Received February 14, 2020, accepted March 4, 2020, date of publication March 9, 2020, date of current version March 25, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2979597

An Efficient Access Model of Massive Spatiotemporal Vehicle Trajectory Data in Smart City

LIANJIE ZHOU, (Student Member, IEEE), QINGQUAN LI, AND WEI TU
MNR Key Laboratory for Geo-Environmental Monitoring of Great Bay Area, Shenzhen University, Shenzhen 518060, China
Guangdong Key Laboratory of Urban Informatics, Shenzhen University, Shenzhen 518060, China
Shenzhen Key Laboratory of Spatial Smart Sensing and Services, Shenzhen University, Shenzhen 518060, China
Research Institute for Smart Cities, Shenzhen University, Shenzhen 518060, China
School of Architecture and Urban Planning, Shenzhen University, Shenzhen 518000, China

Corresponding authors: Qingquan Li (liqq@szu.edu.cn) and Wei Tu (tuwei@szu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2103104, in part by the Shenzhen Scientific Research and Development Funding Program under Grant KQJSCX20180328093453763 and Grant JCYJ20180305125101282, in part by the Open Fund of Key Laboratory of Urban Land Resources Monitoring and Simulation, MNR, under Grant KF-2018-03-004, in part by the Funding Program of the Department of Education of Guangdong under Grant 2018KTSCX196, in part by the China Postdoctoral Science Foundation funded Project under Grant 2019M663071, and in part by the Open Fund of Key Laboratory of Urban Land Resources Monitoring and Simulation, Ministry of Land and Resource, under Grant KF-2018-03-004.

ABSTRACT Daily trajectory data scale of vehicle monitoring networks in smart cities is growing rapidly, reaching daily volumes of 1 billion. Accessing hyper massive spatiotemporal trajectory data (HMSTD) in transport, the Internet of Things, or other fields is difficult and limited based on the current spatiotemporal data index techniques. Therefore, we propose path-divided Hadoop Distributed File System (HDFS) data blocking (PDDB) based on the Apache Impala (PDDB-Impala) method to optimize the efficient access manner of HMSTD to enhance the efficiency of hyper data sharing. Moreover, PDDB parquet data partitioning rules are proposed. In experiments, 35,809 buses equipped with BD positioning sensors, creating 1.03 billion data records each day. The bus distribution in Shenzhen city is collected from 7:00 a.m. to 9:00 a.m. and 11:00 a.m. to 01:00 p.m. Moreover, PDDB-Impala achieves about 8 times, 9 times, 29 times, and 110 times higher performances than those in MongoDB or HBase for data scales of 1 billion, 10 billion, 50 billion, and 100 billion, the results of which outperform those of the equipartition in the Impala, MongoDB, and HBase methods.

INDEX TERMS Tmassive trajectory data sharing, BeiDou positioning sensor, spatial-temporal indexing, Apache Impala.

I. LIST OF ABBREVIATIONS

HDFS	Hadoop Distributed File System
BD	BeiDou
GPS	Global Position System
PB	petabyte
SOS	Sensor Observation Service
IoT	Internet of Things
RAM	Random Access Memory

The associate editor coordinating the review of this manuscript and approving it for publication was Miltiadis Lytras.

II. LIST OF SYMBOLS

$N_{Parquet}$	The data blocks number
RAM_s	The of single node
$Degree_{mean}$	The degree of a partitioned data block
$longitude_{i+1}$	The longitude of road i +1
$latitude_{i+1}$	The latitude of road i +1
$P_{longitudeR}$	The longitude of parameter R
ST_{tn}	The spatiotemporal range of request
$array_{roads}$	Roads interesting with spatial range
$block_{tn\_spatial}$	Data blocks meeting the spatial range
P_{begin}	Observation begin time
$partition_{temporal}$	Data partition meeting the temporal range
$Point_{trajectory}$	Trajectory data in HBase

TABLE 1. Heterogeneous, typical and massive vehicle networks in 2019.

vehicle monitoring network	Tokyo taxi autopilot network	Beijing taxi network	Shenzhen bus network	New York taxi network
Location	Tokyo, Japan	Beijing, China	Shenzhen, China	New York, America
Positioning device	Quasi-Zenith Satellite System	GPS	BD Navigation Satellite System	GPS
Start date	Jan. 2018	Mar. 2017	Aug. 2018	Jun. 2009
Vehicle number	5000	60736	35809	33000
Observation attributes	location, direction, speed	location, direction, speed	location, status, speed	location, direction, speed
Observation frequency (seconds)	0.1	30	30	30
Daily data generated	8.74 billion	1930 million	1.03 billion	984 million

III. INTRODUCTION

Harrison *et al.* [1] in International Business Machines Corporation states that a "smart city" signifies an "instrumented, interconnected and intelligent city." "Instrumented" means the capability of capturing and integrating real life data through sensors, personal devices, appliances, and other similar perception devices. "Interconnected" refers to the integration of these perceptual data into a network computing platform that facilitates the exchange of heterogeneous data among the heterogeneous web services. "Intelligent" means the combination of complex analytics, spatiotemporal data modelling, mining, association, and visualization to make better intelligent decisions. Through the "Instrumented" stated, the sensors are ubiquitous, such as buses, private cars, bicycles and electric motorcars, or even persons in transportation. In the meantime, precision and reliability in positioning devices have been promoted dramatically. Simultaneously, mobile positioning services have become pervasive around the world [2]. As "Interconnected" states, the data into computing platform results in the difficulty of efficient vehicle trajectory data management [3], [4].

The Fifth Generation of Mobile Communications technology will lead to the rapid growth of scenario cases demanding higher data access capacity and supporting the data rate with a lower latency, higher frequency, and faster and more scalable network, providing better user Quality of Experience [5]–[7]. At present, a large number of vehicle monitoring networks are springing up, aiming to provide helpful public travel service in different urban cities, even including Volunteered Geographic Information networks [8]. Table 1 shows several existing larger vehicle monitoring networks, containing the Tokyo taxi autopilot network, Shenzhen bus network, Beijing taxi network and New York taxi network. As described, the data scale of the dataset created each day in the Tokyo taxi autopilot network is approximately 8.74 billion; the data scale in the Beijing taxi network is approximately 1.93 billion; the data scale in the Shenzhen bus network is approximately 1.03 billion; the data scale in the New York taxi network is approximately 984 million. Sharing the desired spatio-temporal data records with specific temporal and spatial ranges from massive data is challenging and requires long accumulation periods.

With so many vehicles, as illustrated in Table 1, even in the field of autopilot with high update frequency, monitoring the real-time operation status is necessary and essential [6], [9]. Buses or taxis are among the most common vehicles in an urban city. To monitor the running state, the installed sensors contain positioning devices, speed measuring instruments and other devices [10]. Thus, the data scale of sensor data generated per day is great, reaching a scale of one hundred million, so we define it as hyper massive spatiotemporal trajectory data (HMSTD) in the study. Therefore, the data observed with the vehicles are hyper massive, which hampers effective and efficient data sharing. In the real-time or near-real-time traffic scenarios, the time consumption of data retrieval makes a difference for urban traffic flow monitoring, traffic congestion alleviation, pedestrian flow detection, and other applications. Efficient data structures and retrieval algorithms are urgent for vast vehicle trajectory data [11], [12].

In relational database patterns, massive data management is a bottleneck that must be addressed at all times [13]–[15]. What counts is to design the spatiotemporal index in vehicle trajectory data retrieval, which can promote data retrieval efficiency sustainably. A spatial index can greatly promote Earth Observation metadata query performance by integrating spatial relationships among different features [16]. The R tree [17] solves the problems of high-dimensional-space search well, which is extended from the B tree in spatial indexing. Optimizations for the R tree have been accomplished and lead to R tree variants, such as the R* tree [18], R+ tree [19], Hilbert R tree [20], and APR-tree [21], distributed R tree [22]. The R tree index and its family have been widely used in spatial database design, such as Oracle Spatial [23], PostgreSQL [24], and others. A trip-oriented data indexing mechanism has been able to achieve efficient massive vehicle trajectory data retrieval [25], which exhibits more excellent query efficiency than Brakatsoulas and Leonardi's two trajectory-segment-based indexing methods [26]. A dynamic queue of Small Files algorithm has been proposed to solve analysis and processing difficulty of numerous small files [27]. A distributed caching method has been proposed to enhance the performance of small-file access in the Hadoop distributed file system [28], which significantly outperforms existing methods in massive small-file-access temporal costs. An approach to dynamically replicate stored data based on the predictive analysis has been proposed [29], which enhances the availability in comparison with the default scheme. A medical image file accessing system has been developed based on HDFS, which can enhance medical imaging storage capability, transmission stability, and transmission reliability while providing a convenient management interface [30].

NoSQL data sharing structures have been proposed to perform larger-scale data sharing in a cost-effective manner instead of relational database patterns, which is one of the most popular and novel databases designed for high-efficiency management of massive unstructured or non-unstructured data [31]. HBase, as one of the most common NoSQL database solutions, can be qualified with high rates of record insertion for location updates with millions of devices, while supporting efficient real-time latest location analysis [32]. The other widely used NoSQL databases include Apache Cassandra [33], Google BigTable [34], SimpleDB [35] and so on. An effective method is proposed to achieve the integration of fine-grained access based on purpose control into a MongoDB cluster, providing support for privacy policy enforcement in MongoDB [36]. Typically, NoSQL techniques sacrifice some functions, such as database-wide transaction consistency, to promote others, such as availability and scalability [37]. At the same time, cloud computing has achieved rapid development that meets immense storage and processing requirements [38], which possesses high availability, high performance, and elastic scalability [39]. Chang *et al.* have demonstrated both performance and scalability advantages of cloud databases compared with HBase for queries and aggregation operations [40]. In [41], an efficient spatiotemporal data retrieval schema is proposed based on Apache Impala to promote vast spatiotemporal data sharing at million data scale. Storing large-scale data reaching 100 billion is feasible, such as with MongoDB and HBase, while retrieving the data with a specified spatiotemporal range in vast data efficiently requires more efforts based on existing techniques. With the growing data size reaching PB data scale, existing solutions can be prohibitively inadaptable and may suffer degradation of performance in hyper massive spatiotemporal data.

