H-TDMS: A System for Traffic Big Data Management

Xingcheng Hua[1], Jierui Wang[1], Li Lei[1], Bin Zhou[2], Xiaolin Zhang[2], and Peng Liu[1](✉)

[1] College of Information Science & Electronic Engineering, Zhejiang University, Hangzhou 310027, China
{hua2009x,jrw,leili,liupeng}@zju.edu.cn

[2] Zhejiang Uniview Technologies Co., Ltd., Hangzhou 310051, China
{zhoubin,zhangxiaolin}@uniview.com

Abstract. Massive traffic data is produced constantly every day, causing problems in data integration, massive storage, high performance processing when applying conventional data management approaches. We propose a cloud computing based system H-TDMS (Hadoop based Traffic Data Management System) to capture, manage and process the traffic big data. H-TDMS designs a configurable tool for data integration, a scalable data scheme for data storage, a secondary index for fast search query, a computing framework for data analysis, and a web-based user-interface with data visualization service for user interaction. Experiments on actual traffic data show that H-TDMS achieves considerable performance in traffic big data management.

Keywords: Traffic big data · Cloud computing · Data integration · Secondary index · Data analysis

1 Introduction

The last few years have witnessed an explosion of traffic data due to the rapid improvement in Intelligent Transportation System (ITS). Surveillance system plays an important role in modern intelligent traffic management and produces massive and complex traffic data every day. Usually traffic data is stored as records, which are metadata extracted from the collected media information such as images and videos. The volume of records of a big city in China may exceed one hundred billion in a year.

In order to fully exploit traffic big data potential, there remain many technical challenges that must be addressed. The most critical challenges of traffic big data management system are: (1) integrating data from heterogeneous sources in different formats to solve the problem of Data Island (data sets in isolated storages with different specifications); (2) providing high availability and scalability to support large volume of collected data; (3) equipping enormous processing capacity to handle the analyzing of the traffic data; and (4) providing diverse mining algorithms and models for deep analysis, such as criminal detection and risk pre-alarming [6,10,14,20]. All these problems call for well-adapted infrastructures which can efficiently handling traffic big data integration, indexing and query, and mining and analysis.

Cloud computing in current era plays a critical role when conventional data platforms fail in the "Big Data" scenario. Hadoop [1] is a popular framework for cloud computing running on commodity hardware. With the advantage of Hadoop, we propose a cloud computing based system H-TDMS, providing traffic big data management to support decision making and knowledge discovery. This proposed system has several key features:

(1) Flexible data import and distributed data storage
H-TDMS integrates a flexible and efficient tool to capture and extract data from various databases and provides a storage system for massive traffic data with high performance and sufficient scalability based on a distributed database.
(2) Fast data indexing and query
H-TDMS adopts a secondary index structure to build a lightweight and powerful search engine. The answer of a search query is returned back within a tolerable response time limit, usually in seconds.
(3) Intelligent analysis and mining
H-TDMS integrates and encapsulates diverse algorithms and models for traffic characteristics analysis and criminal detection. It provides both on-line and off-line data processing services to support traffic management and public security issues.
(4) Web-based user-interface and data visualization
H-TDMS provides a web-based user-interface to hide the complexity for accessing and managing data. Analysis results are interpreted by data visualization to help produce and comprehend insights from massive traffic data.

The rest of the paper is organized as follows. Section 2 outlines the the background and related work. Section 3 presents our design in detail. Section 4 gives the evaluation results based on the prototype system and Sect. 5 concludes.

2 Related Work

Big data and cloud computing have brought great opportunities for managing data. Hadoop is a framework for cloud computing, including a distributed file system HDFS (Hadoop Distributed File System) and a parallel processing framework MapReduce. Based on HDFS, HBase [2] is developed as a scalable, distributed database that supports data storage for large tables. Sqoop [4] is an open source software used for efficiently transferring data between Hadoop and structured relational databases (e.g., PostgreSQL [5]). Spark [3] is a fast framework for large-scale data processing. Compared with MapReduce, Spark runs some programs faster due to its in-memory computing.

