the table is actually created, the table name is added with elements such as project, observation period, and observation areas. For example, tbl_value_stjc_1-min_Area_1 is used to store all observation data within Area_1 for a period of 1 min.This study contains 16 such data tables; these take full advantage of observation periods such as 1 min, 10 min, 30 min and other observation factors, including the four study areas.
Tbl_querylocator	When a user selects a period, an observation area and a project in the shareddata system, the program automatically matches the observation table (such as tbl_value_x-min_area_1). That is, all the requested data are retrieved from this table

system. Meanwhile, different data (data are divided by period or area) are stored in different tables. Although these two strategies substantially improve query efficiency and have high scalability, the data query efficiency still has bottlenecks due to the massive amounts of streaming data. To improve this situation, the query efficiency of the observed data must be further optimized using partitions [49]–[51] and indexes [52].

Data partitioning is a physical database design technology whose purpose is to reduce the total amount of data read and written during execution of a specific SQL operation, which reduces the response time. The goal of data partitioning is not to generate new data tables but to evenly distribute the existing data of the table to different hard disks, systems, or different server storage mesons. In our system, the time field is used to perform partitioning on a monthly basis (Fig. 5). The partition time rangs from 2012/01/01 to 2032/01/01, resulting in 241 partition tables (the 241st partition is a store for abnormal data) for each data storage table. When data increase in the future, we can further expand the partition (by adding a new partition to store data collected after 2032).

In addition, an observation data table has a highly selective query in this system, and users often query data based on the observation station. Therefore, the observation station

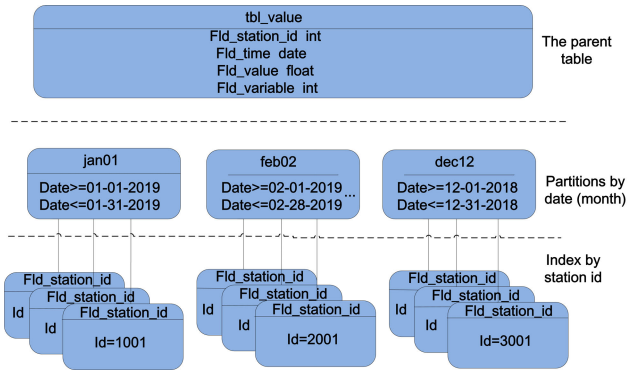

FIGURE 5. Optimization strategy.

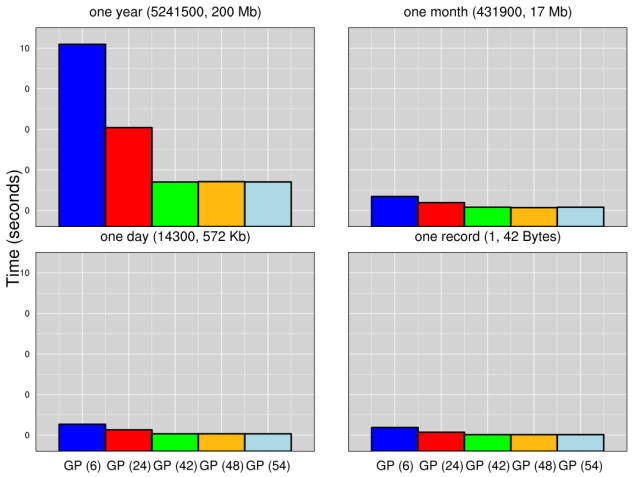

FIGURE 6. Querying performance comparison of GP with different nodes.

data are indexed in this study to improve query performance (Fig. 5).

F. EXPERIMENT

In this section, we first use a large amount of test data to determine the optimal number of segment nodes, then analyze the query process of the distributed database, and finally validate the query performance of the system.

1) DETERMINING THE OPTIMAL NUMBER OF SEGMENT NODES

Before formal application, we have to obtain the optimal number of segment nodes. Therefore, we first create one billion random time series of sensor data and conduct a comparative test on the query performance of the distributed database for five different numbers of nodes (i.e., 6, 24, 42, 48 and 54). As shown in Fig. 6, no matter how much data we query (one record, one day, one month, or one year), GP with more nodes has obvious query advantages. This is because increasing the segment node can make full use of the CPU and IO ability of each data node. However, when the number of segment nodes is greater than 42, the change in query efficiency is minimal (Table 4) because the performance is also limited by our four-server configuration. In general, each node occupies one CPU core and requires 4 GB of memory. Therefore, we deploy 42 segment nodes for our GP database.

