Journal of Science and Engineering Research
Vol. 2, No. 1, 2023, pp.41-50
eISSN: 2957-7187

Building and Mining a Big Data Platform for Traffic Data

Shengyi Zhang[1] and Jinan Fiaidhi[2*]

[1,2]Department of Computer Science, Lakehead University, Canada*
[2]jfiaidhi@lakeheadu.ca*

DOI: http://dx.doi.org/10.56828/jser.2023.2.1.5
Article history: Received (December 10, 2022); Review Result (February 9, 2023); Accepted (April 14, 2023)

Abstract: With the popularization of surveillance cameras and the improvement of computing and the internet, the traffic bureau can watch road conditions in real-time, but the analysis and processing of massive traffic data is still worth studying. Big data technologies and cloud services have been widely used in the internet such as monitoring the internet traffic. However, there are not many applications in traditional industries. This project aims to explore how to use the latest technologies, such as Hadoop 3. x, Zookeeper, Hive and Spark Core, Spark Streaming, and Spark MLib to build a modern big data platform and deeply mine and analyze traffic data, to obtain more valuable information.

Keywords: Big data technology, Traffic data, Traffic prediction, Internet traffic, Hadoop

1. Introduction

With the advent of the economy, the number of vehicles, and the number of roads increased, as well as the monitoring points continue to be deployed; it requires a new method to analyze the daily road condition and congestion. The traffic data generated every day is big, and it is difficult to store such data in a traditional relational database such as MySQL for data analysis. In addition, it is difficult to guarantee the quality of data, coupled with the diversification of data formats and fluctuations in transmission rates, the whole problem becomes complicated, and it is difficult to perform efficient data analysis with traditional tools.

In this paper, the aim is to design a modern big data analysis platform, mock road, and vehicle information and save them into a distributed system, and use big data technology to analyze these data, for example, perform road vehicle-related statistical calculations, road flow conversion rate calculation, real-time road congestion monitoring, and road congestion prediction in the future, etc. After generating corresponding reports after analysis, the results will be saved into a relational database, and based on the result, the user can perform queries from the front-end page. In the project, big data and machine learning technologies are integrated to build intelligent decision-making and regulating traffic systems.

Although the technologies of big data, such as Hadoop, Spark, and Flink have been widely used on the Internet, it has not been widely used in traffic monitoring. With the evolution of computer vision and deep learning, more and more surveillance cameras are now installed on the road bayonet. Using different computer vision technologies, such as YOLO, pattern recognition, etc., we can extract vehicle information in real-time, such as license plates, and

car speed. However, many traffic bureaus conduct traffic monitoring by large screens and do not analyze these data from the perspective of time and space to get greater value. If a unified platform can be established to manage all these data, we can combine GIS technologies to present the regional real-time traffic situation spatially, and from that time-wise, big data technologies can be carried out to do statistical analysis on historical data and improve efficiency respectively. In addition, historical data can be used to predict road congestion for some time in the future. Some works have proposed how to establish a big data system for traffic or traffic prediction through a time series model, but there are incomplete, this project will combine different technologies to build real-world front-end and back-end projects using big data technologies.

The paper will explore how to analyze traffic data by modern distributed data analysis frameworks, such as Hadoop, Spark core, Spark SQL, Spark Streaming, and Kafaka, and attempt to forecast traffic conditions based on data collected in the past. In brief, the purpose of this paper is to address the following questions:

1. How to build a distributed big data platform for traffic data, and what technologies are required?
2. How to perform real-world traffic data analysis using Spark Core and Spark SQL?
3. How to predict future traffic congestion, and what methods we can adopt?

Spring Cloud will be used to build different microservices, and this project will use spark core, spark SQL, etc, for traffic data analysis, finally, the results will be written back to the database for front-end query. By using mature spring cloud, various microservices will be constructed, including authentication and authorization, file service, performance monitoring service, code generation service, gateway service, system service, etc., and HTTP Client will be used to connect with big data analysis service, to decouple of the service, and facilitate cluster deployment. The data analysis service will be implemented based on Spark and Hive. Real-time data will be simulated and analyzed through Kafka and Spark Streaming, and the analysis results will be stored in Redis. Model training and prediction will be tested using Spark Mlib, and the model training results will be stored. into HDFS for future use.