Consequently, we attempt to propose a path-divided HDFS data blocking (PDDB) model based on Apache Impala (PDDB-Impala) method to optimize the efficient access manner of HMSTD in this study, which has unique advantages in the very-large-data data retrieval field [42]. Apache Impala, concentrating on t massive data processing on parallel, is ten times faster than Hive for conventional issues [43]. The approach is widely applied into a large bus network in the city of Shenzhen in Guangdong Province, China, and performance evaluation and comparison with existing methods are introduced in the end. Our contributions can be summarized as following:

(1) PDDB-Impala is developed to overcome the challenge of efficient managing and indexing of HMSTD in a Smart City. Different from most existing data blocking schemas, PDDB-Impala is based on path-divided indexing and optimizing the data partition schema to achieve better performance.

(2) PDDB-Impala is especially suitable to long-term and hyper vast spatiotemporal trajectory data, as proved in the Shenzhen bus network, which exhibits 110 times higher performance than MongoDB and HBase at a data scale of 100 billion.

The organization of this study is described as following. Section 2 gives a detailed description of PDDB-Impala. The feasibility and performance of PDDB-Impala are validated in Section 3 by a group of experiments. Moreover, the performance of PDDB-Impala is tested and shows advantages compared with other methods. Section 4 concludes the study and explores future work.

IV. METHODOLOGY

A. PDDB-IMPLA DATA PARQUET PARTITIONING METHOD

The data partition mechanism divides the data in HDFS into multiple data blocks to accelerate the data access speed. Because the volume of spatiotemporal data is massive, as described in Section 1, dividing the data in HDFS is imperative. The data blocking method is a type of data indexing technology that labels data blocks to achieve fast data retrieval. Unlike a data uniform partitioning method, the path-divided HDFS data blocking method creates non-uniform partitions to achieve data division. HMSTD are mainly distributed in road areas, and there is no track data distribution between roads, so there is no trajectory data distribution between roads. Thus, we propose the PDDB method in our study. In PDDB, the data blocking mechanism divides the data in HDFS into multiple data blocks through the trajectory data division and roads.

The PDDB method is illustrated in Fig. 1. In Fig. 1, the HMSTD are distributed on actual crossroads, such as other straight roads, left turn, right turn, three forks and so on. The abstraction of crossroads is the four grey lines, which refer to roads. The red block refers to a data partition block covering the trajectory data locating on roads. In HDFS, the trajectory data are stored in partition data blocks to organize the structure in HDFS. Additionally, the entire width of a data block is m, and the entire length is n. The trajectory data area is covered by data blocks, and non-trajectory data are not partitioned. In this idea, we propose PDDB in our study and describe the computational formulas when computing and executing PDDB. The partition mechanism of block division

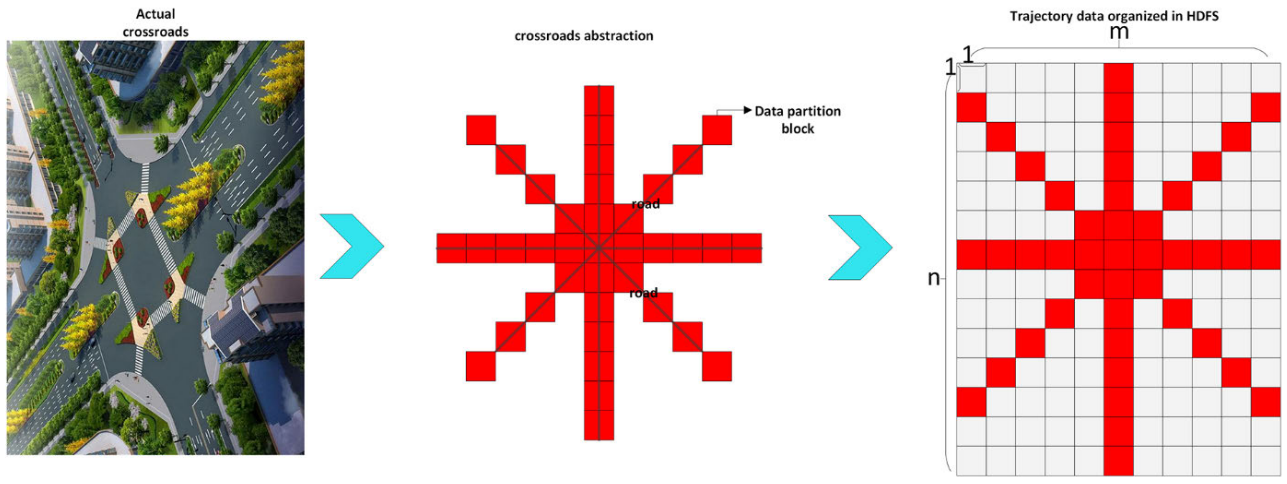


FIGURE 1. The PDDB trajectory data organization method. The transverse distance is m, and longitudinal distance is n.

is designed to be a longitudinal grid and latitudinal grid. The number of partitioned data blocks in HDFS is determined by the total Random-Access Memory (RAM) of the computation cluster and the standard data block size in Apache Impala. The size of a partitioned data block is identified by the entire latitude and longitude of the entire road network and the number of partitioned data blocks in HDFS.

$$N_{Parquet} = \frac{RAM_s \times N}{2 \times RAM_0} \tag{1}$$

In formula (1), $N_{Parquet}$ refers to data blocks number; RAM_s refers to RAM of single node; N refers to the number of whole nodes. RAM_0 refers to the standard data size of a single data block, which is 20 Mbyte in general in our experience.

$$Degree_{partition} = \sqrt{\frac{\sum_{i=1}^{n_n}(longitude_{i+1} - longitude_i) \times (latitude_{i+1} - latitude_i)}{N_{Block}}} \tag{2}$$

In formula (2), n refers to the number of road segments, $Degree_{mean}$ refers to the degree of a partitioned data block; $longitude_{i+1}$ refers to the longitude of road i +1; $latitude_{i+1}$ refers to the latitude of road i +1. Thus, N_{Block} and $Degree_{partition}$ can be calculated via formula (1) and formula (2). The total of the entire computation environment is $RAM_s \times N$. The standard data size of a single data block should be neither too high nor too low to balance the data access performance and size of required RAM in Impala. In our study, the standard data size is flexible, and 20 MB is suitable. As the spatial metadata are carried with massive trajectory data, the degree of partitioned data block is acquired via the accumulation of latitude and longitude of the road network.

Apache Impala supports the data storage in Hadoop Distributed File System (HDFS). Thus, data are stored in HDFS, which is a highly fault-tolerant system that provides high throughput data access. However, to achieve data management in HDFS, HBase is employed to reduce the data management burden. As a distributed and open-source database, HBase has been integrated with Impala implementation in an Apache Impala prototype. In HBase, structured query language has been used to obtain the requisite data from HDFS. It is noticeable that Apache Impala is applied especially in real-time data analysis, such as commercial computing systems and real-time Geographic Information Systems. The Hilbert curve encoding is a classic spatial filling curve coding mechanism to code the adjacent geographical discrete element [44]. A Hilbert curve can linearly penetrate every discrete element in two or higher dimensions according to the characteristics of the filling curve in its own space. Moreover, it passes through only once and performs linear sorting and coding for each discrete element, maintaining the topological properties of the data. In our study, we employ the geohash to achieve geo encoding of the Hilbert coding units.

Fig. 2 shows the Hilbert encoding process and quadtree indexing method of the parquet data blocks. Parquet, a determinant storage format for analytical business that is developed by Twitter and Cloudera and graduated from Apache's incubator in May 2015, is a column-oriented binary file format designed to achieve efficient high-concurrency queries. The parquet files are stored in memory, which speeds up the parquet table reading and writing dramatically. This characteristic accelerates the query efficiency in Impala. Firstly, data partition blocks are traversed through the Hilbert encoding pattern. Secondly, the hierarchical quadtree-based method achieves the parquet data block indexing with the quadtree numbers. Thirdly, the quadtree numbers are inserted into HBase for access.

To support PDDB data management, parquet data blocks in HDFS are stored with HBase. Every parquet data block has its unique latitude and longitude range, and they do not intersect each other. The hierarchical quadtree-based

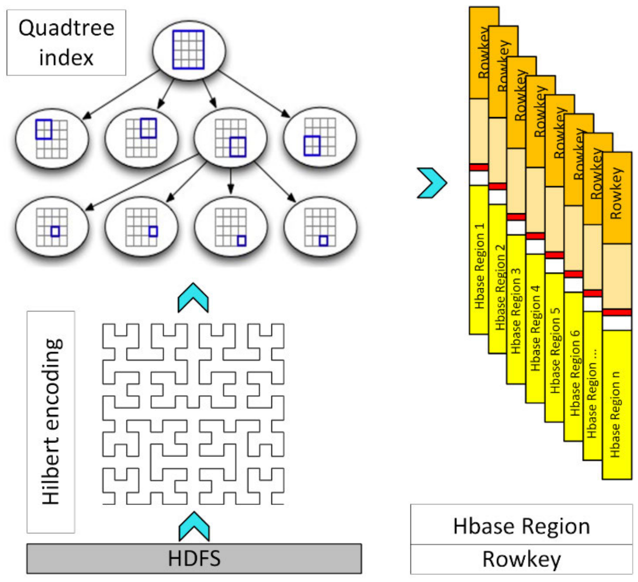


FIGURE 2. **The parquet data block indexing in HBase. The Hilbert encoding pattern and hierarchical quadtree-based method are employed to organize parquet data blocks.**

indexing mechanism helps to find the parquet data intersecting with query space. However, the parquet data blocks are located on the road network, so those parquet data block code values on PDDB are uneven, and there are many empty nodes in the quadtree index structure. The query method is also relatively simple. For example, to retrieve the spatial elements within and intersecting the edges of a polygon, it is only necessary to retrieve all of the spatial elements in leaf nodes covered by the query polygon and its father and ancestor nodes and then perform the necessary spatial operations to retrieve the required spatial elements from them.