2) ANALYZING THE QUERY PROCESS OF 42 SEGMENT NODES

We also use the EXPLAIN ANALYZE command to further analyze the query process of the abovementioned 42 segments, which shows the advantages of splitting subtasks. Fig. 7 shows that an SQL command is divided into several subcommands and executed in different partitions: when querying data for one year (select from tbl_vlue_10min_test where fld_site_id=1 fld_time > ''2010-09-01'' and fld_time < ''20110901''), our SQL statement is divided into 13 sub-SQL statements (the 13 arrows in Fig. 7 show that the query process is performed in 13 different partitions). The first 12 statements read monthly data according to the field fld_time (the data for each month is stored in a different partition), the last SQL statement is used to read the data of the extended partition (it is not executed here, because the data queried are all distributed in the first 12 partitions). When the sub-SQL statement queries data in each partition, it is further divided into 42 query tasks because all the data in the partition are randomly stored in 42 segment nodes (''gather motion 42:1'' means that the 42 segments send results back to the master for presentation to the client), and each segment node queries only the data that meets the conditions locally. The time required for the 42 segment nodes to query a certain partition (in this experiment, one partition corresponds to one month of data) is approximately 0.03-0.05 s (actual time). In the end, the system automatically merges the results of the 12 partitions, and the total time consumed is approximately 2.155 s (total runtime).

3) PERFORMANCE COMPARISON OF THREE DATABASES BY QUERYING TIME

Under the same configuration and query conditions, the query time is the most important indicator of database performance. In this subsection, we compare our desired distributed database with two traditional standalone databases with regard to the querying time. These three types of databases have different characteristics, as follows:

PgSQL: a standalone PostgreSQL instance (all data are stored in one big table called Tbl_value)

PgSQL(optimized): an optimized standalone PostgreSQL instance (different data are stored in different tables, such as Tbl_value_20190610, according to the time required; this database system can create a new table automatically every day)

GP(42): GP with 42 nodes (all data are randomly assigned to different tables and 42 segment nodes, which are installed on four computers)

The comparison results are shown in Table 5. Here, we count only the time required for the query in the local database and not the time to query a remotely connected

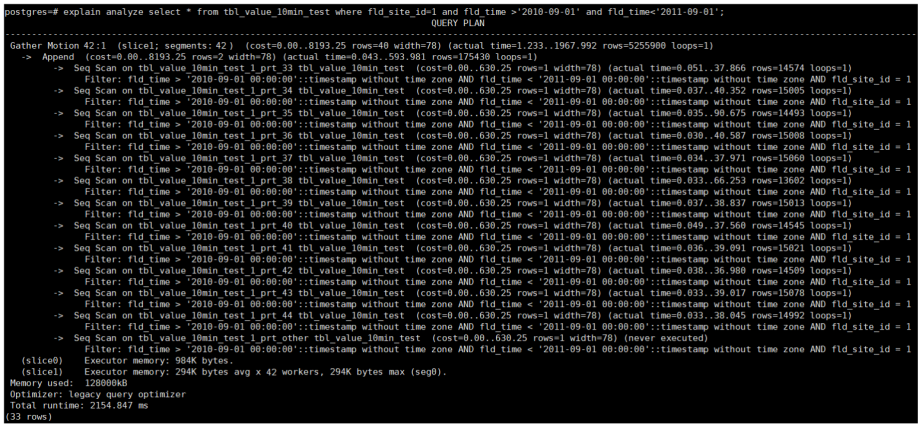

FIGURE 7. Query process analysis of GP (42).

TABLE 4. Comparison of query times for GP with different numbers of nodes (unit: seconds).

Database	Abbreviation	5241500	431900	14300	1
		One year of data	One month of data	One day of data	One day item
GreenPlum distributed database (6 nodes)	GP (6)	12.29	1.032	0.80	0.56
GreenPlum distributed database (20 nodes)	GP (24)	6.13	0.58	0.39	0.21
GreenPlum distributed database (42 nodes)	GP (42)	**2.10**	0.24	0.095	**0.032**
GreenPlum distributed database (48 nodes)	GP (48)	2.13	**0.21**	0.096	**0.032**
GreenPlum distributed database (54 nodes)	GP (54)	2.11	0.24	**0.094**	0.033

database because the query results from different databases can differ significantly based on network conditions.

As shown in Fig. 8, the GP cluster has substantial advantages in data queries, especially when the query data volume becomes very large. When querying one year's data locally, the query efficiency of GP is approximately 190 times better than that of the optimized PostgreSQL. Meanwhile, the efficiency of querying one month's data is approximately 249 times higher than executing that same query on a split-table database. Additionally, optimized databases are available only when querying small-scale data. For a nonoptimized database, querying any data is time consuming, which is not suitable in the big data environment.

4) PERFORMANCE COMPARISON OF THREE DATABASES BY THE TPS

The transactions per second (TPS) is also an important indicator of the system performance and is a computer software and hardware metric that represents the number of transactions completed in 1 s by an information system.