2. Literature Review

Fiosina and Maxims [1] analyzed relevant cloud services and application cases for ITS and discussed the structures for building the related processing and mining platform for big data, and provided insights into the transportation business. D'Alconzo et al., [2] presented a survey on big data technologies applied in network traffic monitoring and analysis, the author discussed which methodologies and tools can be adopted to manage the big network traffic data, and briefly listed potential combinations between machine learning and big data. Asadianfam et al., [3] designed a traffic violation detection system, which includes four phases, such as monitor phase, analysis phase, and plan phase, etc. Amini et al., [4] proposed a framework based on distributed computing architecture for real-time traffic management, and built a prototype platform that uses Kafka to handle the stream and integrates Kafka with other software. Nallaperuma et al., [5] built a smart traffic management platform (STMP), which incorporated data streams from multiple data sources, data that can come from IoT smart sensors or social media, which are combined and analyzed to monitor normal and abnormal events, distinguish between recurring and non-recurring traffic events, and the correlation of events. Zeng [6] proposed a framework of smart traffic using big data technologies and presented the key technology in ITS (Intelligent transportation system). Tseng et al., [7] used the API provided by Apache Storm and built a real-time model which is

capable of forecasting traffic jams by evaluating huge streaming information, for example, previous traffic and rainfall. Guerreiro et al., [8] presented a survey of modern open-source technologies and software for big data and explored which tools can be used to estimate traffic congestion.

2.1. Related Technologies

(1) Hadoop

Big data is proposed for such a scenario in which case the collection of files that cannot be stored, analyzed, and processed by a traditional relational database and software within a certain time frame. Big data has four characteristics: 1) from the perspective of data collection, massive amounts of data need to be collected. 2) From the aspect of the growth rate of data size, huge data are added daily and need to be processed daily or in real-time. 3) From the view of data sources and type, there are various data sources and different data are not in the same format. 4) From the value of data, each piece of data carries low valuable information and may be inaccurate due to missing values or network fluctuation during the process. To handle such big data, Hadoop, a distributed computing infrastructure is proposed by the Apache Foundation. Based on Google's previously published papers: Big Table, Google File System, and MapReduce, Hadoop designed a robust and reliable distributed architecture, it can make use of small machines, save files into different machines and perform computation on various nodes, then use a separate machine to collect the result, thus solve the big data problem, the final result can be saved to a relational database for further query. Hadoop has its type of file system and offers an ecosystem that incorporates analytical algorithms, tasks/workflow managers, and NoSQL stores. Hadoop 2. x consists of HDFS, MapReduce, and Yarn. Yarn is responsible for resource scheduling, MapReduce is for data operations, and HDFS is used for file storage.

a. HDFS

HDFS (Hadoop Distributed File System) is a distributed file management system that manages files on multiple machines. First, it can be used to store files and locate files through a directory tree; second, it is distributed, and many servers are combined to achieve its functions, and the servers in the cluster have their roles.HDFS usage scenarios: suitable for writing once and reading multiple times. After a file is created, written, and closed, it does not need to be changed.

b. MapReduce

MapReduce is a programming framework for distributed computing programs, and it is the core framework for users to develop "Hadoop-based data analysis applications".The core function of MapReduce is to integrate the business logic code written by the user and the built-in default components into a complete distributed computing program, which runs concurrently on a Hadoop cluster.

c. Yarn

YARN is mainly composed of three components as Resource Manager (RM), Node Manager, Application Master (AM), and Container. Yarn is a resource scheduling platform responsible for providing server computing resources for computing programs, which is equivalent to a distributed operating system platform, while computing programs such as MapReduce are equivalent to applications running on the operating system. YARN can be used by other frameworks also, for example, Spark can be deployed with Yarn.

The flow of a Hadoop-based data analysis application is as follows:

1. The MR program is submitted to the node where the client is located.
2. Yarn Runner applies an Application from the Resource Manager.
3. RM returns the resource path of the application to YarnRunner.
4. The program submits the resources required for operation to HDFS.
5. After the program resources are submitted, apply for running mr App Master.
6. RM initializes the user's request into a Task.
7. One of the Node Managers receives the Task.
8. The Node Manager creates a Container and generates an MR App master
9. Container copies resources from HDFS to local.
10. MR App master applies to RM to run Map Task resources.
11. RM assigns the task of running Map Task to the other two Node Managers, and the other two Node Managers receive tasks and create containers respectively.
12. MR sends a program startup script to the two Node Managers that have received the task, and the two Node Managers start Map Task respectively, and Map Task sorts the data partitions.
13. Mr. App Master Waits for all Map Tasks to finish running then applies for a container to RM and runs Reduce Task.
14. Reduce Task obtains the data of the corresponding partition from Map Task.
15. After the program runs, MR will apply to RM to cancel itself.

