Big traffic data processing framework for intelligent monitoring and recording systems

Yingjie Xia, Jinlong Chen, Xindai Lu, Chunhui Wang, Chao Xu

1. Introduction

With the development of modern surveillance equipment, intelligent monitoring and recording system (IMRS) has been widely used for monitoring traffic system in cities. IMRS can automatically capture the moving vehicle and recognize the vehicle features. We can benefit from the IMRS in the areas such as dynamic road control, traffic management, crimes striking and prevention, and emergency disposal.

IMRS is made up of two components, namely front-end data acquisition system and the back-end data processing platform. The front-end system is in charge of capturing the image data of the passing vehicles, identifying the vehicle features, and sending the vehicle information to the back-end data processing platform [1]. In the back-end platform, many machine learning algorithms are used to filter the data and mine valuable information. The traditional data storage and analysis schemas of IMRS are mainly based on the Relational Database Management System (RDBMS), which can only support some simple database operation tasks such as query, insert or some statistical applications. However, as the scale of data amount increase in a great deal, and the data type becomes more diversified, the traditional RDBMS is unable to deal with these Big Data both in storage and processing.

In order to solve the problem mentioned above, this paper proposes a HBase-based [2,3] big traffic processing framework to improve the vehicle behavior analysis efficiency. The proposed framework provides IMRS the ability to deal with the large scaledata. HBase originates from a technique named BigTable [4] which was proposed by Google for dealing with the big data storage and retrieval task. With the HBase, Hadoop is able to support real-time random access to very large datasets. The advantage of HBase is the high scalability, high availability and fault-tolerance in distributed data storage and real-time big data processing. Although it is not relational and does not support SQL, it can provide queries in large scale sparse dataset on distributed clusters.

The proposed HBase-based big traffic data processing framework for IMRS faces two main challenges. The first challenge is that in HBase the records are stored based on KVS (Key-value Store) [5,6]. But KVS only supports simple search functions. To solve the challenge, we propose a three-table based schema for adapting different requirements in three vehicle behavior analysis cases on IMRS. The second challenge is the data transmission problem. HBase integrated...
MapReduce which has the ability to analyze the distributed data. However, in the data processing process based on MapReduce, the big data transmission will generate huge network traffic which will decrease the performance with the data scale increment. To solve this challenge, HBase Coprocessor (HC) can be used in the data analysis process which requires the HC programs to be deployed on the same physical machine with the data. It avoids the data transmission between different machines and reduces the network traffic overhead.

Overall, our contribution can be summarized as follows:

1. We propose an HBase-based big traffic processing framework to solve the big data storage and analysis problem for the vehicle behavior analysis.
2. We propose a method to design appropriate Rowkey of HBase to increase the data query speed a lot.
3. We introduce HBase Coprocessor based distributed framework to process the data to improve the computing efficiency.

The remainder of this paper is organized as follows. First, Section 2 presents some related research work in big data processing from several areas. Section 3 introduces the proposed framework. Then we introduce a solution for IMRS data storage based on KVS in HBase table in Section 4. In Section 5, we describe a solution to solve the data transmission problem. In Section 6, we describe three cases to illustrate the process of big data analysis based on HBase in IMRS. The experiments study the HBase performance by comparing the processing efficiency of our framework with the three cases in Section 7. At last, we conclude the paper and present directions of future work.

2. Related work

In this section, we will give a review on some related work of big data storage and processing with traditional solutions and HBase-based solutions.

2.1. Traditional solutions for big data storage and processing

In recent years, researchers have recognized the limitations of traditional relational database mainly on the big data distributed storage and processing. To solve the shortcoming of traditional database, various systems have been designed. For example, distributed relational database management systems (DRDBMSs) and in-memory computing [7] are respectively developed for improving capacity of data storage and computing efficiency. However, DRDBMSs is unable to maintain and retrieve data among servers efficiently for the reason that it takes too much time to implement the data consistency [8]. Besides, DRDBMSs are severely restricted by the relational database feature set, e.g. the joins, complex queries, triggers, views, and foreign-key constraints make it difficult to run on a scaled relational database. In-memory computing is the storage of information in the main random access memory (RAM) of dedicated servers rather than complicated relational databases operating on comparatively slow disk drives, for real-time processing big data. It can be used to analyze the common data and load the data probability into memory, so as to improve the efficiency of data processing.