Hadoop was proven to be an efficient framework for big data storage and query. Hadoop-GIS [7] stored large scale spatial data in HDFS and built a spatial index to support high-performance spatial queries. SpatialHadoop [9] was designed specifically to handle huge datasets of spatial data, which employed a two-level spatial index structure and some efficient spatial operations. Lee et al. [11] presented a lightweight spatial index for big data stored in HBase. Le and Takasu [15] proposed a scalable spatio-temporal data storage for ITS based on HBase and a spatio-temporal index structure using a hierarchical text-encoding algorithm. However these systems supported limited query constraints. As we know, more complex search queries should be provided for traffic big data.

Data mining has attracted wide attention as an approach to discovering information from traffic data. Moriya et al. [13] developed an algorithm using feature-based non-negative matrix factorization to predict the number of accidents and cluster roads to identify the risk factors. Benitez et al. [8] presented a two-step trajectory pattern recognition process including a k-means clustering and a classification over a Self-Organizing Map. Yue et al. [19] proposed a multi-view attributes reduction model for discovering the patterns to manage traffic bottleneck. Lv et al. [12] utilized a deep learning approach considering the spatial and temporal correlations inherently to predict the traffic flow. Xu and Dou [17] implemented an assistant decision-supporting method for urban transportation planning. Since so many works studied accident detection, pattern recognition, traffic flow prediction, investment decision, etc., diverse data mining approaches could be adopted in H-TDMS.

Several traffic big data platforms, based on cloud computing, were researched in recent years. RTIC-C [18] was a system designed to support traffic history data mining based on MapReduce framework. Kemp et al. [10] presented a big data infrastructure for managing data and assisting decision making for transport systems using service oriented architecture. Xiong et al. [16] discussed the design of ITS, including the current situation and future trend of related research and development areas. Different from these solutions, H-TDMS focuses more on how to provide data integration, search query, data analysis and user interaction in one system for traffic big data management, and achieve considerable performance based on cloud computing techniques.

3 System Design

Cloud computing is an inevitable trend for traffic data processing due to its great demands on big data analysis and mining. H-TDMS constructs three layers to meet these demands, as shown in Fig. 1. The data layer stores all the massive traffic data and provides high read/write performance when supporting transparent usage of physical resources. The processing layer provides diverse modular functions for upper layer services, and helps improve performance by parallel-based data processing. The application layer provides the entrances for users to call functions of the lower layers as well as http-based services for end users to access and use H-TDMS.

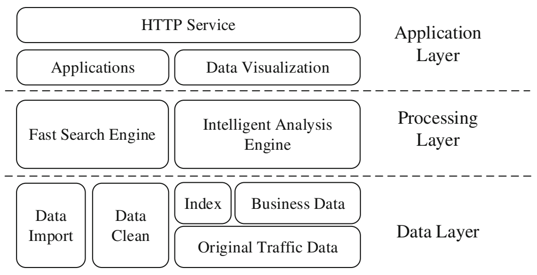
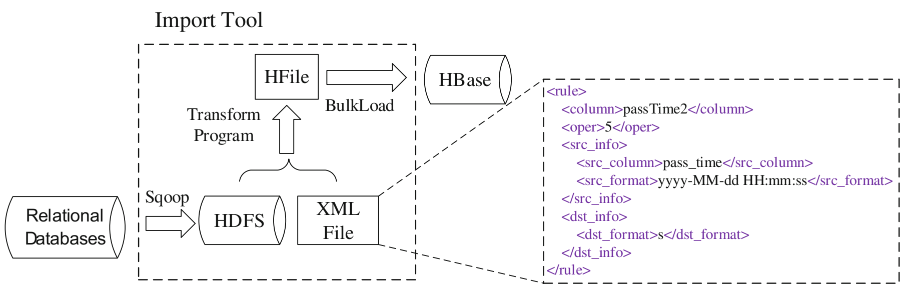