(2) Hive

Hive is a data statistics tool open-sourced by Facebook for solving massive structured logs. It is a data warehouse tool based on Hadoop, which can map structured data files into a table and provide an SQL-like query function (HQL). Hive converts HQL written by users into MapReduce programs for execution.

1. The data processed by Hive is stored in HDFS
2. The underlying implementation of Hive analysis is MapReduce
3. The executor runs on Yarn

(3) HBase

HBase is a distributed, scalable, NoSQL database that supports massive data storage. Logically, the data model of HBase is very similar to relational databases. Data is stored in a table with rows and columns.

But from the perspective of HBase's underlying physical storage structure (K-V), HBase is more like a multi-dimensional map. The data model of HBase includes Name Space (similar to Database in relational databases), Region (similar to tables), Row, Column, Time Stamp, and Cell. HBase includes Region Server and Master. HBase requires HDFS to provide data storage services, and Zookeeper, which is responsible for the high availability of the Master, monitoring Region Server, and managing configuration.

(4) Spark

Spark is a fast, general-purpose, and scalable big data analysis engine developed in Scala, which includes Spark Core, Spark SQL, Spark Streaming, Spark MLib, and more. Spark Core provides the most basic and core functions of Spark. Spark SQL is Spark's component for manipulating structured data. With Spark SQL, users can use SQL or the Apache Hive version of SQL-like (HQL) to query the data. Spark Streaming is a component on the Spark platform that performs streaming computing on real-time data, providing rich API for processing data streams. Compared to Hadoop, Spark is a project for the rapid analysis of

distributed data. Its core technology is Resilient Distributed Datasets (Resilient Distributed Datasets), which provide a richer model than MapReduce and can quickly perform in-memory processing of datasets in multiple iterations to support complex data mining algorithms and graph computing algorithms. The biggest difference between Spark and Hadoop is the data communication between multiple jobs: Spark is memory based while Hadoop is disk-based.

A spark task is divided into Application, Job, Stage, and Task

1. Application: Initialize a Spark Context to run an Application.
2. Job: An Action operator will generate a Job.
3. Stage: Stage is equal to the number of wide dependencies plus 1.
4. Task: In a Stage stage, the number of partitions of the last RDD is the number of Tasks.

Note: Each layer of Application->Job->Stage->Task is a 1-to-n relationship.

(5) Zookeeper

Zookeeper is an open-source distributed Apache project that provides coordination services for distributed applications. The structure of the ZooKeeper data model is very similar to the UNIX file system. It can be regarded as a tree as a whole and each node is called a ZNode. Each ZNode can store 1MB of data by default, and each ZNode can be uniquely identified by its path. The services that Zookeeper can provide include unified naming service, unified configuration management, unified cluster management, dynamic online and offline server nodes, soft load balancing, etc.

(6) Kafka

Apache Kafka is an open-source messaging middleware written in Scala, based on the producer and consumer design pattern. It is an open-source messaging system project developed by the Apache Software Foundation. Kafka was originally developed by LinkedIn and open-sourced in early 2011. The goal of this project is to provide a unified, high-throughput, low-latency platform for processing real-time data. Kafka is a distributed message queue, which classifies messages according to topics when saving them. The sender of the message is called the Producer, and the message receiver is called the Consumer. In addition, the Kafka cluster consists of multiple Kafka instances, and each instance (server) is called the broker. Both the Kafka cluster and the consumer rely on the zookeeper cluster to save some meta information to ensure system availability.

3. Methods

3.1. Environment

Three Virtual Machines (VM) hadoop102, hadoop103, and hadoop104 will be used for building a Hadoop system. Each virtual machine is installed CentOS-7.5-x86-1804, and each machine has 4Core, 4GB RAM, and 50GB ROM.