TPS can be calculated with the following formula:

$T \div S = TPS$

where

T = Number of transactions

S = Number of seconds

TPS = Transactions per second

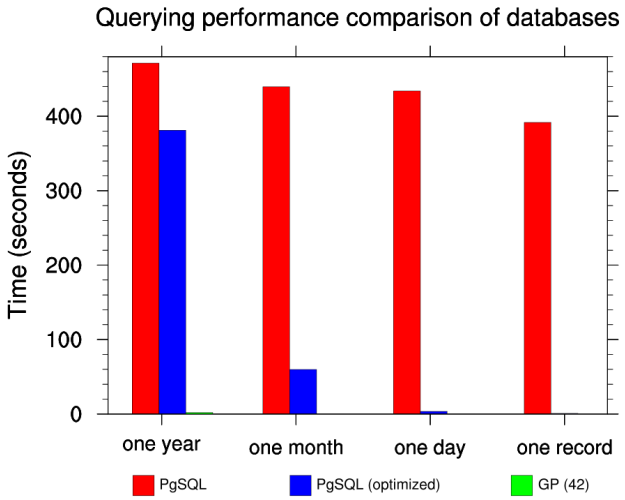

FIGURE 8. Performance comparison of three databases by querying time.

We further compare the performance of GP(42) and PgSQL (optimized) by calculating the TPS. We use Jmeter to simulate the real scenario of data query: 32 users simultaneously query data for four different periods (different query conditions correspond to different transactions; a transaction refers to the process in which a client sends a request to the server and then the server responds), lasting 30 min.

TABLE 5. Comparison of query times for three types of databases (unit: seconds).

Database	Abbreviation	5241500	431900	14300	1
		One year of data	One month of data	One day of data	One day item
PostgreSQL standalone database (not optimized)	PgSQL	471.33	439.65	433.97	391.68
PostgreSQL standalone database (optimized by partitioning)	PgSQL (optimized)	381.14	59.85	3.50	0.82
GreenPlum distributed database (42 nodes)	GP (42)	**2.10**	**0.24**	**0.095**	**0.032**

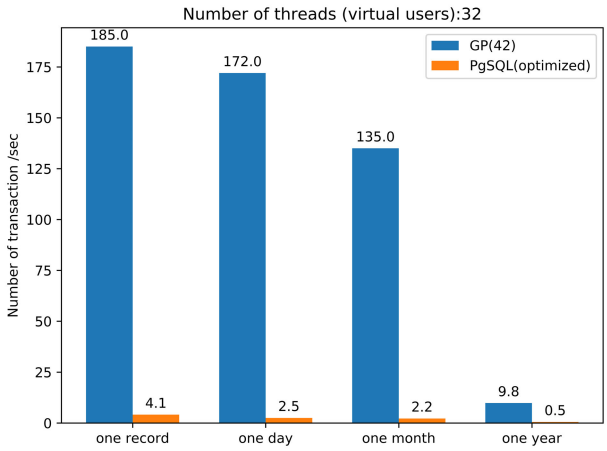

FIGURE 9. Performance comparison of two databases by TPS.

From the analysis results (Fig. 9) GP(42) has higher data reading performance than PgSQL (optimized) in the case of a similar hardware configuration.

IV. IMPLEMENTATION AND DEMONSTRATION APPLICATION

A. STUDY AREA

A project called "Innovative development of equipment and Internet-of-things techniques for ecosystem monitoring and its demonstration" was initiated in 2016 and included three typical fragile ecosystems as field testing and demonstration sites (a karst rocky desertification area in Southwest China, a rare and endangered wild vertebrate reserve—the National Park for Amur Tigers—in Northeast China, and a transition area of desert and oasis in Northwest China). In these areas, the IoT technique was introduced to establish a prototype ecosystem monitoring system in an isolated environment [53]–[55]. In this case, 10 million observation records are generated every day (approximately 455 MB/day, 159 GB/year). Although we previously implemented a PostgreSQL-based data management system to provide users with data services, a single-machine database management method is insufficient to meet the requirements of managing massive amounts of observation data because of the rapidly increasing data volume.

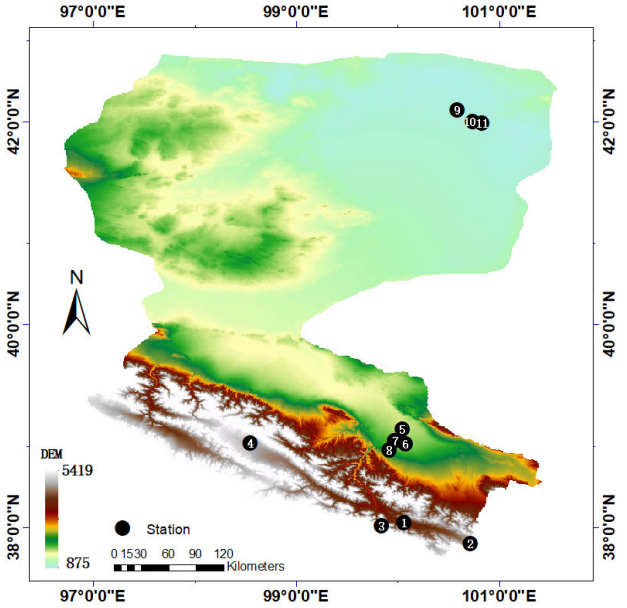

FIGURE 10. Transfer of ecologically fragile areas in the desert oasis in the northwest region.