B. PDDB-IMPALA-ENABLED DATA ACCESS MECHANISM IN HDFS

In terms of PDDB-Impala-enabled data access, this section introduces the spatiotemporal trajectory data access capability of PDDB-Impala. Impala supports the customized spatiotemporal query request. In an SOS data access request, the spatiotemporal query range is specified, which is the filter rules of PDDB-Impala data. In the PDDB-Impala method, the parquet data block is located on the road network. Consequently, the desired parquet data block can be obtained through the spatial query information and road networks. The outputs are the desired trajectory data collection. In Algorithm 1, the algorithm for vast spatiotemporal trajectory data retrieval based on PDDB is described.

Through the data access extension interface embedded in Apache Impala, HMSTD can be accessed via the data query extending function and return the desired data. In algorithm 1, *spatialQueryRoads*($P_{longitudeRange}$, $P_{latitudeRange}$) supports the array$_{road}$ nodes query. The specified parquet data blocks of array$_{roads}$ can be

Algorithm 1 Massive Spatiotemporal Trajectory Data Retrieve Based on PDDB

```
**Input**: Spatiotemporal range of data query $P_{ST}(P_{longitudeRange}, P_{latitudeRange}, P_{begin}, P_{end})$
**Output**: data retrieve results *trajectoryCollection*
**Use**: *SensorObservationService(ST_tn, Algorithm_PDDB, ResponseFormat)* inherits data access interface of *SOS* implementation
*spatialQueryRoads*($P_{longitudeR}$, $P_{latitudeR}$) *return the bus routes that intersect with or locate in query spatial range*
    *spatialQueryPartition*($P_{longitudeR}$, $P_{latitudeR}$) return the data partition blocks meeting the spatial range
    *temporalQueryPartition* ($P_{begin}$, $P_{end}$) return the data partition blocks meeting the temporal range
    *spatialPartitionInOrNot*($P_{longitude}$, $P_{latitude}$, $Point_{trajectory}$) determine whether trajectory is in specified spatial range
    *temporalPartitionInOrNot*($P_{begin}$, $P_{end}$, $Point_{trajectory}$) determine whether trajectory is in specified temporal range
    **STEP 1:** Extend the mandatory functions in *SOS* containing the *SensorObservationService*($ST_{tn}$, *Algorithm*$_{PDDB}$, *RF*) function in extended PDDB-Impala implementation.
    **STEP 2:** Capture the parameters $P_{ST}(P_{longitudeR}, P_{latitudeR}, P_{begin}, P_{end})$ from $ST_{tn}$ through the *get4*($ST_{tn}$), which is encoded with *Observation& Measurement* encoding model. The spatiotemporal metadata are used to make up a spatiotemporal query statement, which is used to acquire the specified data from the HBase cluster.
    **STEP 3:** initialize the function *spatialQueryRoads*($P_{longitudeR}$, $P_{latitudeR}$) to acquire the *array*$_{roads}$ array, locating on the spatial query range. The function judges whether the spatial location of road nodes is in the specified spatial range. The parquet data blocks of the *array*$_{roads}$ can be retrieved via the all data blocks.
    **STEP 4:** initialize the function *spatial QueryPartition*($P_{longitudeR}$, $P_{latitudeR}$) to acquire the specified *block*$_{tn\_spatial}$ meeting the spatial range. Through the function, the data block partition of *partition*$_{spatial}$ in the HBase are obtained. The data block partitions whose temporal range located in the acquired ($P_{begin}$, $P_{end}$) can be obtained from step 5 in detail.
    **STEP 5:** initialize the function *temporal QueryPartition*($P_{begin}$, $P_{end}$) to acquire the specified *block*$_{tn\_temporal}$ meeting the temporal range. Through the function, the data block partitions *partition*$_{temporal}$ in the HBase are obtained. The data located in the acquired *block*$_{tn\_spatial}$ can be obtained from step 6 in detail.
    **STEP 6:** initialize the function *spatial Point*($P_{longitude}$, $P_{latitude}$, $Point_{trajectory}$) to determine whether $Point_{trajectory}$ is in the spatial range between $P_{longitude}$ and $P_{latitude}$ or not. Through intercepting all the
```

Algorithm 1 *(Continued.)* Massive Spatiotemporal Trajectory Data Retrieve Based on PDDB

```
trajectory data in $partition_{spatial}$, the trajectory data $trajectory_{spatial}$ meeting the spatial range are acquired. The function judges whether the point in $partition_{spatial}$ is in the specified spatial range.
    STEP 7: initialize the function t $emporalPoint(P_{begin}, P_{end}, Point_{trajectory})$ to determine whether $Point_{trajectory}$ is in the temporal range between $P_{begin}$ and $P_{end}$. Through intercepting all the points in $partition_{temporal}$, the trajectory data $trajectory_{temporal}$ meeting the temporal range are discovered. Integrating the $trajectory_{spatial}$ and $trajectory_{temporal}$ to compose the $trajectoryCollection$ and returning it back to the application or users encoded in the response document in SOS.
```

retrieved via all data blocks. Moreover, $SpatialPointInOrNot(P_{longitude}, P_{latitude}, Point_{tn\_m})$ supports the spatial trajectory data query, while $TemporalQueryBlock(P_{begin}, P_{end})$ supports the temporal trajectory data query. Via the three judgements, the spatiotemporal trajectory data meeting the query criteria can be retrieved.

C. APPROACH FOR INTEGRATING HBASE AND SOS IN HDFS

In PPDB, the vast spatiotemporal trajectory data and metadata are stored in HDFS. To integrate HBase with SOS, there are some improvements that should be made. Firstly, the SOS is a flexible web service that couples in-situ sensors observation with remote sensing observations for the data extension interface. To support HMSTD storage and access, the connection between SOS and HDFS should be achieved and stable. Thus, the data interface of data access in SOS should be implemented. Secondly, the metadata of sensors and observation data, namely, observation metadata, platform metadata, data product metadata, and trajectory data, should be stored at the same time for effective use in follow-up applications. Thirdly, the technical cross-platform difficulty should be addressed. Apache Impala supports a C++ programming interface for spatial computation functions. Moreover, OpenSceneGraph is a third-party library that can create interactive graphics programs with high performance and cross-platform compatibility more quickly and conveniently. Consequently, OpenSceneGraph library can be put into use, and the self-defined spatiotemporal query function based on PDDB is developed with the help of OpenSceneGraph library.

OGC provides the data encoding schema of the Observation& Measurements [45], [46], encoding the sensor observations and metadata. Relatively, the tables in HBase contain the observation sensor, observation task, observed objects, observation spatial range, and observed value. Fig. 4(a) describes the data type mapping from the PostgreSQL (version 9.3) database to HBase (version 1.4.10). The arrows in Fig. 3(a) shows the data type conversion between different data storages. In HBase, those tables can then be retrieved with Rowkey or Scan query manner. In addition, Apache Impala supports Structured Query Language analytical patterns. The Geometry is converted to GeoJson, which can record the geometry and feature in a JavaScript object notation encoding file. The other data types are converted into Char type. The data and metadata are stored as binary files in HDFS. Fig. 3(b) describes the tables in HBase for the storage of vast spatiotemporal trajectory data and metadata. The entire tables are comprised of Regions, which contain multiple data tables, such as SensorType, Sensor, observation and so on. The tables in HBase are described mainly as follows:

(1) sensorType: storage of sensor type and phenomenon.

(2) sensor: storage of sensor information containing the sensor metadata, phenomenon, and the observation time range.

(3) observation: storage of observation information when the sensor finishes observation task, which contains the observation time, the observation result, and the spatial range of observation data types.

(4) phenomenon: storage of phenomenon information containing the phenomenon name, observation value type, and observation value unit information so on.

(5) offering: storage of organization information containing the organization name, observation phenomenon name and involved sensor information.

(6) featureOfInterest: storage of observation spatial arrangement containing the observation sites name and coordinate encoded with GeoJson.

V. EXPERIMENTS AND RESULT DISCUSSIONS

A. PDDB-IMPALA ACCESS FRAMEWORK

Widely applied in business companies, such as Twitter and Facebook, Apache Impala is the open-source query and analysis engine for Apache Hadoop [47], reaching PB data scale. As stated by Apache Foundation, Impala can achieve an efficiency that is 3 to 90 times higher than that of Hive [48]. As Fig. 1 describes, the PDDB-Impala access framework contains three parts: The Sensor Observation Service (SOS) web service, Apache Impala, and trajectory data.

The SOS web service provides the query spatiotemporal data insert and access request. Published by Open Geospatial Consortium (OGC), the SOS [49] provides data access based on pull and standardized data access to observations and metadata [50]. SOS provides the sensor observation communication, such as observation insertion and acquisition. SOS provides the bridge between the observation and data storage centre and facilitates the users [51]. In Sensor Web Enablement [52], SOS plays an essential role in in-situ observation communication as a web service. However, in emerging technological areas, such as the IoT and 5G fields, SOS plays the role of sensor observation data sharing in massive, ubiquitous, heterogeneous sensors. Additionally, the SOS provides solid data support for smart city applications, such as smart transportation, smart government, smart security, smart education,