However, because of the shortcomings of linear scalability, the size of the data processed by in-memory computing are limited [9]. DRDBMSs and in-memory computing are not designed for large-scale and scalable distributed processing, since all of those solutions are unable to achieve large-scale linear expansion. Xia et al. [10] designed a parallel computing framework by formalizing computational intensity of big traffic data. A computational domain theory is proposed to formally represent heterogeneous big traffic data and evaluate the computation intensity, which is leveraged to decompose the domain into sub-domains for load-balanced parallel computing. However, this computational domain supports only the structured data, and the traffic image and video in IMRS cannot be directly and accurately encapsulated into the domain.

In addition, some other systems have also been developed to meet the requirement of computing scalability and availability, such as Cassandra [11] and SimpleDB [12], however, both of them provide weak consistency. SimpleDB is another service from Amazon that offers HBase the functionalities which are like those of HBase. However, the value of HBase is an uninterpreted array of bytes, while SimpleDB can only store strings SSDS [13] which has string, number, datetime, binary and boolean data types. Dynamo [14] is another distributed storage system of Amazon. It focuses on writing. Microsoft Boxwood Project [15] provides components with similar functionality as HDFS and HBase. However, Boxwood is just a research project and there is no performance evaluation for large scale of deployment and practical applications [16].

2.2. HBase-based solutions for big data storage and processing

In this paper, HBase-based vehicle tracking is introduced to improve the computing efficiency. HBase belongs to the Hadoop Ecosystem. As an open-source software, the Hadoop platform is easy to program and run applications for processing big data. And the distributed storage [17] and processing [11] strategies are adopted by Hadoop [6]. In this system, the distributed computing framework MapReduce stores data in the distributed file system (DFS) [12], which guarantees the data analysis and processing efficiency [18]. And the redundancy backup mechanism is used to ensure the data security. Moreover, the high fault tolerance of Hadoop allows that it can be deployed on cheap server cluster [19]. Hadoop is made up of distributed file system (HDFS) and distributed computing framework (MapReduce).

As a part of the Hadoop ecosystem, HBase which built on the basis of HDFS, is a column-oriented and scalable distributed storage system with high reliability and performance. It is designed to solve the big data processing problem that traditional relational database faces today. The traditional relational database is lack of scalability and reliability because of the data consistency [20]. While HBase is designed for big data storage and high-speed reading and writing, these data can be accessed at high speed by a large number of concurrent users. Google cooperates with INRIX to use the collected massive GPS data from more than 30,000,000 taxis, trucks, etc. to estimate the traffic states. The massive GPS data are stored on HBase and processed by Hadoop.

In this paper, we design a HBase-based framework using the scalable distributed storage capacity and processing power provided by HBase for big traffic data analysis in the IMRS system.

3. Overview of the proposed framework

Fig. 1 shows the HBase-based framework for the big data analysis in IMRS. The whole process can be divided into three phases. Firstly, the front-end data acquisition system collects the data related to the vehicle passing records which contain the vehicle plate number and passing time information [21]. Those data will be sent to the back-end data processing platform which is built on HBase clusters [22] through IMRS network and distributed storage in the HBase table [23]. Secondly, the back-end data analysis is based on the HBase table structure which greatly affects the data query operations [24]. Thirdly, the HBase Coprocessor is introduced to provide distributed computing capacity and reduce the
data traffic overhead in the network. This will improve the data analysis efficiency.

We use three vehicle tracking applications in the framework, which are collision area analysis, short time passing vehicle analysis, and vehicle trajectory analysis. The main process of vehicle analysis is to query HBase table to obtain the required data. The data query operation is executed on the HRegionserver where some simple data processing can be done by HC. For example, each HRegionserver can compute the union of the query results from multiple HRegions in order to reduce the amount of data that is transmitted to the client [25].

3.1. The process of hbase based data storage

Fig. 2 shows the process that HBase stores the data from front-end data acquisition system. In the front-end system, the image...
collectors automatically obtain the passing vehicle image, and extract the license plate number by the license plate recognition technology. And then, the plate number will be sent to the data processing center to be associated with the vehicle passing time and the road crossing ID through the IMRS special network [26]. In this paper, the whole system is built on legacy system in which the image collectors are installed on the road crossing, and the IMRS system stores some information about the road crossing, for example, the location information which we will use for the HBase-based vehicle analysis [27]. So, the road crossing ID can be regarded as the unique number of image collector. The data processing center receives the records from the front-end system and each record will be stored in the table. For each vehicle tracking application, there is a HBase table respectively designed for vehicle data storage. Therefore, different applications will query different tables, even though there are some overlapped data.