This study builds a large-scale distributed database based on the transfer of data from ecologically fragile areas in the desert oasis in Northwest China. The goal is to achieve efficient sharing of the collected scientific data. Fig. 10 shows that 11 stations are currently running in desert and oasis transition areas in Northwest China. Three stations are located in the upstream region (Nos. 1, 2, 3 and 4 in Table 6), four stations are located in the midstream region (Nos. 5, 6, 7, and 8 in Table 6), and three stations are located in the downstream region (Nos. 9, 10, and 11 in Table 6) of the Heihe river basin. The collected data include flux data (collected by a vortex correlation instrument and a large-aperture scintillator), meteorological data (collected by an automatic weather station), vegetation parameters (physical flow meter, phenology, and the leaf area index), and soil moisture sensor network data.

TABLE 6. Hydrometeorological observatory stations (Arou, Daman and Sidaoqiao are superstations; the other stations are ordinary stations).

No.	Station	Landscape	Location	Longitude	Latitude	Duration	Number
1	Arou	Subalpine meadow	Upstream	100.46	38.05	Jun 2008–present	38133278
2	Jingyangling	Alpine meadow	Upstream	101.12	37.84	Aug. 2013–present	14155740
3	Yakou	Alpine meadow	Upstream	100.24	38.01	Jan. 2014–present	10467189
4	Dashalong	Marsh alpine meadow	upstream	98.94	38.84	Aug. 2013–present	13928148
5	Zhangye wetland	Reed	Midstream	100.45	38.98	June 2012–present	18838347
6	Heihe Remote Sensing	Grassland	Midstream	100.48	38.83	Aug. 2014–present	11407001
7	Daman	Maize	Midstream	100.37	38.86	May 2012–present	55126323
8	Huazhaizi	Kalidium foliatum desert	upstream	100.32	38.77	Jun 2012–present	11375716
9	Desert	Reaumuria desert	Downstream	100.99	42.11	Apr. 2015–present	9651679
10	Sidaoqiao	Tamarix	Downstream	101.14	42.00	July 2013–present	17550417
11	Mixed Forest	Populus euphratica and Tamarix	Downstream	101.13	41.99	July 2013–present	2419640

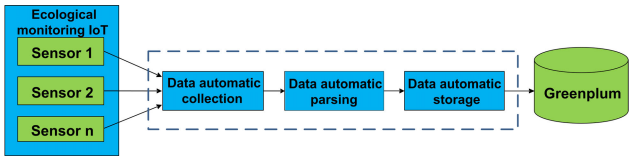

FIGURE 11. Data parsing and warehousing middleware.

B. DATA PARSING AND WAREHOUSING MIDDLEWARE

At present, the IoT acquisition equipment lacks a unified technical standard, which leads to differences in data reception, storage, and wireless transmission protocols for data acquisition instruments of different manufacturers. In this case, automatic acquisition and automatic storage of multi-source heterogeneous data are difficult for ecological monitoring IoT. To address this issue, we develop a multi-source heterogeneous data automatic collection and aggregation middleware. The middleware is composed of three modules: a data automatic collecting module, data automatic parsing processing module and data automatic storage module (Fig. 11). These modules have high cohesion and low coupling and closely cooperate to complete full automatic processing of the monitoring streaming data.

C. WEB-BASED DATA SHARING PLATFORM

In most data-sharing platforms, long-term sequential data are typically stored in the form of files, but this method is not conducive to fine-grained querying of data. Therefore, we build a GP distributed database to store uploaded long-term sequence data and develop an online data visualization platform through which users can query and download data quickly and easily. This system is developed using PHP and libraries, such as openlayer and mapserver. These modules are installed on different computers, such as the web server and WebGIS server (Fig. 12)

The system visualizes the observed data based on the needs of scientific researchers. The visualization tools use open source projects such as Highcharts (statistical charts) and Highstock (time series data display), which are chart libraries written in pure JavaScript. These tools have good

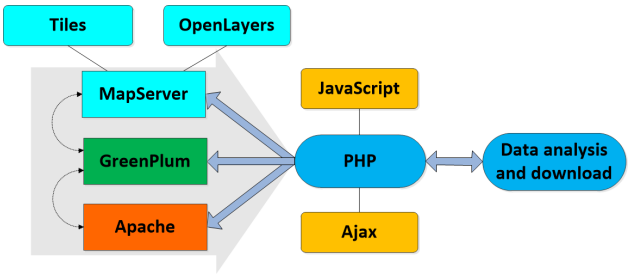

FIGURE 12. Framework of the data sharing platform.

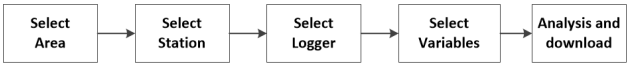

FIGURE 13. Operational processes of visualization.

performance and compatibility, adapt to the device size and provide full support for most current browsers. Using these two tools, we achieve visualization of the 10-min and one-min data. A user views the variable graph according to the logical sequence of the observation station, observation point, data acquisition instrument, and observation variable (Fig. 13). Fig. 14 illustrates that different data can be easily browsed and downloaded through our system according to different condition parameters.