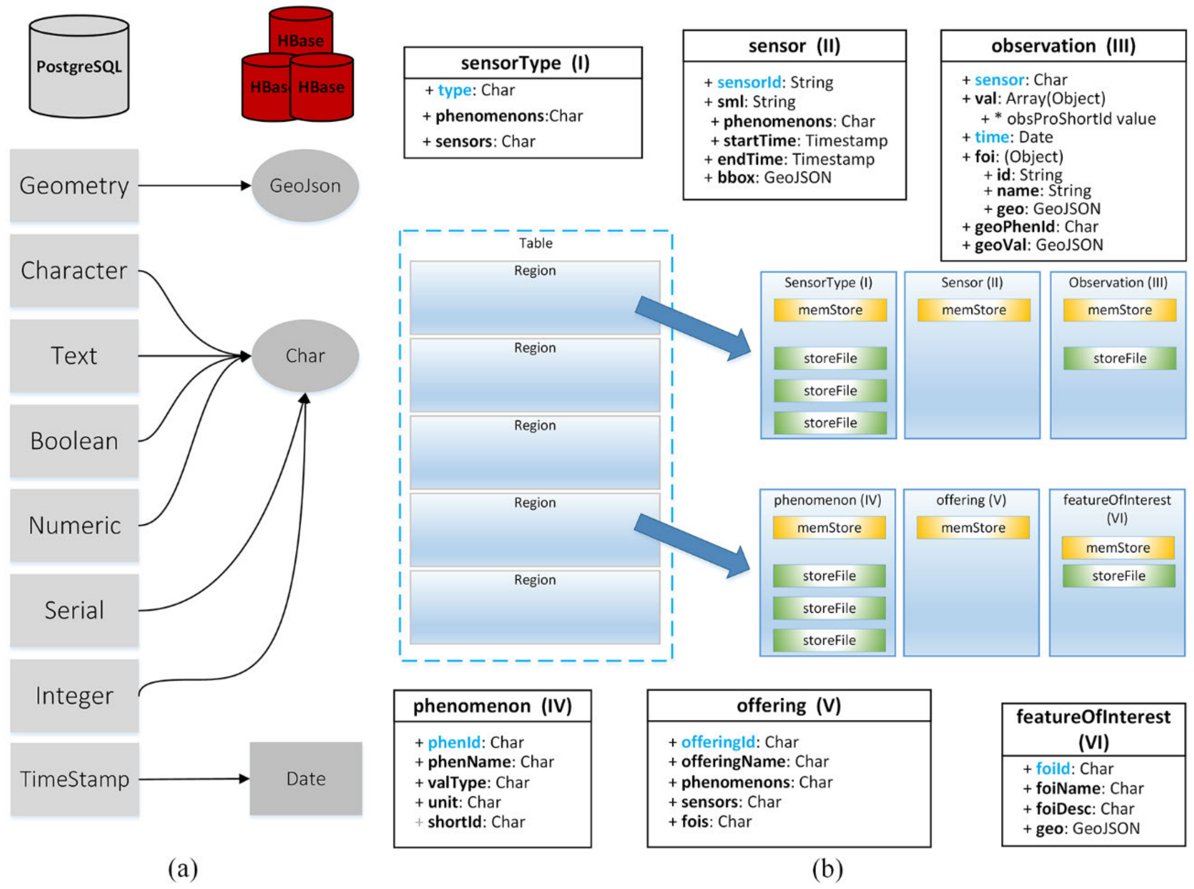


FIGURE 3. **(a) The data type mapping from the PostgreSQL (version 9.3) database to HBase (version 1.4.10). (b) Tables in HBase storing vast spatiotemporal trajectory data and metadata.**

and smart medical treatment, which form the foundation and connotation of a smart city.

In Apache Impala, PDDB-Impala is proposed. By extending the parquet data blocking schema, PDDB-Impala achieves the function of data insertion into designated path parquet data blocks located on the road network. In the data query process, the desired trajectory data are obtained from the parquet data blocks located on the road network. The query planner is the core part of Apache Impala, which achieves the distributed query of HMSTD in memory. Moreover, the metadata of an Apache Impala cluster is stored in state storage nodes to achieve state management.

In an HBase data storage centre, the distributed database supports massive trajectory data storage for distributed storage capability. HBase [53] is distributed, scalable, and column-oriented big data storage. Different from general relational databases, HBase is suitable for unstructured data storage [54]. In an HBase storage cluster, the metadata of buses, taxis, and other means of transport vehicles are stored, as illustrated in Fig. 4. Moreover, the HMSTD and the spatio-temporal indexing data are stored in a distributed manner to provide effective and efficient data access to HMSTD.

B. BD TRAJECTORY DATA AND EXPERIMENTAL ENVIRONMENT

In terms of the experiment data, the Shenzhen BD bus network is chosen as the experimental area in our experiments, containing 35,809 BD buses. In the Shenzhen BD bus network, every 30 seconds, observed data are recorded consisting of bus basic information, such as bus location, speed, and observation time. In addition, the number of data records is about 1.03 billion each day, as shown in Table 1. In the south of China, Shenzhen is in the south of Shenzhen River and Hong Kong, which is the link and bridge between Hong Kong and the Chinese mainland.

Fig. 5 displays the spatial location of Shenzhen bus stops, bus routes and administrative boundary, where red dots stand for bus stops and black lines refer to bus lines in Shenzhen city, containing ten districts, about 4461 bus stations, 541 bus lines in the city, and about 35,809 running BD buses. Widely used in road traffic, shipping, vehicle monitoring, and vehicle navigation fields, the ATGM332D BD positioning module installed on buses is an easy-to-use positioner with small volume, low power consumption, and high precision. Following American Global Positioning System and Russian Global Navigation Satellite System position system,

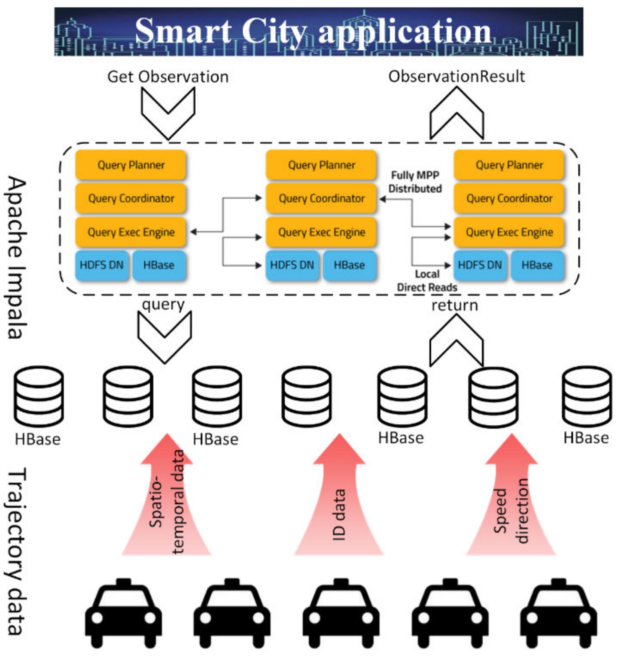


FIGURE 4. The PDDB-Impala access framework to access HMSTD. The SOS web service, Apache Impala, and trajectory data are involved.

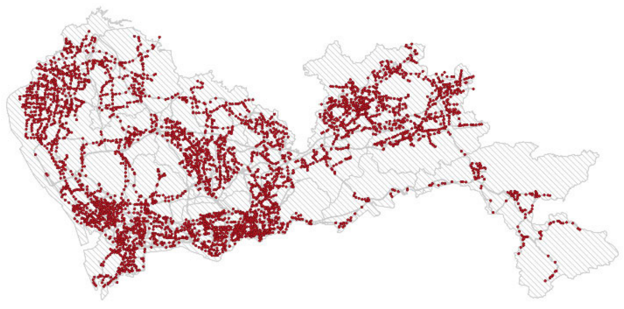


FIGURE 5. Spatial distributions of Shenzhen bus stops and bus routes. The grey outline is the administrative boundary. Nanshan district, Guangming district is shown specially.

the Chinese BD satellite navigation system is one of the most popular and sophisticated satellite navigation systems in the world. Besides, the spatial positioning accuracy of the Chinese BD satellite navigation system is generally equal to that of GPS [55].

In terms of the experimental environment, the hardware environment is completely distributed, containing six Datanodes and one Namenode. The Apache Impala implementation is deployed on the seven nodes with the same configuration, while operating systems (CentOS 7.0) and HBase are deployed on each node. The configuration of each node is an Intel i5 8300H processor and 64 GigaByte RAM, all of which are connected to 100 GB/s InfiniBand.

In comparison, the 52° North SOS [56] is applied in our experiments to make the contrasting experiment test. In extension, the 52° North SOS implementation is made extension to support data access to HDFS, which is based on a Data Access Object logic interface, containing GetCapabilityDAO, GetObservationDAO, and DescribeSensorDAO. Moreover, GetObservationDAO can be utilized to extend the capability of observation access to HDFS. Coupling with the HDFS cluster closely, Impala can be accessed with an HDFS web address to understand the Impala cluster situation.

To validate data access performance in general and mature solutions at the moment, such as MongoDB [57] and HBase, we utilized the different data scales of HMSTD to test the feasibility and advantages of the PDDB-Impala method compared with other methods, such as the equipartition in the Impala method, MongoDB method, and HBase method. In the different cases, we can perform a clear and detailed data access test and verification.

C. DATA RETRIEVAL AT DIFFERENT TIMES

In terms of data blocking mechanism to increase data access efficiency, more parquet data blocks work. However, too many data blocks will occupy too much memory, which is potentially out of the memory of the computation cluster. Consequently, the size of the parquet data block must be set up. As described in Section 2.2, the BD spatiotemporal trajectory data partitioning method is described. Moreover, to avoid memory overflow, the memory occupation of the resource consumption mechanism of Impala should not exceed cluster memory. Hence, the number of parquet data blocks should be calculated. The number of block partitions is 22937, as described in formula (1). The degree range of parquet data blocks degree range is 0.0037° according to formula (2). Consequently, the used memory of an Impala cluster remains under the cluster memory limit. As shown in Fig. 6, parquet data block distribution in HDFS in the PDDB method is described. Parquet data blocks, which evenly scatter on bus lines, contain data sets satisfying this spatial range, disjoint with each other. The distribution density of a Parquet data block in the Nanshan district, Guangming district, and Longhua district is the same, as illustrated in Fig. 6, storing the trajectory data in HDFS for access.

For example, firstly, a parquet data block is encoded with the Hilbert curve as described in Section 2.2, and the parquets are retrieved with the specified spatiotemporal range based on Hilbert encoding mechanism. By decoding the specific Hilbert encoding value, the rank and row are calculated. Secondly, Apache Impala support rapid spatiotemporal data retrieval, based on its data partition mechanism and memory mechanism. Thus, the Parquet data blocks in HDFS are retrieved in a short time. Thirdly, encapsulated in the response document of the SOS web service, the vast required spatiotemporal trajectory data in parquet is returned to the users or application quickly.