Table 1: This is an overview of services that runs on each virtual machine

Hadoop102	Hadoop103	Hadoop104
NameNode[1]		SecondaryNameNode[1]
DataNode[1]	DataNode[1]	DataNode[1]
	ResourceManager[2]	
NodeManager[2]	NodeManager[2]	NodeManager[2]
ZkNode	ZkNode	ZkNode
Kafka	Kafka	Kafka

1HDFS.
2Yarn.
Note: Then Hadoop-3.1.3, Hive, Zookeeper-3.5.7, Kafka, Scala, and Spark are installed in all three VM.

3.2. Data Processing

To preprocess the data, the data should be stored in the HDFS cluster, and then the data is cleaned and prepared using Hive. After that, the structured data is put in Hive or HDFS. For the data processing part, SparkMllib or SparkSQLcan be used to analyze and process the data. The analyzed result is stored in the database, such as Redis, Mysql, and Oracle, for web query and display.

Kafka will be used to generate real-time data, it can put different types of data into different topics, then Spark Streaming or Apache Flink can be used to clean the data, analyze and after that, the results will be stored in the database such as Redis, MySQL, for web query and display.

3.3. Task Submission

A spark task can be executed via a bash script or run using a task scheduler platform such as Azkaban.

1. Use bash to execute the task

The script: spark --master url --class jarPathtaskId

```
Process proc = Runtime.getRuntime().exec("task.sh");
proc.waitFor();
```

```
String cmd = "sh /root/test/test.sh" + args[0]+" "+args[1];
Process proc = Runtime.getRuntime().exec(cmd);
String flag;
BufferedReaderbufferedReader = new BufferedReader(new
InputStreamReader(proc.getInputStream()));
While((flag=bufferedReader.readLine())!=null){
System.out.println("result---" + flag);
}
bufferedReader.close();
proc.waitFor();
```

2. Use a task platform to submit and execute the task

After submitting, the data is first stored in MYSQL, and the taskId is used as the unique primary key. If the task execution fails and the previously submitted parameters can be queried from the database to execute the next retry.

3.4. Data Generation

The data is generated using MockData.java to simulate the real-time traffic data and store the data in Hive tables. The monitor_flow_action table is used to monitor traffic flow data.Themonitor_camera_info table indicates the camera number corresponding to a certain monitor location.

Table 2: monitor_flow_action table

column	description
date	Datetime
monitor_id	Monitor ID
camera_id	Camera ID
car	Car plate
action_time	Camera shooting time
speed	Car speed
road_id	Road ID
area_id	Area ID

Table 3: monitor_camera_info table

column	description
monitor_id	Monitor ID
camera_id	Camera ID

After the data is generated and put into HDFS, Hive can be used to generate an intermediate table for Spark to use.

3.5. Data Processing

To provide the backend service, the task is mapped into a POJO using Spring, which consists of fields of task_id, task_name, create_time, start_time, finish_time, task_type, task_status, and task_params. The general pipeline of executing a task and calling spark service is as follows.

1. Parse the task parameters.
2. Obtain the corresponding taskId according to the parameters.
3. Query the parameters from MySQL.
4. Build related RDD according to the parameters.
5. Perform data conversion on the RDD using different operators.
6. Use a customized accumulator to collect the converted data.
7. Put the converted data back into MySQL.

```
set hive.support.sql11.reserved.keywords=false;

CREATE TABLE IF NOT EXISTS traffic.monitor_flow_action(
date string ,
monitor_id string ,
camera_id string ,
car string ,
action_time string ,
speed string ,
road_id string,
area_id string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

load data local inpath'/root/test/monitor_flow_action'
into table traffic.monitor_flow_action;

CREATE TABLE IF NOT EXISTS traffic.monitor_camera_info(
monitor_id string ,
camera_id string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

load data local inpath'/root/test/monitor_camera_info'
into table traffic.monitor_camera_info;

```

3.6. Traffic Prediction

To predict the traffic, we need to average the speed of different roads. The average speed of a road is an obvious measure of the current conditions of the road, normally, the greater the average speed, the more likely traffic congestion will occur.

The general process of using the average speed of the nearby road to predict the current road is as follows:

1. Calculate the sum of the number and speed of vehicles passing by on the road per minute, and get the real-time congestion situation on the road.
2. The predicted road congestion is affected by the current road and nearby road congestion in the past few minutes. The predicted road congestion situation can be predicted based on the road congestion situation of each nearby road and the current road in the first 3 minutes. Use the congestion of each nearby road and the road 3 minutes before the current road as the dimension. Count the congestion of these roads in the first 3 minutes of every minute in the past 5 hours to construct a dataset.
3. Train the machine-learning model.
4. Save the model.
5. Use models to predict road congestion.