The optimized distributed database provides good query efficiency, but if the query results are directly pushed to the user to download, they are limited by the network bandwidth and the user's browser. For example, approximately 2 s is required to query five million records from one billion observed data records, but if those five million records generate a comma-separated variable (csv) file and then push it to the browser, the process can be very time-consuming because five million records consume approximately 200 MB at a bandwidth speed of 1 MB/s, requiring approximately 200 s to complete the transfer and thus constituting a poor user experience. Moreover, sometimes, the user's browser crashes.

Therefore, based on the amount of query data, we provide two different methods to view the query results (Fig. 15): data requested by the user for less than one month can be

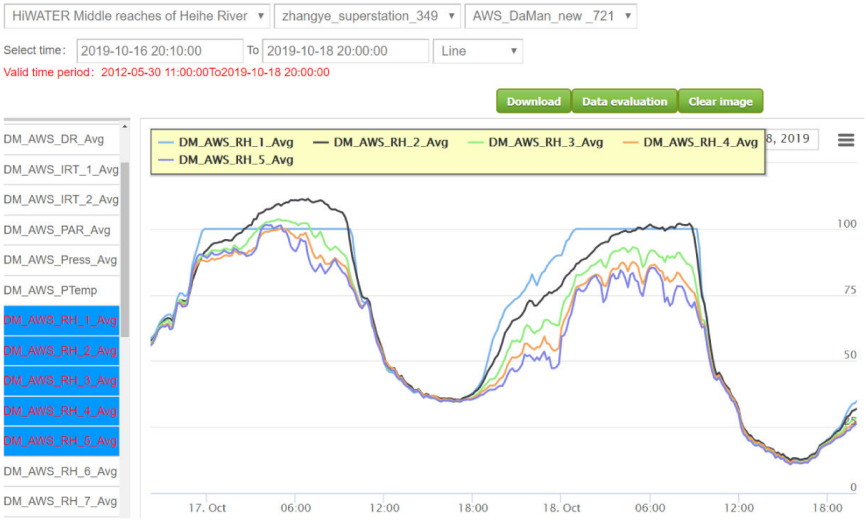

FIGURE 14. Interface of the visualization system.

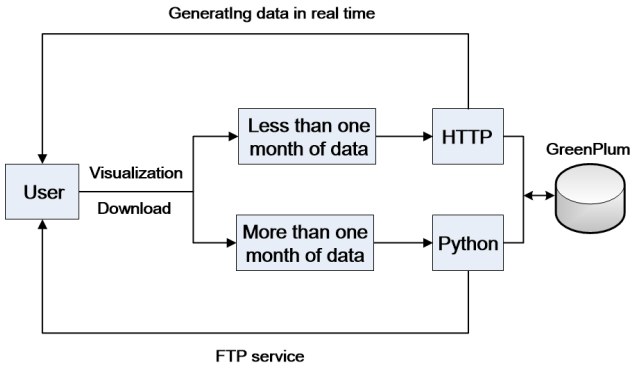

FIGURE 15. Interface of visualization system.

pushed directly to the user via HTTP, while for data longer than one month, we design a background Python program that compresses the retrieved data and sends a cloud disk address to the user, and the user can download the data via FTP, which is quite friendly.

V. CONCLUSION

Using streaming data from ecologically fragile areas in the desert oasis in Northwest China as an example, this study focuses on huge amounts of structured data and conducts data sharing architecture research based on the GP distributed database, which provides a new idea for massive ecological monitoring IoT streaming data. First, we optimize the query efficiency from the database physical structure and database table structure. We then verify the query efficiency of the MPP system. Compared with traditional database systems (Postgresql), our MPP system is easy to deploy and has better query performance. Moreover,our data sharing system supports downloads of massive streaming data and eliminates the impact of network bandwidth, substantially improving the efficiency and experience of users. Although this study takes special ecological monitoring applications as an example, our data sharing architecture can be applied to other fields where structured data must to be shared. In addition, all components of our system are open source; thus, the system is especially suitable for small and medium-sized enterprises and individuals and will be widely used and strongly promoted.

This study focuses on the sharing architecture of structured data (streaming data) without considering the storage of spatial data. In future work, PostGIS will be introduced into the distributed GP, and some spatial query functions will be improved to complete the storage and efficient query of massive remote sensing images.