In terms of the data retrieval part, the aim is to illustrate the spatiotemporal data visualization of retrieval in different spatial ranges and different times. In this part, we made the HMSTD access based on PDDB-Impala. In fact, the input is embedded in a GetObservation request in SOS, encoded with the schema of SOS. The output is the BD bus trajectory and metadata that meets the spatiotemporal range and Identification, which are BD bus observation

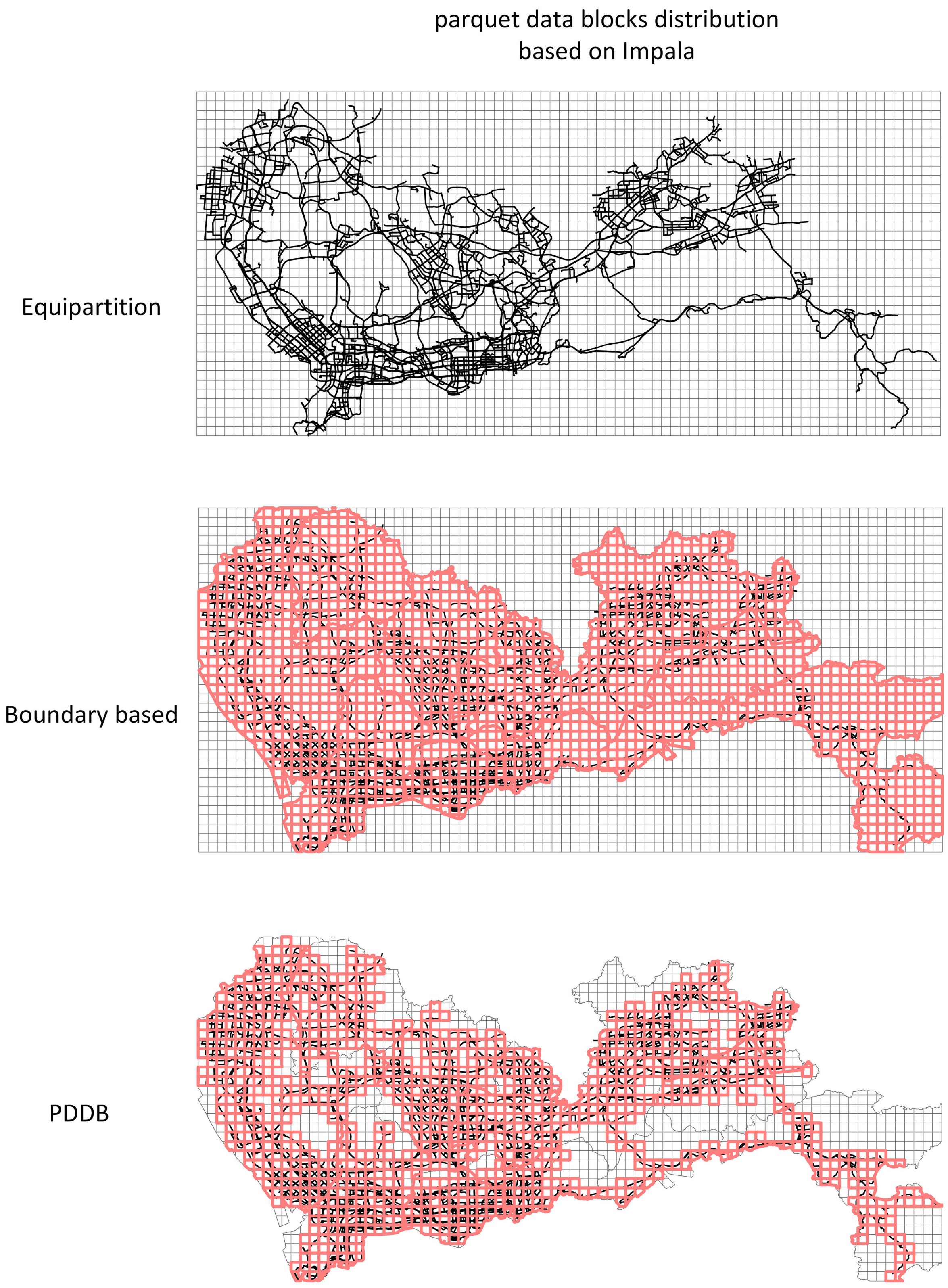


FIGURE 6. **Parquet data block distribution in HDFS through the equipartition, boundary based, and PDDB-Impala methods. Red squares stand for parquet data blocks, which evenly scatter on bus lines.**

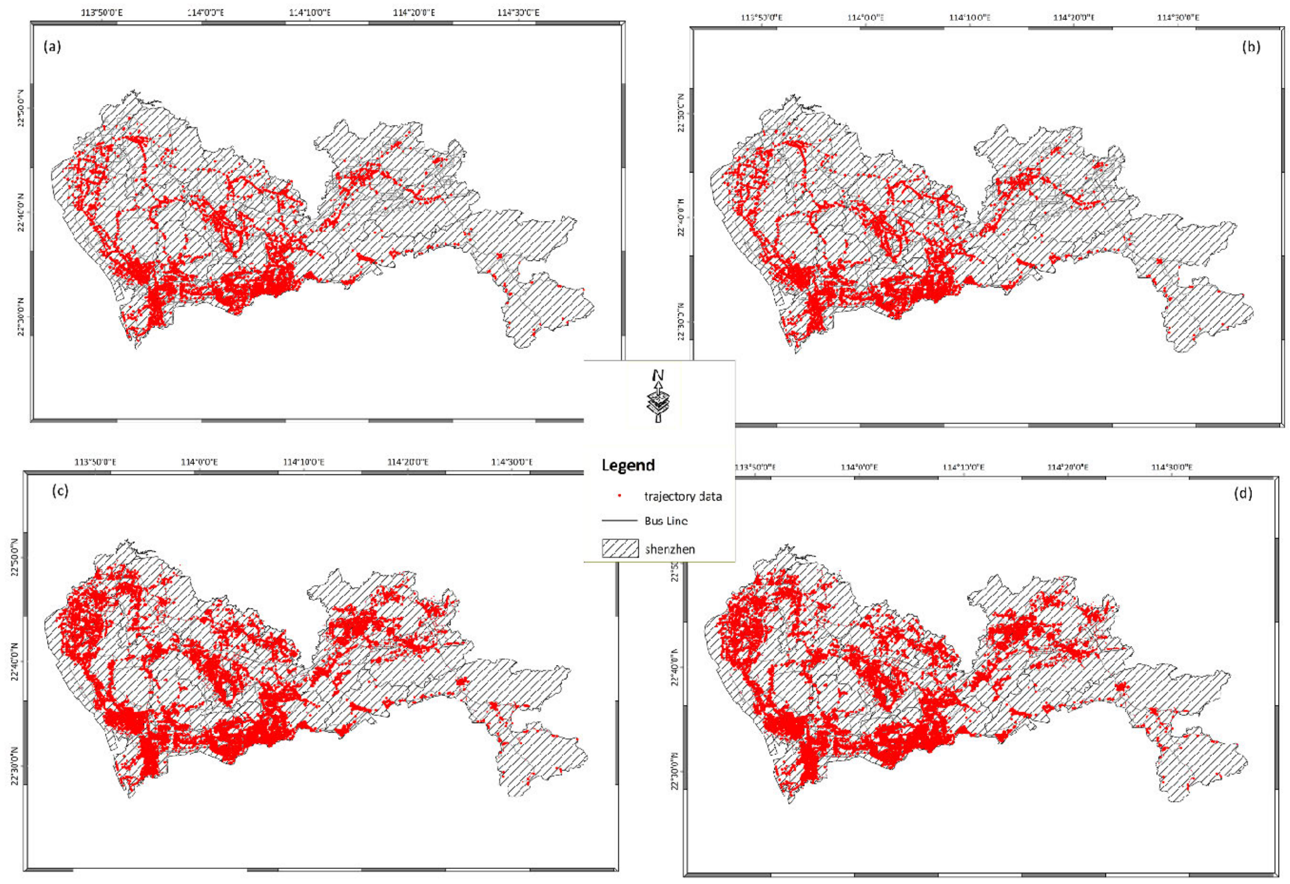


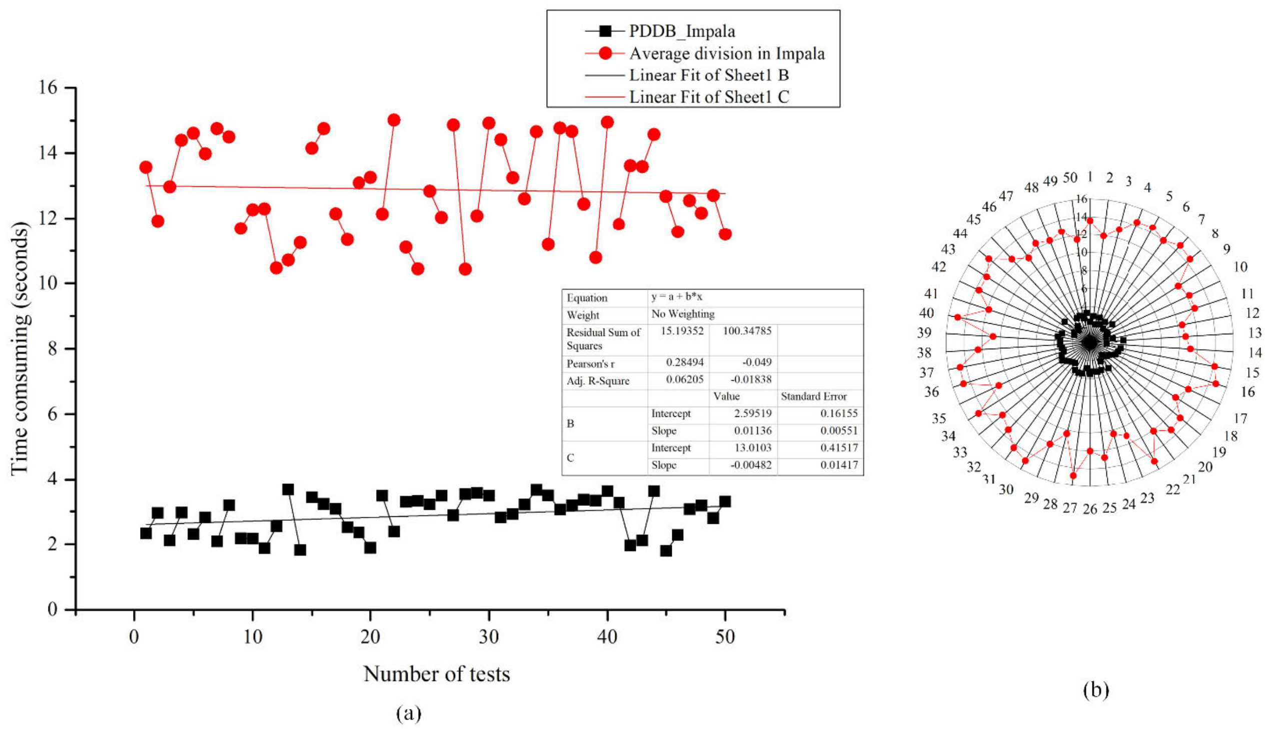
FIGURE 7. **Retrieved BD buses located in Shenzhen city at different times: (a) 7:00 a.m. to 9:00 a.m., (b) 11:00 a.m. to 01:00 p.m., (c) 4:00 p.m. to 6:00 p.m., (d) 8:00 p.m. to 10:00 p.m.**