Fig. 1. H-TDMS architecture

3.1 Data Collection and Storage

Traffic data is stored as records in heterogeneous databases in different formats due to historical reasons (e.g., different times of construction, different application scenarios, and different equipment manufacturers). H-TDMS aims to integrate all kinds of data into one distributed database to fully exploit their potential. There is a critical need to flexible import data from various databases. A data import tool is designed as shown in Fig. 2. Though the process of crawling real-time data from the surveillance system is not shown in the figure, H-TDMS supports collecting real-time data into its storage system directly as well.



```
<rule>
  <column>passTime2</column>
  <oper>5</oper>
  <src_info>
    <src_column>pass_time</src_column>
    <src_format>yyyy-MM-dd HH:mm:ss</src_format>
  </src_info>
  <dst_info>
    <dst_format>s</dst_format>
  </dst_info>
</rule>
```

Fig. 2. Data import tool

Sqoop is used for transferring data between HDFS and structured relational databases, which are frequently applied to store traffic data. The transform program is capable of transforming the data to the target format and storing it in the low-level storage files of HBase called HFiles. The HFiles are moved to the regions of a table by BulkLoad, which is a function native supported by HBase. Both Sqoop and the transform program are based on MapReduce to accelerate their processes. An XML file which specifies the transform rules including the target field of a record, the source field of a record and the transform operation between them, is another input used in the transform program to guarantee the flexibility of the import tool. An example of time transformation is shown in Fig. 2.

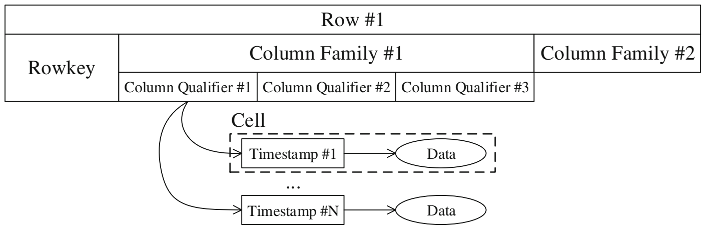


Fig. 3. HBase four-dimensional data model

H-TDMS builds a storage system based on HBase due to its high performance and scalability. The HBase data model is shown in Fig. 3. A well-adapted data scheme is designed to store traffic record data. A complete record consists of a list of fields including the information about a vehicle and its passing events. Table 1 shows the format of the obtained record. Every day all records are stored in a table named by date (e.g., *Table_20160101*) for management reasons. Each record is indexed with a unique rowkey calculated according to the *recordId* and stored in a row of the table. Some key fields such as *PlateCode* and *TollgateCode* are stored in a cell separately and others like *pic1Name* and *relateVideoAddr* are concatenated and stored in one cell to obtain the most considerable performance. Because there is a trade-off between one-column based and multi-column based table structures. The former achieves faster import speed but less access flexibility while the latter is just the opposite.

Table 1. Record format

Field	Example	Field	Example
recordId	1	backendPlateCode	浙A12345
tollgateCode	222051	vehicleColor	A
laneIndex	4	vehicleSpeed	10
passTime	00:00:00	pic1Name	...
plateCode	浙A12345	relateVideoAddr	...
plateColor	1	...	...

There is a lot of business data to be stored while the system is running, including fundamental data like road network information and result data generated by some applications. As a result, many different HBase tables are constructed to hold the corresponding data for maintenance and extension.

3.2 Fast Search Engine

A fast search engine is implemented to support search query, especially multi-condition search and fuzzy search. Equipped with this engine, user could search out the records whose fields containing the specified values within a tolerable response time limit. Since data access is done by relating rowkeys to values in HBase. The problem here is how to figure out the rowkeys of the required records as soon as possible.

H-TDMS builds a secondary index and uses a customized calculation method to figure out the required rowkeys. Logically data in HBase tables is stored in alphabetical order of rowkeys, which are used as a primary index in fact. However there is no way but scanning and filtering the whole table to search out the required records if the rowkeys are not known. Unfortunately, the filter operation in HBase is quite slow and inefficient. HBase would get all the records in a table and then check the content to find out the records that contain the specified values. Instead of using filter operation, a secondary index is designed to relating the specified values to their rowkeys. Figure 4 (a) shows the construction of table *Index_Tollgate* for instance. There is one index table for one specified field of a record and all index information for that field goes into the same index table. When a row is put into the record table, the index information is put into the corresponding index tables. Table *Index_Time* is constructed in another way shown in Fig. 4 (b), utilizing the timestamps generated by a convert function. Any time interval can be specified by setting the range of timestamp. To support fuzzy search of *plateCode*, table *Index_PlateX* is constructed similar as table *Index_Tollgate*, but a special management is employed. When a row is put into the record table, seven rows of index data are generated and put into table *Index_PlateX* at the same time as shown in Fig. 4 (c). A fuzzy search of *plateCode* can be decomposed into several scans, in which the prefixes can be obtained by shifting the *plateCode*. As the rowkeys with the same prefix are stored at a near place in HBase, the scans can be completed very soon. Other index tables such as table *Index_Color* and table *Index_Speed* are constructed in a similar way as table *Index_Tollgate*.