REFERENCES

[1] M. D. Gwynne, "The global environment monitoring system (GEMS) of UNEP," *Environ. Conservation*, vol. 9, no. 1, pp. 35–43, 1989.
[2] R. R. Christian, "Coastal initiative of the global terrestrial observing system," *Ocean Coastal Manage.*, vol. 46, nos. 3–4, pp. 313–321, Jan. 2003.
[3] K. Vanderbilt and E. Gaiser, "The international long term ecological research network: A platform for collaboration," *Ecosphere*, vol. 8, no. 2, pp. 7–14, 2017.
[4] R. Stöckli, D. M. Lawrence, K. Oleson, G. Y. Niu, K. W. Oleson, P. E. Thornton, Z. L. Yang, G. B. Bonan, A. S. Denning, and S. W. Running, "Use of FLUXNET in the community land model development," *J. Geophys. Res. Biogeosciences*, vol. 113, no. G1, pp. 1–19, 2015.
[5] M. Fernandez, L. M. Navarro, and A. Apaza-Quevedo, "Challenges and opportunities for the Bolivian biodiversity observation network," *Biodiversity*, vol. 16, nos. 2–3, pp. 86–98, 2015.
[6] J. E. Hobbie, S. R. Carpenter, N. B. Grimm, J. R. Gosz, and T. R. Seastedt, "The US long term ecological research program," *Bioscience*, vol. 53, no. 1, pp. 21–32, 2003.
[7] J. M. Sykes and A. M. J. Lane, "The United Kingdom environmental change network: Protocols for standard measurements at terrestrial sites," *Nature*, vol. 138, no. 7, pp. 550–551, 1996.
[8] Y. P. Zhang, "An overview of the Chinese ecosystem research network (CERN)," *J. Agricult. Meteorol.*, vol. 52, no. 2, pp. 181–184, 1996.
[9] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[10] Y. K. Chen, "Challenges and opportunities of Internet of Things," in *Proc. Asia South Pacific Design Autom. Conf.*, 2012, pp. 383–388.
[11] F. Xia, L. Yang, L. Wang, and A. Vinel, "Internet of Things," *Int. J. Commun. Syst.*, vol. 25, no. 9, pp. 1101–1102, 2012.
[12] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.
[13] J. Guo and F. Liu, "Automatic data quality control of observations in wireless sensor network," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 4, pp. 716–720, Apr. 2015.
[14] S. Sen and C. Jayawardena, "Analysis of network techniques and cyber-security for improving performance of big data IoT and cyber-physical communication internetwork," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Feb. 2019, pp. 780–787.
[15] Y. Q. Hu, Y. X. Gao, J. M. Wang, G. L. Ji, Z. B. Shen, L. S. Cheng, J. Y. Chen, and S. Q. Li, "Some achievements in scientific research during HEIFE," (in Chinese), *Plateau Meteor*, vol. 13, no. 3, pp. 225–236, 1994.
[16] X. Li, X. W. Li, and Z. Y. Li, "Watershed allied telemetry experimental research (WATER) datasets are available for open access," *Remote Sens. Technol. Appl.*, vol. 25, no. 6, pp. 761–765, 2010.
[17] X. Li, G. D. Cheng, M. G. Ma, Q. Xiao, R. Jin, Y. H. Ran, W. Z. Zhao, Q. Feng, R. S. Chen, Z. Y. Hu, and Y. C. Ge, "Digital Heihe River Basin. 4: Watershed Observing System," *Adv. Earth Sci.*, vol. 25, no. 8, pp. 866–876, 2010.
[18] X. Li, G. D. Chen, S. M. Liu, Q. Xiao, M. G. Ma, R. Jin, T. Che, Q. H. Liu, W. Z. Wang, and Y. Qi, "Heihe watershed allied telemetry experimental research (HiWater) scientific objectives and experimental design," *Bull. Amer. Meteorological Soc.*, vol. 94, no. 8, pp. 1145–1160, 2013.
[19] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," *Proc. 21st ACM SIGACT-SIGMOD-SIGART Symp. Princ. Database Syst.*, 2002, pp. 1–16.
[20] A. Shukla, T. Sharma, and Y. and Simmhan, "Characterizing distributed stream processing systems for IoT applications," in *Proc. IEEE Int. Conf. High Perform. Comput. Workshops*, 2015, pp. 61–65.
[21] X. Zheng, P. Li, Z. Chu, and X. Hu, "A survey on multi-label data stream classification," *IEEE Access*, vol. 8, pp. 1249–1275, 2020.
[22] A. Katal, M. Wazid, and R. H. Goudar, "Big data: Issues, challenges, tools and good practices," in *Proc. 6th Int. Conf. Contemp. Comput. (IC)*, Aug. 2013, pp. 404–409.
[23] D. Puiu, P. Barnaghi, R. Tonjes, D. Kumper, M. I. Ali, A. Mileo, J. Xavier Parreira, M. Fischer, S. Kolozali, N. Farajidavar, F. Gao, T. Iggena, T.-L. Pham, C.-S. Nechifor, D. Puschmann, and J. Fernandes, "CityPulse: Large scale data analytics framework for smart cities," *IEEE Access*, vol. 4, pp. 1086–1108, 2016.
[24] H. Cheng, Z. Su, N. Xiong, and Y. Xiao, "Energy-efficient node scheduling algorithms for wireless sensor networks using Markov random field model," *Inf. Sci.*, vol. 329, pp. 461–477, Feb. 2016.
[25] P. Štefanič, M. Cigale, A. C. Jones, L. Knight, I. Taylor, C. Istrate, G. Suciu, A. Ulisses, V. Stankovski, S. Taherizadeh, G. F. Salado, S. Koulouzis, P. Martin, and Z. Zhao, "SWITCH workbench: A novel approach for the development and deployment of time-critical microservice-based cloud-native applications," *Future Gener. Comput. Syst.*, vol. 99, pp. 197–212, Oct. 2019.
[26] S. Koulouzis, P. Martin, H. Zhou, Y. Hu, J. Wang, T. Carval, B. Grenier, J. Heikkinen, C. Laat, and Z. Zhao, "Time-critical data management in clouds: Challenges and a dynamic real-time infrastructure planner (DRIP) solution," *Concurrency Comput., Pract. Exper.*, vol. 32, no. 16, pp. 1–17, Aug. 2020.
[27] Y. Yang, X. Zheng, W. Guo, X. Liu, and V. Chang, "Privacy-preserving smart IoT-based healthcare big data storage and self-adaptive access control system," *Inf. Sci.*, vol. 479, pp. 567–592, Apr. 2019.