3.2. The design of hbase data table

The main process of HBase-based vehicle tracking is to query the specific record stored in the HBase table. As it is described above, HBase store records by KVS. However, the Rowkey must be designed for the specific application to support complicated functions. Besides, the characteristics of different query methods also must be taken into consideration. In HBase, there are two kinds of query operations: get and scan. Get operation can retrieve the values mapped by a specified Rowkey, and scan operation can return all the values mapped by a range of Rowkeys. So the query efficiency will be improved to store the related records together. There are several tables that should be created in HBase. The first one is the data table that is used to store all IMRS data. Its Rowkey is the combination of vehicle passing time (T), road crossing ID (C), and vehicle plate number (P). Moreover, there are other three index tables that should be created for the specified applications according to the permutation and combination of vehicle passing time (T), road crossing ID (C) and vehicle plate number (P).

The data table in which Rowkey is designed as T-C-P mainly used for storing the passing records of vehicle. The records in the data table are sorted by time, and the records which have the same time are sorted by road crossing ID. The index table in which Rowkey is designed as C-T-P stores the records that have the same road crossing ID and the records are sorted by the vehicle passing time, so it is suitable for querying the vehicles that pass the specified road crossing within a specified time period. The records stored together that have the same plate number are also sorted by the vehicle passing time in the index table in which Rowkey is designed as P-T-C, so it is suitable for querying which road crossing the specified vehicles pass through according to the time sequence.

3.3. IMRS data analysis based on hbase coprocessor

In the process of data analysis, the computing efficiency highly depends on the HBase table. For example, if we want to find out which road crossing the specified vehicle has passed through within a specified time period, the query index table P-T-C will have higher efficiency than the other two tables. In addition, the distributed computing ability based on Hadoop can be used since HBase has integrated MapReduce in the data storage layer. However, high latency is the fatal weakness of the MapReduce distributed computing framework, and it cannot satisfy the requirements of too many real-time distributed queries and computing service. While HBase Coprocessor (HC) is a distributed analysis component similar to MapReduce while greatly simplifies the MapReduce framework and provides a framework that enables the users to flexibly write custom coprocessor. HBase Coprocessor has two implementations, which are Observer mode and Endpoint mode. In the IMRS data analysis, the Endpoint mode will be used to execute some task so as to improve the computing efficiency. The analysis process of HC is described as Fig. 3.

As shown in Fig. 3, the users need to custom their code at first and the code will be deployed and executed on the HRegionserver. Then, the distributed HRegionserver analyzes the IMRS data stored in HBase according to the users custom code. This process is just like the process of map in MapReduce. And at last, the analysis results of each HRegionserver will be sent back to the Client for further processing. This not only makes full use of the computing ability of HRegionserver, but also reduces the data transmission overhead.

HBase library provides a variety of interfaces and classes, so that we can easily use endpoint. In order to build and use our own endpoint, we need to:

1. Define a new interface which extends CoprocessorProtol.
2. Implement the Endpoint interface. The implementation will be loaded into and executed from the region context.
3. Extend the abstract class BaseEndpointCoprocessor.
4. On the client side, the Endpoint can be invoked by two new HBase client APIs: coprocessorProxy and coprocessorExec, which executing against a single region and over a range of regions, respectively.

4. Experiment and results analysis

4.1. IMRS data analysis cases

4.1.1. Collision area analysis

Collision Area analysis is a process to obtain the intersection of vehicles records in different areas by analyzing the passing records. We can reduce the number of vehicles in the process of subsequent analysis by collision area analysis. The detailed analysis of Collision Area includes the following two rules:

1. Single Area analysis: List the vehicle records from one or more road crossings within a selected time period, and a road crossing is called an area.
2. Collision Area analysis: Analyze the records from several specified areas, and find the intersection of those areas from the analysis results.

The following table describes the input arguments, output results and their data specification.
The analysis process mainly includes the following procedures:

1. The query condition is generated in the form of \([< \text{road crossing ID}, [\text{time period}] > ]\) based on the areas list.
2. Use \(HC\) to query related data block of \(HRegion\) in \(C-T-P\) index table, which can quickly obtain the records in the specified road crossing and time period to generate vehicle plate number set in the form of \([< \text{area number, } \text{Set} < \text{vehicle plate number } > ]\).
3. Merge (a union operation) the analysis results of the \(HRegion\) on the same server to reduce the data volume sent to the client.
4. Extract the data like \([< \text{area number, } \text{Set} < \text{vehicle plate number } > ]\) from the nodes that have intermediate data after all \(HC\) calls have returned.
5. Merge the results from all nodes and seek the intersection after calculating the union of the results coming from the same region.