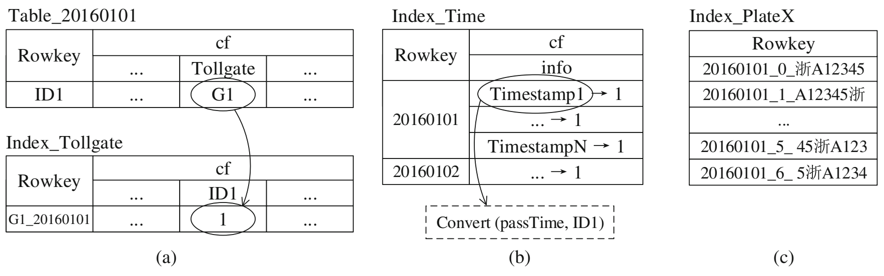


Fig. 4. Construction of index tables

Based on the index tables, the fast search engine defines two basic operations "OR" and "AND" to calculate the index information. Index information is obtained from the index tables and stored in bit sequences, in which the positions of "1"s represent the rowkeys in record tables. "OR" and "AND" are bit-wise operations which are quite fast for processor to perform. The whole process is divided into several sub-processes according to the time condition, usually by the day. In a sub-process, a multi-condition search is decomposed into several steps. Firstly, the search engine decomposes the conditions to get the corresponding index data from the index tables. For each kind of index information, each row of the index data is stored as a bit sequence, which is initialized with "0"s and inserted with "1"s according to the index data. Then all bit sequences are calculated by "OR"/"AND" operations to generate a unique bit sequence for each kind of index information. The calculating logic is determined based on the search conditions in advance. Finally the result bit sequence is generated by calculating the bit sequences of all kinds of index information and the rowkeys of required records are obtained by figuring out the offsets of the "1"s. All of the sub-processes are performed in parallel but commited sequential to make sure that the response can be returned as quickly as possible.

The fast search engine composes of the secondary index and the fast calculation method as discussed above. The secondary index is constructed when the original record data is imported and only sparse "1"s are stored in the index tables. The calculation method takes full advantage of processor's basic bit operations to improve the performance. Both of them make the engine lightweight and powerful.

3.3 Intelligent Analysis Engine

In modern society, people's everyday life has a close connection with traffic issues. In other words, a lot of knowledge can be achieved from the massive traffic data and then be used in traffic management and public security areas. To mine and utilize the knowledge, an intelligent analysis engine is developed based on the fusion of physical, cyber and cognitive spaces. The three-dimensional space model is shown in Fig. 5 (a).

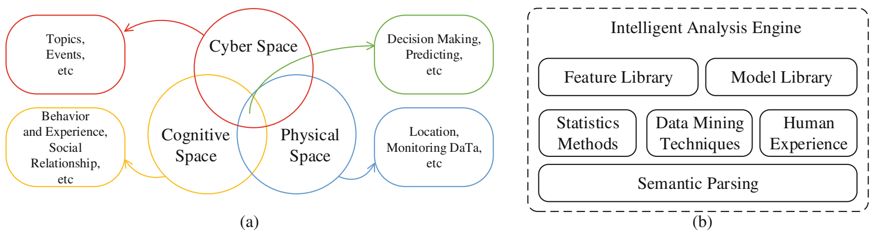


Fig. 5. (a) Three-dimensional space model. (b) Intelligent analysis engine

The conventional analysis methods of pattern recognition and data mining usually collect related data from the physical space which is consisted of fundamental data such as location and monitoring data and process them in the cyber space to form events and topics such as traffic jams. However human's