[28] A. Alelaiwi, "Evaluating distributed IoT databases for edge/cloud platforms using the analytic hierarchy process," *J. Parallel Distrib. Comput.*, vol. 124, pp. 41–46, Feb. 2019.
[29] X. Liu, H. B. Song, and A. F. Liu, "Intelligent UAVs trajectory optimization from space-time for data collection in social networks," *IEEE Trans. Netw. Sci. Eng.*, early access, Aug. 19, 2020, doi: 10.1109/TNSE.2020.3017556.
[30] S. B. Huang, A. F. Liu, S. B. Zhang, W. Tian, and N. X. Xiong, "BD-VTE: A novel baseline data based verifiable trust evaluation scheme for smart network systems," *IEEE Trans. Netw. Sci. Eng.*, early access, Aug. 7, 2020, doi: 10.1109/TNSE.2020.3014455.
[31] Y. Ren, T. Wang, S. Zhang, and J. Zhang, "An intelligent big data collection technology based on micro mobile data centers for crowdsensing vehicular sensor network," *Pers. Ubiquitous Comput.*, pp. 1–10, Aug. 2020, doi: 10.1007/s00779-020-01440-0.
[32] D. J. Dewitt and J. Gray, "Parallel databases systems: The future of high performance database systems," *Commun. ACM*, vol. 35, no. 6, pp. 85–98, 1992.
[33] S. Ghemawat, H, Gobioff, and S. T. Leung, "The Google file system," *ACM SIGOPS Operating Syst. Rev.*, vol. 37, no. 5, p. 29, 2003.
[34] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Trans. Comput. Syst.*, vol. 26, no. 2, pp. 1–26, Jun. 2008.
[35] J. Dean and S. U. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. 6th Symp. Oper. Syst. Design Implement.*, 2004, pp. 137–149.
[36] C. Li, J. Yang, J. Han, and E. Haihong, "The distributed storage system based on MPP for mass data," in *Proc. IEEE Asia–Pacific Services Comput. Conf.*, Dec. 2012, pp. 384–387.
[37] Y. Yang, X. Zheng, and C. Tang, "Lightweight distributed secure data management system for health Internet of things," *J. Netw. Comput. Appl.*, vol. 89, pp. 26–37, Jul. 2017.
[38] M. Waas, "Beyond conventional data warehousing-massively parallel data processing with greenplum database," in *Proc. Informal 2nd Int. Workshop Bus. Intell. Real-Time Enterprise*, vol. 27, 2009, pp. 89–96.
[39] E. Rajput, H. Yadav, and A. Singh, "Comparative study of EMC greenplum and oracle exadata," *J. Eng. Comput. Appl. Sci.*, vol. 2, no. 4, pp. 50–54, 2013.
[40] L. Antova, A. El-Helw, M. A. Soliman, Z. X. Gu, M. Petropoulos, and F. Waas, "Optimizing queries over partitioned tables in MPP systems," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2014, pp. 373–384.
[41] C. Zou, F. Zhao, Y. Xie, H. Zhou, and J. Qin, "Live migration in greenplum database based on SDN via improved gray wolf optimization algorithm," in *Proc. Conf. Res. Adapt. Convergent Syst.*, Sep. 2019, pp. 153–160.
[42] V. Raghavan, A. Denissov, F. Guerrero, R. O. Albertini, B. Divya, L. N. Owen, S. Mani, and L. Jain, "Platform extension framework (PXF): Enabling parallel query processing over heterogeneous data sources in greenplum," in *Proc. ACM SIGMOD Rec.*, 2019, pp. 1–12.
[43] J. Patel, G. Tadi, O. Basarir, L. Hamel, D. Sharp, Y. Fei, and X. Zhang, "Pivotal greenplum for kubernetes: Demonstration of managing greenplum database on kubernetes," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 1969–1972.
[44] O. R. Albertini, D. Bhargov, A. Denissov, F. Guerrero, N. Jayaram, N. Kak, E. Khanna, O. Kislal, A. Kumar, F. McQuillan, L. Owen, V. Raghavan, D. Valdano, and Y. Zhang, "Image classification in greenplum database using deep learning," in *Proc. ACM SIGMOD Rec.*, 2020, pp. 1–4.
[45] F. Alam, R. Mehmood, I. Katib, N. N. Albogami, and A. Albeshri, "Data fusion and IoT for smart ubiquitous environments: A survey," *IEEE Access*, vol. 5, pp. 9533–9554, 2017.
[46] A. Bröring, S. Schmid, C.-K. Schindhelm, A. Khelil, S. Käbisch, D. Kramer, D. Le Phuoc, J. Mitic, D. Anicic, and E. Teniente "Enabling IoT ecosystems through platform interoperability," *IEEE Softw.*, vol. 34, no. 1, pp. 54–61, Jan./Feb. 2017.
[47] F. S. Zeng, "Research and improvement of database storage method," *Appl. Mech. Mater.*, vols. 608–609, pp. 641–645, Oct. 2014.
[48] A. Floratou, U. F. Minhas, and F. Özcan, "SQL-on-Hadoop: Full circle back to shared-nothing database architectures," *Proc. VLDB Endowment*, vol. 7, no. 12, pp. 1295–1306, Aug. 2014.
[49] S. Matalqa and S. Mustafa, "The effect of horizontal database table partitioning on query," *Int. Arab J. Inf. Technol.*, vol. 13, no. 1, pp. 184–189, 2016.
[50] W. Qu and S. Dessloch, "Distributed snapshot maintenance in wide-column NoSQL databases using partitioned incremental ETL pipelines," *Inf. Syst.*, vol. 70, pp. 48–58, Oct. 2017.
[51] C. Huang, H. Hu, X. Wei, W. Qian, and A. Zhou, "Partition pruning for range query on distributed log-structured merge-tree," *Frontiers Comput. Sci.*, vol. 14, no. 3, pp. 16–20, Jun. 2020.
[52] D. Taniar, J. W. Rahayu, and R. B. N. Tan, "Parallel algorithms for selection query processing involving index in parallel database systems," *Comput. Syst. Sci. Eng.*, vol. 19, no. 2, pp. 95–114, 2004.
[53] S. Liu *et al.*, "The heihe integrated observatory network: A basin-scale land surface processes observatory in China," *Vadose Zone J.*, vol. 17, no. 1, pp. 1–21, 2018.