data containing the bus name, running speed, spatial location and other information. As shown in Fig. 7, most bus trajectory data are located on the main streets, such as Shennan Road, Binghai Road, Baoan Road, Huanggang Road, and Xiangmihu Road. The desired trajectory data are retrieved from the HDFS via Apache Impala implementation. After sending a retrieve request to the PDDB-Impala server, the specified data in the HBase are retrieved and spread on the Map of Shenzhen. Taking the "粤 CJ202" bus as an example, the bus location, bus passenger status, running speed and other information are retrieved from the HBase ("longitude": "114.063", "latitude": "22.528", "passengerStatus": "1","busID": "100", "speed": "0","busNumber":"103"). In our experiments, due to the parquet data block partitioning mechanism, most buses are located on or near urban roads, leading to a dense state on urban roads, while buses in residential business zones, communities, and other non-road areas are sparse or empty, as Fig. 7 describes.

Fig. 7 shows the bus locations in Shenzhen city at different times, such as (a) 7:00 a.m. to 9:00 a.m. and (b) 11:00 a.m. to 01:00 p.m. Red dots represent the bus stations, and grey lines represent the main roads in Shenzhen city. At different times, the red pots density is different at same areas according to Fig. 7. As shown in Fig. 7, about 79,462 buses are retrieved from 7:00 a.m. to 9:00. About 48,716 buses are retrieved from 11:00 a.m. to 01:00 p.m. The bus density of Fig. 7(b) is obviously larger than that of Fig. 7(a), such as in the Baoan central district, shajing area, and futian central area. Different time periods lead to different densities of people flow, so bus density varies with time, which is greatest in traffic peaks in one day.

D. EXPERIMENTAL PERFORMANCE EVALUATION

To test the PDDB-Impala efficiency, experimental performance evaluation and comparison between the PDDB-Impala and existing mechanisms or methods are presented in this section. In data retrieval input, the spatial retrieval range is the bounding box of (114.321 22.832 114. 321 22.371 114. 914 22. 371 114. 914 22.837 114.321 22.832), and the temporal range is "2019-07-03 12:00:00" to "2019-07-03 13:30:00"; the bus ID is "urn:szu:mobilesensor:shenzhenbus: 粤 B71692." In the experiments, the vast trajectory data whose data scales vary from 1 billion to 100 billion are tested. Through the data block partition method described in Section 2, the entire spatiotemporal trajectory data are divided into 22,937 data



Equation	y = a + b*x		
Weight	No Weighting		
Residual Sum of Squares	15.19352	100.34785	
Pearson's r	0.28494	-0.049	
Adj. R-Square	0.06205	-0.01838	
		Value	Standard Error
B	Intercept	2.59519	0.16155
	Slope	0.01136	0.00551
C	Intercept	13.0103	0.41517
	Slope	-0.00482	0.01417

FIGURE 8. **(a) Polygonal chart of time consumption in 10 billion data scale between PDDB-Impala and Impala with equipartition in the Impala method. (b) Radar chart of time consumption in 10 billion data scale between PDDB-Impala and Impala with equipartition in the Impala method.**

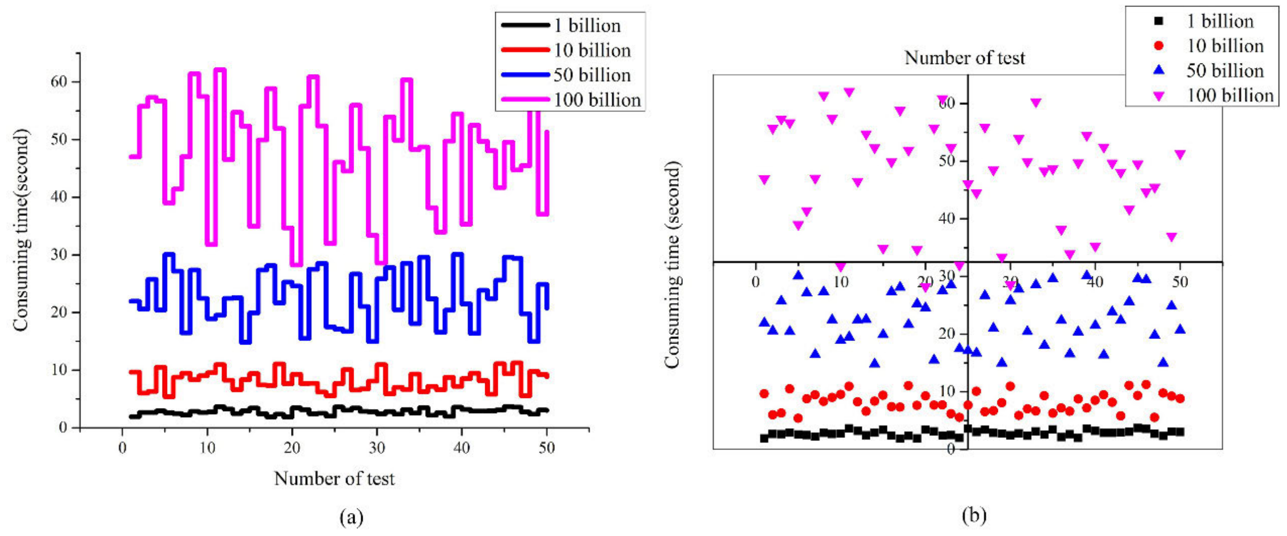


FIGURE 9. **(a) Horizontal step chart of time consumption in 1 billion, 10 billion, 50 billion, and 100 billion data scale in the PDDB-Impala method. (b) Scatter chart of time consumption from 1 billion to 100 billion data scale in the PDDB-Impala method.**

blocks according to spatial area and compressed in the format of a parquet file.

Fig. 8(a) shows the time consumption with PDDB-Impala with the equipartition in the Impala method in fifty hundred billion trajectory data retrievals. When the data scale is 10 billion, the mean consumption time of retrieved records is 6.00 seconds in PDDB-Impala. In equipartition in the Impala method, the mean consumption time is 15.043 seconds. According to Fig. 8, the consumption time of PDDB-Impala is about 2.4 times higher than that of equipartition in the Impala method. Based on the PDDB partition algorithms in Section 2, the efficiency of HMSTD sharing can be promoted greatly, as proved in this experiment. In this phase, it is found that PDDB-Impala outperforms equipartition in the Impala method in the 10 billion data scale, so the performance of the proposed PDDB-Impala is about 2.4 times higher than that of equipartition in the Impala method. Fig. 8(b) shows that the efficiency of HMSTD access in PDDB-Impala and

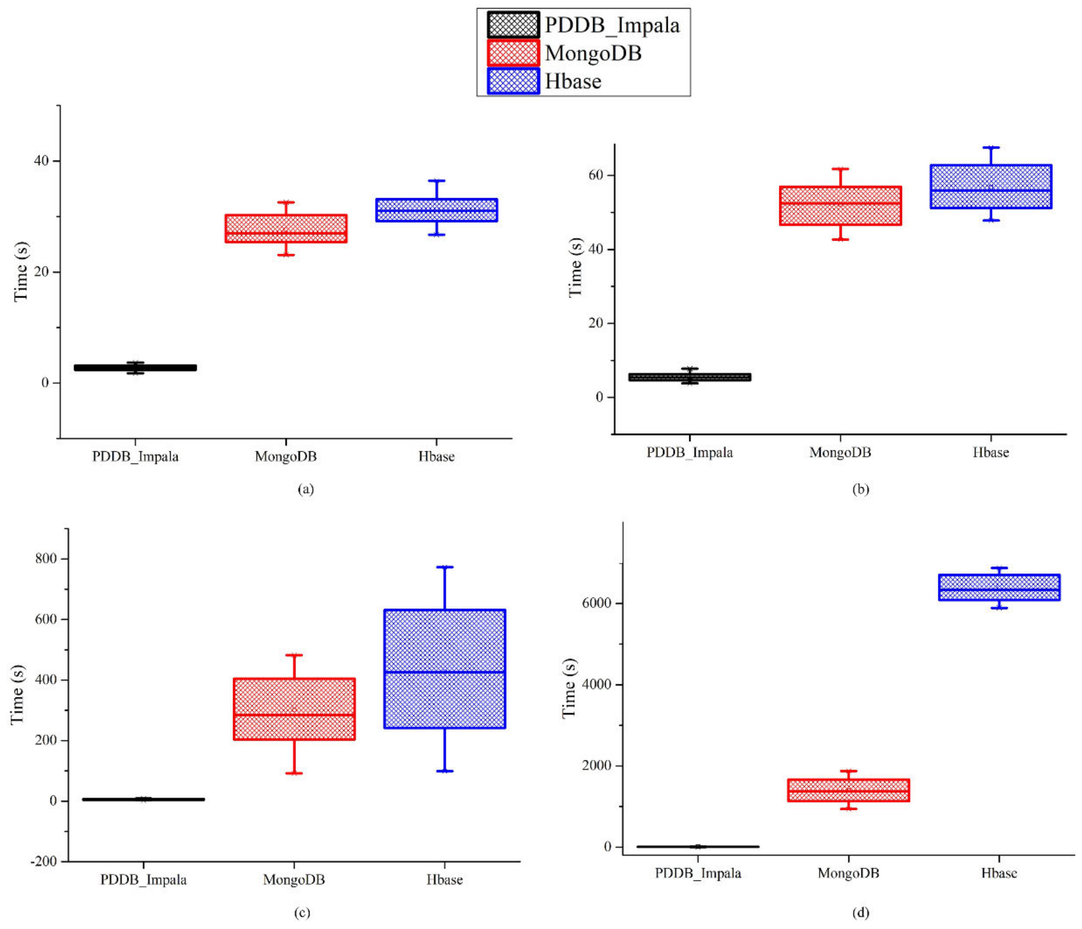


FIGURE 10. Time consumption comparisons between PDDB-Impala (black box), MongoDB (red box), and HBase (blue box) for different scales: (a) 1 billion data, (b) 10 billion data, (c) 50 billion data, and (d) 100 billion data.

equipartition in the Impala method fluctuated slowly, which shows stable performance.