### 4.1.2. Short time passing vehicles analysis

Short time passing vehicles analysis refers to querying the vehicle records in a specified period of time from a certain road crossing to some specific road crossings (Table 1). Table 2 describes the input arguments, the output data and those data specification.

The algorithm of short time passing vehicle analysis is described as follows:

1. Obtain all the vehicle passing records according to the reference road crossing within the specified time period from the \(C-T-P\) index table, and then group the results based on their positions in the \(P-C-T\) table.
2. Query \(P-C-T\) table to obtain the records that meet the following conditions, and then send the results to Clients.
   - i. Have the same vehicle plate number with the query results in (1).
   - ii. The road crossing ID is included in the specified road crossing array.
   - iii. The passing time must be within the specified time which begins from reference road crossing, which is decided by the time threshold described in the table.
3. Acquire the required final result set from the returned records after the above procedures.

### 4.1.3. Vehicle trajectory tracking analysis

Vehicle trajectory tracking analysis is a process that queries the records to find which road crossing the vehicle has passed through within a specified time period. We can calculate the vehicle trajectory by the location of the road crossing. Table 3 describes the input arguments, the output data and those data specification.

### 4.2. Experiment setup

The experiments were performed on a set of physical machines, in the environment of \(HBase\). The environment is built on three nodes, installing JDK 1.7 and Hadoop-cdh-4.0.0. The name node of the \(HBase\) cluster is with Intel Xeon E5620 @ 2.40 GHz CPU, 32 GB Memory and 7.8 TB Disk. The other two data nodes are with Intel Xeon E5620 @ 2.40 GHz CPU, 64 GB Memory and 7.8 TB Disk. The performance of \(IMRS\) is tested through using the dataset stored in \(HBase\) running on \(HDFS\) in a small in-house cluster with different workload compositions. \(HBase\) enables us to scale by simply adding more nodes. As a comparison test, the same dataset was tested on Oracle to evaluate the performance of computing time.

### 4.3. Results and analysis

#### 4.3.1. Performance test on collision area analysis

In the performance test on Collision Area analysis, we chose one area and five areas respectively for tests. In each test, the time period is set as one day, one month, one season and one year, respectively. And the Fig. 4 (a) and (b) shows the performance of the execution on \(HBase\) and Oracle.

From the experimental results, the computing efficiency based on \(HBase\) is much better than that on Oracle. By \(HBase\), it just takes 2.5 min to have the collision area analysis with five areas involving 25 road crossings within the time period of one year. In comparison, Oracle costs 20 min to complete this task. With the increment of the time period, the time consumption of Oracle is greater than \(HBase\). \(HBase\) has a better performance because in the process of query index \(C-T-P\), \(HC\) is used for querying the plate number according to the road crossing ID and time span. The query operation is actually performed on three machines, and \(HC\) merges the query results in the same \(HRegion\), which greatly reduces the amount of data processed on the client.

#### 4.3.2. Performance test on short time passing analysis

In this experiment, the time span is set at 1 day, 7 days, and 360 days, respectively. The number of the specified reference road crossing is set to 1. The experiment is carried on for three times when the time threshold is 3600, 5×3600 and 10×3600 s. Figs. 5–7 show the time consumption comparison of the short time passing analysis with time span increasing.

From the figures, no matter how much the time threshold is set, the time consumption of \(HBase\) is less than Oracle, the execution efficiency of \(HBase\) is usually 4 or 5 times that of Oracle. Moreover, as the time consumption increasing, the efficiency of \(HBase\) is also better than Oracle.

#### 4.3.3. Performance test on vehicle trajectory analysis

In the experiment of vehicle trajectory tracking, we consider the performance comparison in two situations. The first tests were carried out respectively on the \(HBase\) and Oracle in the condition of the same time span, and we compared the performance of multiple vehicle trajectory tracking. The second experiment was
carried out in the condition of the same number of vehicles; we compare the performance of single vehicle trajectory tracking with the time span increasing. In the first experiment, three wildcards were used for the test. Each wildcard character is performed for several tests, and the average values were recorded. Fig. 8 shows the time consumption comparison between two strategies.