[54] X. Li, N. Zhao, R. Jin, S. Liu, X. Sun, X. Wen, D. Wu, Y. Zhou, J. Guo, S. Chen, Z. Xu, M. Ma, T. Wang, Y. Qu, X. Wang, F. Wu, and Y. Zhou, "Internet of Things to network smart devices for ecosystem monitoring," *Sci. Bull.*, vol. 64, no. 17, pp. 1234–1245, Sep. 2019.
[55] M. Zhang and X. Li, "Drone-enabled Internet-of-Things relay for environmental monitoring in remote areas without public networks," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7648–7662, Aug. 2020.

ADAN WU (Member, IEEE) is currently pursuing the Ph.D. degree from the Northwest Institute of EcoEnvironment and Resources, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing, China. He has developed several information system, such as web-based visualization system for a wireless sensor network in heihe watershed allied telemetry experimental research, decision-making system for preventing and forecasting of disaster in qinghai-tibet permafrost engineering corridor, the new digital heihe river system, data-sharing system for Gansu data and application center, automatic assembling system for observation data in Qinghai, and information service system for Arctic northeast passage. His research interests include the Internet of Things, smart information service, and big data analysis.

JIANWEN GUO is currently pursuing the Ph.D. degree with the Northwest Institute of Eco-Environment and Resources, Chinese Academy of Sciences. He is also an Associate Professor with the Northwest Institute of Eco-Environment and Resources, Chinese Academy of Sciences. He has presided over and completed more than 20 related scientific research projects. He is also the Project Leader, he undertake the research work of Key Technology Research and Development of Ecological Monitoring Internet of Things in the Key Research and Development Project of Ministry of Science and Technology Key Parameters Monitoring Equipment Development and Ecological Internet of Things Demonstration. His research interests include geographic information systems, ecological and environmental monitoring, wireless sensor networks, and data sharing. He is a member of the Theory and Method Committee, China Geographic Information System Association, China Resource Information System Professional Committee.

PENGFEI YANG is currently pursuing the master's degree with Northwest Normal University. He has built distributed database, such as greenplum, and developed several information system, such as siming mountain forest ecological monitoring information system, and southwest university observation data integration system. His research interests include urban change in the direction of big data, data analysis, distributed database, and web service-based on B/S architecture.