As to different data scales of HMSTD, the experiments based on different data scales should be compared. We choose different data scales of HMSTD in our experiments, from 1 billion to 100 billion trajectory data. Fig. 9(a) and Fig. 9(b) show the time consumption in different data scales with PDDB-Impala. The black column stands for time consumption with PDDB-Impala in 1 billion data scale; the red column stands for time consumption with PDDB-Impala in 10 billion data scale; the blue line stands for time consumption with PDDB-Impala in 50 billion data scale; the pink line stands for time consumption with PDDB-Impala in 100 billion data scale. Overall, the time consumption with PDDB-Impala in 100 billion data scale is higher than that in 1 billion, 10 billion, and 50 billion data scale during 50 tests, as shown in Fig. 9. In Fig. 9(a), the mean time of data retrieval in 1 billion data scale is 2.68 seconds, while it is 7.96 seconds, 21.33 seconds, and 46.13 seconds in 10 billion, 50 billion, and 100 billion data scale. The performance of data access in billion data scale is 2 times higher than that in 10 billion data scale, 6 times higher than that in 50 billion data scale, and 12 times higher than that in 100 billion data scale. From Fig. 9(b), the data access performance in 100 billion data scale presents higher volatility than that in 1 billion, 10 billion, and 50 billion data scale. Therefore, the performance of PDDB-Impala in 1 billion and 50 billion remains stable instead of jumping.

As introduced in the beginning of Section 3.1, the consumption time comparisons between PDDB-Impala, MongoDB and HBase in different data scales should be achieved. Fig. 10(a)-(d) show the consumption time of PDDB-Impala compared with that of MongoDB and HBase from 1 billion to 100 billion. In 1 billion data scale, the mean time consumption of PDDB-Impala is 2.9 seconds, while the time consumptions achieved by MongoDB and HBase are 27.4 seconds and 31.5 seconds; in 10 billion data scale, the time consumption of PDDB-Impala is 5.5 seconds, and the time consumptions of MongoDB and HBase are 51.9 seconds and 58.8 seconds; in 50 billion data scale, the mean time consumption of PDDB-Impala is 8.9 seconds, and the time consumptions of MongoDB and HBase are about 270.8 seconds and 395.5 seconds; in 100 billion data scale, the mean time consumption of PDDB-Impala is 12.4 seconds, and the time consumptions of MongoDB and HBase are 1378.9 seconds and 6312.7 seconds.

Consequently, from 1 billion to 100 billion data scale, the time consumption of PDDB-Impala is about 8 times, 9 times, 29 times, and 110 times higher than that of MongoDB and HBase. The consumption time of HMSTD access suffers from the data scales as revealed in the four sub Figure, which is a key factor influencing efficiency of HMSTD access. Moreover, from 1 billion to 100 billion data size, the consumption time gap for HMSTD access of PDDB-Impala is growing larger than that of MongoDB and HBase, so the efficiency of PDDB-Impala is growing higher and higher compared to that of MongoDB and HBase in HMSTD retrieval. Thus, the larger the data size in HDFS is, the greater the performance gap in HMSTD access is.

In conclusion, firstly, PDDB-Impala achieves better efficiency than Impala with equipartition in the Impala method on one hand. On the other hand, PDDB-Impala can achieve efficient data retrieval at the one hundred million data scale compared to the retrieval efficiencies of MongoDB and HBase. Secondly, in data scales of 1 billion, 10 billion, 50 billion, and 100 billion, the performance of PDDB-Impala is about 8 times, 9 times, 29 times, and 110 times higher than that of MongoDB and HBase. Thus, for HMSTD access, especially at the 100 billion data scale, PDDB-Impala shows unique advantages compared with MongoDB and HBase. Thirdly, the larger the data size in HDFS is, the bigger the performance gap in HMSTD access is. The performance advantage is more obvious when the data scale reaches to 100 billion.

VI. CONCLUSION AND FUTURE WORK

Confronted with the ever-increasing scale of data sets reaching as scales as large as 100 billion records, traditional relational database or NoSQL methods can encounter dramatic performance degradation and may suffer limited scalability in terms of hyper massive spatiotemporal data. In this study, PDDB model is proposed to enhance the HMSTD sharing performance in smart cities. On one hand, the proposed PDDB-Impala can promote the efficiency of massive data retrieve compared with that of Impala with equipartition in Impala, as proved experimentally. On the other hand, compared with high-performance data access techniques such as MongoDB and HBase, PDDB-Impala outperforms the others, as shown experimentally. The performance of PDDB-Impala is about 8 times, 9 times, 29 times, and 110 times higher than those in MongoDB and HBase for data scales from 1 billion to 100 billion. Moreover, the bus distribution in Shenzhen city is collected from 7:00 a.m. to 9:00 a.m. and 11:00 a.m. to 01:00 p.m. Compared with other indexing solutions containing Impala and HBase datastores, PDDB-Impala can achieve the most efficient data retrieval, especially at vast data scales reaching 100 billion records, as tested in the above experiments. The consumption time of PDDB-Impala at the 100 billion data scale is about 12.4 seconds, the efficiency of which is about 110 times higher than that of MongoDB and 508 times higher than that of HBase, which can meet the required efficiency of data access in near-real-time traffic scenarios, such as other heterogeneous and massive vehicle monitoring networks.

To put it into practical smart city application, the PDDB-Impala methodology should take the heterogeneity into consideration, such as GPS, Global Navigation Satellite System, and BD. Moreover, the trajectory platform contains buses, taxis, pedestrians, bicycles and so on. Then, PDDB-Impala can provide stable and ubiquitous services for vehicle monitoring networks and other IoT fields.

REFERENCES

[1] C. Harrison, B. Eckman, R. Hamilton, P. Hartswick, J. Kalagnanam, J. Paraszczak, and P. Williams, "Foundations for smarter cities," *IBM J. Res. Develop.*, vol. 54, no. 4, pp. 1–16, Jul. 2010.

[2] S. Ke, J. Gong, S. Li, Q. Zhu, X. Liu, and Y. Zhang, "A hybrid spatio-temporal data indexing method for trajectory databases," *Sensors*, vol. 14, no. 7, pp. 12990–13005, Jul. 2014.

[3] Y. Jiang and X. Li, "Travel time prediction based on historical trajectory data," *Ann. GIS*, vol. 19, no. 1, pp. 27–35, Mar. 2013.

[4] M.-P. Kwan, "Algorithmic geographies: Big data, algorithmic uncertainty, and the production of geographic knowledge," *Ann. Assoc. Amer. Geogr.*, vol. 106, no. 2, pp. 274–282, 2016.

[5] D. Minoli and B. Occhiogrosso, "Practical aspects for the integration of 5G networks and IoT applications in smart cities environments," *Wireless Commun. Mobile Comput.*, vol. 2019, pp. 1–30, Aug. 2019.

[6] M. Yan, G. Feng, J. Zhou, Y. Sun, and Y.-C. Liang, "Intelligent resource scheduling for 5G radio access network slicing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7691–7703, Aug. 2019.

[7] X. Liu, M. Jia, X. Zhang, and W. Lu, "A novel multichannel Internet of Things based on dynamic spectrum sharing in 5G communication," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 5962–5970, Aug. 2019.

[8] M. A. Brovelli, M. Minghini, and G. Zamboni, "Public participation in GIS via mobile applications," *ISPRS J. Photogramm. Remote Sens.*, vol. 114, pp. 306–315, Apr. 2016.

[9] J. J. Fernández-Lozano, M. Martín-Guzmán, J. Martín-Ávila, and A. García-Cerezo, "A wireless sensor network for urban traffic characterization and trend monitoring," *Sensors*, vol. 15, no. 10, pp. 26143–26169, 2015.

[10] S. Suranthiran and S. Jayasuriya, "Optimal fusion of multiple nonlinear sensor data," *IEEE Sensors J.*, vol. 4, no. 5, pp. 651–663, Oct. 2004.

[11] B. Jiang and X. Yao, "Location-based services and GIS in perspective," *Comput. Environ. Urban Syst.*, vol. 30, no. 6, pp. 712–725, 2006.

[12] 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.

[13] P. Amirian, A. Basiri, and A. Winstanley, "Efficient online sharing of geospatial big data using NoSQL XML databases," in *Proc. 4th Int. Conf. Comput. Geospatial Res. Appl.*, Jul. 2013, p. 152.

[14] C. Heipke, "Crowdsourcing geospatial data," *ISPRS J. Photogramm. Remote Sens.*, vol. 65, no. 6, pp. 550–557, Nov. 2010.

[15] B. Omidvar-Tehrani, P. A. Souza Neto, F. M. F. Pontes, and F. Bento, "GeoGuide: An interactive guidance approach for spatial data," in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Social Comput. (CPSCom), IEEE Smart Data (SmartData)*, Jun. 2017, pp. 1112–1117.

[16] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, 1996, pp. 1–6.

[17] S. T. Leutenegger, M. A. Lopez, and J. Edgington, "STR: A simple and efficient algorithm for R-tree packing," in *Proc. 13th Int. Conf. Data Eng.*, Apr. 1997, pp. 497–506.

[18] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles," *ACM SIGMOD Rec.*, vol. 19, no. 2, pp. 322–331, May 1990.