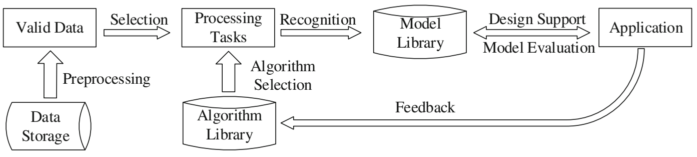


Fig. 6. General process model

experience and judgements are very important and useful in the traffic management and public security areas. Human's feedback will help evaluate and improve the models generated/used during the data processing. So we introduce the human cognitive space which is consisted of human's knowledge into the analysis process. The intelligent analysis engine is hierarchical with three layers as shown in Fig. 5 (b). The semantic parsing layer is applied to define and describe technical terms such as the congestion level of the road section, the speed limit, peak accident times, etc. The algorithm layer integrates and encapsulates various algorithms, including data mining techniques, statistics methods, and human experience. The library layer is adopted to store the features of the traffic and analysis models for prediction, classification, etc.

The intelligent analysis engine is constructed based on Spark instead of MapReduce, because data mining and statistics algorithms benefit a lot from Spark's in-memory computing. Since there is no general one fits all solution in Hadoop, application development is always ad hoc. However, the general process can be modeled as shown in Fig. 6. Human plays an important role during the process, providing professional experience, evaluating results and returning feedback to H-TDMS. Some of the H-TDMS's applications are illustrated and evaluated in detail in Sect. 4.

3.4 User-Interface and Data Visualization

A web-based user-interface is implemented to provide interaction between user and H-TDMS. A set of RESTful web services are created to exchange data. End users access and use H-TDMS through web browsers. Many operations are

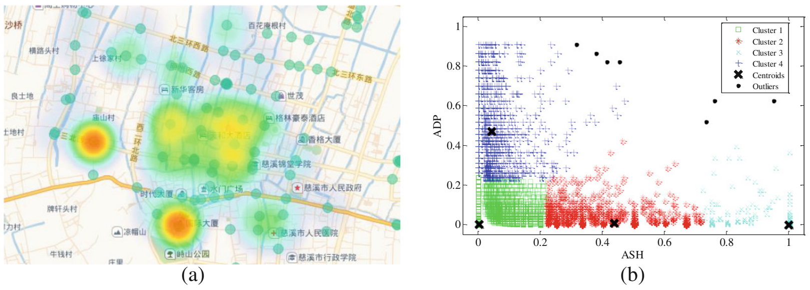


Fig. 7. (a) Sunburst view for a vehicle's activities. (b) Vehicle cluster analysis

defined to allow users to combine their flexibility and creativity. In order to gain insights from the complex analysis results, data visualization is applied to transform various types of data into appropriate visual representations. Figure 7 illustrates two examples of data visualization. Figure 7 (a) is the sunburst view for a vehicle's activities and Fig. 7 (b) shows the result of vehicle cluster analysis.

4 System Evaluation

A prototype system is built on a Hadoop cluster using 3 nodes. One master node takes charge of both cluster management and data processing while the other two slave nodes are only responsible for data processing. The system environment is shown in Table 2. We evaluate our design using actual traffic data of a city in south China, which contains more than 100 million records with a size about 40 GB per month.

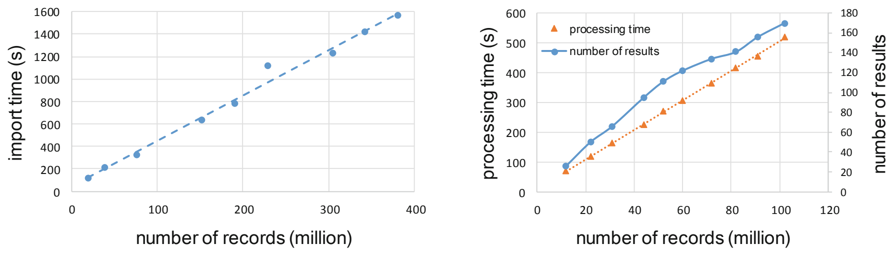
Table 2. System environment

Hadoop environment	
Hadoop version	2.6.0
Sqoop version	1.4.5
HBase version	1.0.0
Spark version	1.3.0
Node environment	
CPU	Intel(R) Core(TM) i7-3770 @ 3.40 GHz
Memory	32 GB
OS	Ubuntu Server 12.04 LTS (64-bit)