4. Results

The final application includes a Spring Cloud-based backend service, which includes the following functions: user management, role management, menu management, and system management, and integrates spark services to provide speeding cars, real-time congestion, monitor condition analysis, and a brief congestion prediction.

4.1. Modules

(1) Speeding car

Use spark Core to analyze the TopNbayonet that vehicles pass through at high speed, count the number of vehicles passing at high speed under each monitor, the number of vehicles passing at medium speed, the number of vehicles passing through normally, the number of vehicles passing at low speed, and then count each monitor, and then convert the results into objects, sort them accordingly and store the top N of the sorted results into the MySQL database for persistence.

(2) Real-time congestion

Analyze road congestion through the real-time average speed of the road, use SparkStreaming for batch processing, and use Kafka to simulate the input data stream, performing adding data to the window, moving out of the window, triggering calculation by Spark Streaming, and calculate the total speed of all vehicles and a total number of vehicles for a given duration of time, to obtain the average speed.

(3) Monitor the condition

According to specific conditions, dynamically check the current status of each bayonet to see if the camera of the current monitor is working normally, count the number of normal bayonets, number of working cameras, and number of normal cameras, and record abnormal information, which includes the monitor id, abnormal camera id.

5. Discussion

Since the data is generated randomly, it has some drawbacks and cannot hundred percent reflect real-world cases, but the process and the overall work pipeline should be the same. Also, the historical data should be saved in a good manner.

References

[1] Fiosina, J. & Maxims Fiosins, J. Ã. (2013). Big data processing and mining for next-generation intelligent transportation systems. JurnalTeknologi, 63(3).

[2] D'Alconzo, A., Drago, I., Morichetta, A., Mellia, M., & Casas, P. (2019). A survey on big data for network traffic monitoring and analysis. IEEE Transactions on Network and Service Management, 16(3), 800-813.

[3] Asadianfam, S., Shamsi, M., & Kenari, A. R. (2020). Big data platform of traffic violation detection system: identifying the risky behaviors of vehicle drivers. Multimedia Tools and Applications, 79(33), 24645-24684.

[4] Amini, S., Gerostathopoulos, I., & Prehofer, C. (2017). Big data analytics architecture for real-time traffic control. In 2017 5th IEEE international conference on Models and Technologies for intelligent transportation systems (MT-ITS), 710-715, IEEE.

[5] Nallaperuma, D., Nawaratne, R., Bandaragoda, T., Adikari, A., Nguyen, S., Kempitiya, T., De Silva, D., Alahakoon, D., & Pothuhera, D. (2019). Online incremental machine learning platform for big data-driven smart traffic management. IEEE Transactions on Intelligent Transportation Systems, 20(12), 4679-4690.

[6] Zeng, G. (2015). Application of big data in the intelligent traffic system. IOSR Journal of Computer Engineering, 17(1), 1-4.

[7] Tseng, F. –H., Hsueh, J. –H.., Tseng, C. –W.., Yang, Y. –T., Chao, H. –C., & Chou, L. –D. (2018). Congestion prediction with big data for real-time highway traffic. IEEE Access, 6, 57311-57323.

[8] Guerreiro, G., Paulo, F., Silva, R., Costa, R., & Jardim-Goncalves, R. (2016). Architecture for big data processing on intelligent transportation systems. An application scenario on highway traffic flows. In 2016 IEEE 8th International Conference on Intelligent Systems (IS), 65-72, IEEE.

[9] The Apache Software Foundation, “Hadoop,” 2014. [Online]. [Accessed 2016].

[10] De Mauro, A., Greco, M., & Grimaldi, M. (2015).What is big data? A consensual definition and a review of key research topics. In AIP conference proceedings, 1644(1), 97-104, American Institute of Physics.

[11] Jiang, W. & Luo, J. (2022). Big data for traffic estimation and prediction: a survey of data and tools. Applied System Innovation, 5(1), 23.

[12] Hadoop Distributed File System (HDFS) [Online]. Available online: https://hadoop.apache.org/hdfs/ (accessed on 20 December 2021).

[13] Apache Hive [Online]. Available online: https://hive.apache.org/ (accessed on 20 December 2021).