[19] T. K. Sellis, N. Roussopoulos, and C. Faloutsos, "The R+ -tree: A dynamic index for multi-dimensional objects," in *Proc. 13th Int. Conf. Very Large Data Bases (VLDB)*, Brighton, U.K., Sep. 1987.

[20] R. Yu, "Characterization and sampled-data design of dual-tree filter banks for Hilbert transform pairs of wavelet bases," *IEEE Trans. Signal Process.*, vol. 55, no. 6, pp. 2458–2471, Jun. 2007.

[21] H.-J. Cho, J.-K. Min, and C.-W. Chung, "An adaptive indexing technique using spatio-temporal query workloads," *Inf. Softw. Technol.*, vol. 46, no. 4, pp. 229–241, Mar. 2004.
[22] J. Xia, C. Yang, and Q. Li, "Building a spatiotemporal index for earth observation big data," *Int. J. Appl. Earth Observ. Geoinf.*, vol. 73, pp. 245–252, Dec. 2018.
[23] M. Vazirgiannis, Y. Theodoridis, and T. Sellis, "Spatio-temporal composition and indexing for large multimedia applications," *Multimedia Syst.*, vol. 6, no. 4, pp. 284–298, Jul. 1998.
[24] B. Baas, "NoSQL spatial: Neo4j versus PostGIS," Technische Universiteit Delft, Delft, The Netherlands, May 2012.
[25] T. Xu, X. Zhang, C. Claramunt, and X. Li, "TripCube: A trip-oriented vehicle trajectory data indexing structure," *Comput., Environ. Urban Syst.*, vol. 67, pp. 21–28, Jan. 2018.
[26] L. Leonardi, S. Orlando, A. Raffaetà, A. Roncato, C. Silvestri, G. Andrienko, and N. Andrienko, "A general framework for trajectory data warehousing and visual OLAP," *GeoInformatica*, vol. 18, no. 2, pp. 273–312, Apr. 2014.
[27] W. Jing, D. Tong, G. Chen, C. Zhao, and L. Zhu, "An optimized method of HDFS for massive small files storage," *Comput. Sci. Inf. Syst.*, vol. 15, no. 3, pp. 533–548, 2018.
[28] K. Bok, H. Oh, J. Lim, Y. Pae, H. Choi, B. Lee, and J. Yoo, "An efficient distributed caching for accessing small files in HDFS," *Cluster Comput.*, vol. 20, no. 4, pp. 3579–3592, Dec. 2017.
[29] D.-M. Bui, S. Hussain, E.-N. Huh, and S. Lee, "Adaptive replication management in HDFS based on supervised learning," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1369–1382, Jun. 2016.
[30] C.-T. Yang, W.-C. Shih, L.-T. Chen, C.-T. Kuo, F.-C. Jiang, and F.-Y. Leu, "Accessing medical image file with co-allocation HDFS in cloud," *Future Gener. Comput. Syst.*, vols. 43–44, pp. 61–73, Feb. 2015.
[31] E. Barbierato, M. Gribaudo, and M. Iacono, "Performance evaluation of NoSQL big-data applications using multi-formalism models," *Future Gener. Comput. Syst.*, vol. 37, pp. 345–353, Jul. 2014.
[32] S. Nishimura, S. Das, D. Agrawal, and A. El Abbadi, "$\mathcal{MD}$-HBase: Design and implementation of an elastic data infrastructure for cloud-scale location services," *Distrib. Parallel Databases*, vol. 31, no. 2, pp. 289–319, 2013.
[33] A. Chebotko, A. Kashlev, and S. Lu, "A big data modeling methodology for Apache Cassandra," in *Proc. IEEE Int. Congr. Big Data*, Jun. 2015, pp. 238–245.
[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] E. Sciore, "SimpleDB: A simple java-based multiuser syst for teaching database internals," *ACM SIGCSE Bull.*, vol. 39, no. 1, p. 561, Mar. 2007.
[36] P. Colombo and E. Ferrari, "Enhancing MongoDB with purpose-based access control," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 6, pp. 591–604, Nov. 2017.
[37] R. Cattell, "Scalable SQL and NoSQL data stores," *ACM SIGMOD Rec.*, vol. 39, no. 4, pp. 12–27, May 2011.
[38] K. Grolinger, W. A. Higashino, A. Tiwari, and M. A. Capretz, "Data management in cloud environments: NoSQL and NewSQL data stores," *J. Cloud Comput., Adv., Syst. Appl.*, vol. 2, no. 1, p. 22, 2013.
[39] M. Klems, D. Bermbach, and R. Weinert, "A runtime quality measurement framework for cloud database service systems," in *Proc. 8th Int. Conf. Qual. Inf. Commun. Technol.*, Sep. 2012, pp. 38–46.
[40] C. R. Chang, M.-J. Hsieh, J.-J. Wu, P.-Y. Wu, and P. Liu, "HSQL: A highly scalable cloud database for multi-user query processing," *IEEE Trans. Comput.*, vol. 30, no. 4, pp. 943–944, Jun. 2012.
[41] L. Zhou, N. Chen, S. Yuan, and Z. Chen, "An efficient method of sharing mass spatio-temporal trajectory data based on cloudera impala for traffic distribution mapping in an urban city," *Sensors*, vol. 16, no. 11, p. 1813, 2016.
[42] B.-R. Chang, H.-F. Tsai, Y.-C. Tsai, C.-F. Kuo, and C.-C. Chen, "Integration and optimization of multiple big data processing platforms," *Eng. Comput.*, vol. 33, no. 6, pp. 1680–1704, Aug. 2016.
[43] K. Li, F. Su, X. Cheng, W. Chen, and K. Meng, "The research of performance optimization methods based on impala cluster," in *Proc. 16th Int. Symp. Commun. Inf. Technol. (ISCIT)*, Sep. 2016, pp. 336–341.
[44] E. Estevez-Rams, C. Perez-Demydenko, B. A. Fernández, and R. Lora-Serrano, "Visualizing long vectors of measurements by use of the Hilbert curve," *Comput. Phys. Commun.*, vol. 197, pp. 118–127, Dec. 2015.
[45] *OpenGIS Observations and Measurements—Part 1-Observation Schema (Version 1.0.0)*, OGC Standard 07-022r1, OGC, 2007, p. 73.
[46] *OpenGIS Observations and Measurements—Part 2-Sampling Features (Version 1.0.0)*, OGC Standard 07-002r3, OGC, 2007.
[47] Apache. (Aug. 1, 2019). *Apache Impala 3.2.0*. [Online]. Available: https://impala.apache.org/downloads.html
[48] Apache. (Aug. 1, 2019). *Impala: A Modern, Open-Source SQL Engine for Hadoop*. [Online]. Available: http://cidrdb.org/cidr2015/Papers/CIDR15_Paper28.pdf
[49] A. Bröring, C. Stasch, and J. Echterhoff, *OGC Sensor Observation Service Interface Standard (Version 2.0)*, OGC Standard 12-006, OGC, 2012.
[50] A. Bröring, J. Echterhoff, S. Jirka, I. Simonis, T. Everding, C. Stasch, S. Liang, and R. Lemmens, "New generation sensor Web enablement," *Sensors*, vol. 11, no. 3, pp. 2652–2699, 2011.
[51] N. Chen, L. Di, G. Yu, and M. Min, "A flexible geospatial sensor observation service for diverse sensor data based on Web service," *ISPRS J. Photogramm. Remote Sens.*, vol. 64, no. 2, pp. 234–242, Mar. 2009.
[52] M. Botts, G. Percivall, C. Reed, J. Davidson, *OGC (R) Sensor Web Enablement: Overview and High Level Architecture*, vol. 4540, S. Nittel, A. Labrinidis, A. Stefanidis, Eds. Geosensor Networks, 2008, pp. 175.
[53] S. Nishimura, S. Das, D. Agrawal, and A. E. Abbadi, "MD-HBase: A scalable multi-dimensional data infrastructure for location aware services," in *Proc. IEEE 12th Int. Conf. Mobile Data Manage.*, Jun. 2011, pp. 7–16.
[54] D. Han and E. Stroulia, "HGrid: A data model for large geospatial data sets in HBase," in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, Jun. 2013, pp. 910–917.
[55] C. Cai, Y. Gao, L. Pan, and J. Zhu, "Precise point positioning with quad-constellations: GPS, BeiDou, GLONASS and Galileo," *Adv. Space Res.*, vol. 56, no. 1, pp. 133–143, Jul. 2015.
[56] 52°North. (Jul. 7, 2019). *SOS Implementation*. [Online]. Available: http://www.envirocloud.se/52n-sos-webapp/index
[57] C. Dobre and F. Xhafa, "Intelligent services for big data science," *Future Gener. Comput. Syst.*, vol. 37, pp. 267–281, Jul. 2014.

LIANJIE ZHOU (Student Member, IEEE) received the B.S. degree in remote sensing science and technology from Chang'an University, Xi'an, China, in 2011, and the Ph.D. degree from the State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing (LIESMARS), Wuhan University, China, in 2017. He is currently a Postdoctoral Fellow at the Shenzhen Key Laboratory of Spatial Smart Sensing and Service, Shenzhen University, Shenzhen, China. His current research interests include spatiotemporal data modeling, sensor web, smart city, and the IoT.

QINGQUAN LI received the Ph.D. degree in geographic information science and photogrammetry from the Wuhan Technical University of Surveying and Mapping, China. From 1988 to 1996, he was an Assistant Professor with Wuhan University, where he became an Associate Professor. Since 1998, he has been a Professor with Wuhan University. He is currently the President and a Professor with Shenzhen University, China. He is a Professor with the State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University. He is also the Director of the Shenzhen Key Laboratory of Spatial Smart Sensing and Service, Shenzhen University. He is an Academician of the International Academy of Sciences for Europe and Asia.

WEI TU received the B.E. and Ph.D. degrees in computer science from Wuhan University, Wuhan, China, in 2007 and 2013, respectively. He is currently an Associate Professor at the Shenzhen Key Laboratory of Spatial Smart Sensing and Service, Shenzhen University, Shenzhen, China. His research interests include spatiotemporal data modeling, spatiotemporal data analysis, and spatiotemporal data mining.