4.1 Data Import and Preprocessing

The actual traffic data is imported from a PostgreSQL database. Several record sets with different sizes are selected to evaluate the import tool. The performance is shown in Fig. 8 (a). As is evident from the linear regression line, the speed of data import keeps stabile and exceeds 15 million records per minute when the number of records ranges from 20 to 400 million. The main reason for this stability is that the import tool fully utilizes all the processor resources of the cluster during its MapReduce process.

The traffic data is organized and categorized by the intelligent engine, utilizing statistics methods, to extract necessary information for later use. For instance, many spatial and temporal features of vehicle activities and the information about road conditions are summarized based on the historical vehicle trajectory data, which is generated by combining the relating records.



(a) (b)

Fig. 8. (a) Performance of data import. (b) Fake plate vehicle detection

4.2 Search Query

Several typical query groups are executed to demonstrate the performance of the search engine. The response times of the queries in the same group differ only tens of milliseconds and an example of each group and its result are listed in Table 3. Query1, Query2 and Query3 are used to get the records of a specified time interval, a specified vehicle and a specified tollgate, respectively, and their response times are less than 300 milliseconds. Query4 to Query9 are the most frequently used search query types, in which the plate number of a vehicle is known or partly known. For a search query with a complete plate number such as Query4 and Query5, the response times are less than 500 milliseconds. The response time of the search queries with more tollgates increases a little as more index information is calculated. Although complex fuzzy search queries like Query6, Query7, Query8, and Query9 have longer response times, they can still return results in just a few seconds. Compared Query9 with Query8, the response time does not increase exponentially along with the time interval's growth because of the process division and parallel computing of the search engine.

Table 3. Performance of the search engine. (The Dash (-) represents an unspecified value, the Question Mark (?) indicates an unknown character, and the Star (*) depicts at least an unknown character.)

	Time interval (hour)	Number of Tollgates	Plate number	Response time (ms)	Numeber of required records
Query1	24	-	-	248	4974959
Query2	24	-	浙BRU683	285	16
Query3	24	1	-	289	6092
Query4	24	1	浙BRU683	287	1
Query5	24	10	浙BRU683	488	9
Query6	24	-	浙B*68?	2298	16037
Query7	24	1	浙B*68?	4924	28
Query8	24	10	浙B*68?	5116	288
Query9	168	10	浙B*68?	7721	2757

4.3 Fake Plate Vehicle Detection

A fake plate vehicle is a vehicle using a plate number that is the same as another legal one. Based on the idea that a vehicle can not appear in more than one location within a short time, H-TDMS employs an application, utilizing trajectory data and road conditions, to detect fake plate vehicles. The number of records analyzed ranges from 12 to 102 million. Figure 8 (b) shows the processing time of the application and the number of fake plate vehicles detected. It takes H-TDMS less than 10 minutes to process 100 million records, revealing the high data processing capability of H-TDMS. The result can be verified by checking the related image and video data.

4.4 Vehicle Cluster Analysis

A case in point to show the usability and scalability of H-TDMS based on users' flexibility and creativity is the vehicle cluster analysis, which can be applied to support risk pre-alarming through outlier detection. The user-interface of H-TDMS provides entrances for users to access data and call functions to construct their own applications. We firstly define a feature vector and then apply rules and precedence to the data to create it. The feature vector is consisted of a vehicle's activity of daily period (ADP) and activity of specified hours (ASH), which represent the temporal features of the vehicle. Then a k-means clustering algorithm is applied to classify 1 million vehicles' feature vectors. The result is shown in Fig. 7 (b). It is convenient for users to construct, run and tune their own applications through the web-based user-interface of H-TDMS.

5 Conclusion

Big data has brought great opportunities for resolving transportation problems. In this paper, we provide H-TDMS for traffic big data management based on cloud computing. Our evaluation shows that H-TDMS achieves considerable performance in data integration, search query, data analysis, and provides usability and scalability for users to combine their flexibility and creativity. In our future work, we plan to develop more customized mining services and encapsulate more open interfaces to support more application functionalities.