From the results of the experiment, it can be found that the vehicle trajectory tracking using HBase strategy outperforms Oracle, and on the average, the query speed of HBase is eight times faster than that of Oracle. In the second experiment, the IMRS data in one day, seven days, one month and one year were used. Fig. 9 shows the results of time consumption of two strategies.

---

**Fig. 4.** The test on collision area against one area and five areas, (a) one area, (b) five areas.

**Fig. 5.** The comparison of time consumption with time threshold set to 3600 s.

**Fig. 6.** The comparison of time consumption with time threshold set to 5*3600 s.

**Fig. 7.** The comparison of time consumption with time threshold set to 10*3600 s.

**Fig. 8.** The comparison of time consumption in the multiple vehicles trajectory analysis.

**Fig. 9.** The comparison of time consumption in the single vehicle analysis with time span increasing.
The results of the experiment show that the time consumption of Oracle strategy is less than that of HBase strategy when the time span is less than one year. However, the time consumption of HBase strategy is relatively stable, which generally maintains at 630 ms, while the time consumption of Oracle strategy substantially increases with time span increasing. So it can be concluded that the HBase strategy has better stability than Oracle strategy.

The time consumption of HBase is higher than that of Oracle when the time span is less than 360 days, it is mainly because the most time-consuming part in this experiment is the process of querying the plate number which matches the specified wildcard, the real process of vehicle trajectory analysis is to obtain data from the index PTC in HBase according to the plate number, and in this process, the time consumption is always at the level of a few hundred milliseconds while having single plate number analysis. In the experiment, there is no need to spend six hundred milliseconds on the trajectory analysis, and most of the time is used to close the query session (nearly 600 ms).

5. Conclusion

This paper introduces a big traffic data analysis framework for IMRS. HBase is column-oriented databases built on Hadoop, and known for have several advantages over traditional row-oriented databases. We provide a solution to solve the traffic data distributed storage and processing based on HBase. In the experiment, we compare the query efficiency between HBase and Oracle. From the results we can see that HBase has the following advantages. The first is high performance, HBase strategy is generally faster than Oracle strategy under the same input parameters, and the performance is not sensitive to the total volume of the data. The second is the scalability, the cluster size can be expanded according to the requirement so as to improve the concurrent capacity and processing speed.

However, HBase also has some shortcomings. For example, it is difficult to design an analysis algorithm to meet the requirement. In order to utilize the characteristic of HBase to develop a high performance data processing program, developers need to be familiar with the operating mechanism and framework of HBase. In addition, more storage space is required to be compared with Oracle strategy because of the redundancy backup mechanism base on HDFS. In the future work, HBase-based vehicle tracking efficiency can be improved by optimizing algorithm.

Acknowledgements

This research is supported in part by the following funds: National Natural Science Foundation of China under grant number 61472113 and 61304188, and Zhejiang Provincial Natural Science Foundation of China under grant number L21F0200004 and LR14F020003 and Zhejiang Xinniao talent projects under grant number 2014R421061.

References


Yingjie Xia received his Ph.D. degree in computer science from Zhejiang University. China. He has been a Postdoc in the Department of Automation, Shanghai JiaoTong University from 2010 to 2012, supervised by Professor Yuncai Liu. Before that, he had been a visiting student at University of Illinois at Urbana-Champaign from 2008 to 2009, supervised by Professor Shaowen Wang. He is currently an associate professor in Hangzhou Institute of Service Engineering, Hangzhou Normal University. His research interests include multimedia analysis, pattern recognition, and intelligent transportation systems.

Jinlong Chen received his bachelor degree in computer science and technology from West Anhui University, China. He is currently a graduate student in Hangzhou Normal University. His research interests include cloud computing and intelligent transportation systems.
Xindai Lu was admitted to the College of Software, Zhejiang University, studying software engineering in September 2006, and received Bachelor Degree of Engineering in June 2010. In the same year, he was recommended for admission to be a master of the school, and received Master Degree of Engineering in June 2012. Now he works in Zhejiang Province Electric Power Company Electric Power Research Institute, engaged in the work of information security.

Chunhui Wang received his bachelor degree in the department of automation from Nanjing University of Posts and Telecommunications, China. He is currently a graduate student in Hangzhou Normal University. His research interests include data fusion, data mining, and intelligent transportation systems.

Chao Xu received her bachelor degree in the School of Foreign Language, East China Institute of Technology in 2011. Now she is a postgraduate student in Institute of Service Engineering in Hangzhou Normal University, Hangzhou, Zhejiang. Her research area includes education technology, English education and educational theory.