References

1. Apache hadoop. http://hadoop.apache.org. Accessed 10 Apr 2016
2. Apache hbase. http://hbase.apache.org. Accessed 10 Apr 2016
3. Apache spark. http://spark.apache.org. Accessed 10 Apr 2016
4. Apache sqoop. http://sqoop.apache.org. Accessed 10 Apr 2016
5. Postgresql. https://www.postgresql.org. Accessed 10 Apr 2016
6. Adiba, M., Castrejon-Castillo, J.C., Oviedo, J.A.E., Vargas-Solar, G., Zechinelli-Martini, J.L.: Big data management challenges, approaches, tools and their limitations. In: Yu, S., Lin, X., Misic, J., Shen, X.S., (eds.) Networking for Big Data, pp. 43–56. Chapman and Hall/CRC, February 2016

7. Aji, A., Wang, F., Vo, H., Lee, R., Liu, Q., Zhang, X., Saltz, J.: Hadoop GIS: a high performance spatial data warehousing system over mapreduce. Proc. VLDB Endowment **6**(11), 1009–1020 (2013)
8. Benitez, I., Blasco, C., Mocholi, A., Quijano, A.: A two-step process for clustering electric vehicle trajectories. In: IEEE International Electric Vehicle Conference (IEVC), pp. 1–8. IEEE (2014)
9. Eldawy, A., Mokbel, M.F.: A demonstration of spatialhadoop: an efficient mapreduce framework for spatial data. Proc. VLDB Endowment **6**(12), 1230–1233 (2013)
10. Kemp, G., Vargas-Solar, G., Da Silva, C.F., Ghodous, P., Collet, C., Lopezamaya, P.: Towards cloud big data services for intelligent transport systems. In: ISPE International Conference on Concurrent Engineering, vol. 2, pp. 377. IOS Press (2015)
11. Lee, K., Ganti, R.K., Srivatsa, M., Liu, L.: Efficient spatial query processing for big data. In: ACM International Conference on Advances in Geographic Information Systems, pp. 469–472. ACM (2014)
12. Lv, Y., Duan, Y., Kang, W., Li, Z., Wang, F.Y.: Traffic flow prediction with big data: a deep learning approach. IEEE Trans. Intell. Transp. Syst. **16**(2), 865–873 (2015)
13. Moriya, K., Matsushima, S., Yamanishi, K.: Traffic risk mining from heterogeneous road statistics. In: IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 1–10. IEEE (2015)
14. Shah, N.K.: Big data and cloud computing: pitfalls and advantages in data management. In: International Conference on Computing for Sustainable Global Development (INDIACom), pp. 643–648. IEEE (2015)
15. Van Le, H., Takasu, A.: A scalable spatio-temporal data storage for intelligent transportation systems based on hbase. In: IEEE International Conference on Intelligent Transportation Systems (ITSC), pp. 2733–2738. IEEE (2015)
16. Xiong, G., Zhu, F., Dong, X., Fan, H., Hu, B., Kong, Q., Kang, W., Teng, T.: A kind of novel its based on space-air-ground big-data. IEEE Intell. Transp. Syst. Mag. **8**(1), 10–22 (2016)
17. Xu, X., Dou, W.: An assistant decision-supporting method for urban transportation planning over big traffic data. In: Zu, Q., Hu, B., Gu, N., Seng, S. (eds.) HCC 2014. LNCS, vol. 8944, pp. 251–264. Springer, Heidelberg (2015)
18. Yu, J., Jiang, F., Zhu, T.: Rtic-c: a big data system for massive traffic information mining. In: International Conference on Cloud Computing and Big Data (CloudCom-Asia), pp. 395–402. IEEE (2013)
19. Yue, X., Cao, L., Chen, Y., Xu, B.: Multi-view actionable patterns for managing traffic bottleneck. In: Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
20. Zheng, X., Chen, W., Wang, P., Shen, D., Chen, S., Wang, X., Zhang, Q., Yang, L.: Big data for social transportation. IEEE Trans. Intell. Transp. Syst. **17**(3), 620–630 (2016)