

haoqingchen

by Haoqing Chen

Submission date: 07-Sep-2022 09:19PM (UTC+0100)

Submission ID: 185788260

File name: 32954_Haoqing_Chen_haoqingchen_430701_2110404420.docx (2.63M)

Word count: 8492

Character count: 45170



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

A Study of the Driving Behaviours Monitoring

by
Haoqing Chen

Supervisor: Stephen Hole

Project submitted as part of the requirements for the award of
MSC-SEAI Software Engineering and Artificial Intelligence

September 2022

DECLARATION

I, Haoqing Chen declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed : 

Date : 6/9/2022

FORM OF CONSENT

I Haoqing Chen hereby consent that my Project, submitted in candidature for the MSC-SEAI Software Engineering and Artificial Intelligence, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.


Signed —

Date 6/9/2022

Real-Time Traffic Data Processing System And Taxi Track Processing Visualization Platform Based On Spark

Abstract:

Big data technology plays an increasingly important role in analyzing and mining traffic big data. In order to quickly and effectively analyze the operating mode and passenger carrying strategy of taxis, this study designed the effectiveness index model to quantificate and sort the taxis' effectiveness. Taking high-effective taxis as the research object, a data processing and visualization platform is developed based on Spark big data framework. Firstly, higheffective taxis trajectory data are processed to obtain characteristic data for visualization. Then visual analysis is carried out, including high-effective taxis operation characteristics obtained from statistical analysis and interactive chart display, using hexagon grid and DBSCAN algorithm to visualize the hotspot of high-effective taxis carrying passenger points in different time periods, implementing interactive trajectory query based on buffer, and extracting the trajectory-related factor. Finally, the validity and reliability of this platform are verified by GPS trajectory data of Chengdu taxi. Traffic big data is the most basic condition for solving urban traffic problems. It is an important guarantee for formulating macro-city traffic development strategy and construction planning. And it is also an important guarantee for carrying out micro-road traffic management and control. In view of the characteristics of intelligent transportation system such as fast data generation, high real-time performance and large amount of data, this paper constructs a real-time traffic data processing platform based on the combination of Spark Streaming and Apache Kafka to process the data collected by dual base stations. Using time window mechanism to get data from Kafka, and process the data to the database according to the rules. In this study, the system architecture and internal structure of the platform are introduced in detail, and the real-time processing capability of the system is verified through experiments, which can be applied under large-scale and high-concurrency data flow.

Keywords: big data; stream processing system; double station data; Spark Streaming;

With the development of social economy, the people's living standards are improving day by day, the ownership of urban means of transportation is increasing, the contradiction between road supply and demand is becoming increasingly prominent, and various urban transportation problems continue to emerge. In order to improve the current situation of urban transportation. It is not enough to rely solely on increasing road infrastructure, but also to continuously improve the level of informatization and intelligence of traffic management. With the gradual implementation of intelligent transportation system, various types of traffic detection equipment provide a large amount of data, these data open up new ideas for solving urban traffic problems, is the most basic condition for solving urban traffic problems, is to formulate macro urban traffic development strategy planning and micro road traffic management and control of important guarantees [1].

Not only that solving the traffic problem by analyzing various spatial data is a hot topic in the current research. Taxis offer a wide range of and flexible transportation services, and are an important part of urban mobility. Taxi trajectory data records the flow information of urban roads and residents, and the mining and analysis of taxi trajectory data is conducive to the construction of urban smart transportation. In formulating reasonable urban transportation policies, rational allocation of urban public transportation, and alleviating urban traffic congestion.

With the development of economic progress and spatial information technology, the scale of taxi trajectory data has increased exponentially. In order to store and analyze the large amount of data generated, a new architecture is needed to handle taxi trajectory data. The development of big data technology provides the possibility for rapid and efficient processing of large-scale spatial data. Spark is an emerging and outstanding distributed computing framework in the Hadoop ecosystem, with high fault tolerance and high scalability. Leverage the spark framework for parallel storage, parallel computing, and the optimization of in-memory computing. Potential, can accurately and effectively analyze and study urban traffic problems, and realize big data-driven intelligent transportation.

Efficient drivers have rich driving experience and operational strategies, and obtaining high-efficiency taxi trajectory data is more meaningful for research. In this paper, a benefit index model is proposed to quantitatively sort the benefits of taxis,

and high-efficiency taxis are extracted as research objects. On this basis, a Spark-based taxi trajectory processing and visualization platform is designed, which can quickly and effectively analyze the operating mode and passenger carrying strategy of high-efficiency taxis.

Therefore, in view of the characteristics of fast data generation, strong real-time and large amount of data in intelligent transportation systems, it is of great significance to study the data processing methods of big data technology in the field of traffic informatization.

content

Real-Time Traffic Data Processing System AND Taxi track processing and visualization platform Based on Spark Streaming.....	12
Abstract:	12
1 Relevant research status	16
2 Framework and model research	18
2.1 Spark Distributed Computing Framework	18
2.2 Dual base station technology	18
3 Platform architecture.....	19
3.1 Visualize the taxi platform architecture	19
3.2 Passenger hotspot visualization function	21
3.3 Kafka Messaging System	21
3.4 Spark Streaming	23
3.5 Platform System Architecture	24
3.6 Handle the internal structure of the platform	25
4 Experimental results and analysis	28
4.1 Taxi data	28
4.2 Analysis of the operational characteristics of high-efficiency taxis	28
4.3 Efficient taxi trajectory query and visualization	29
4.2 System experimental testing	30
5 conclusion	31
References	32

1 Relevant research status

With the rapid development of Location Based Services (LBS), taxi trajectory data can well reflect the operation rules of taxis, urban traffic conditions and residents' travel characteristics. Researchers in different fields have conducted various studies on taxi trajectories.

In terms of taxi operation analysis, Weng C et al. proposed a taxi operation analysis model based on floating car data (including passenger mileage, mileage utilization, driver work intensity and other parameters). Liu L et al. analyze taxi GPS data to understand the operation mode of taxis, classify drivers, and reveal the spatio-temporal characteristics of taxi operations with high profits. Zhang DQ et al. have been able to dig into taxi GPS trajectories from three perspectives: customer seeking strategy, passenger operation strategy, and service area preference Car hire service strategy.

In terms of taxi hotspot analysis, Liu DY. developed the SmartAdP visual analysis system, which uses taxi track data to determine the hotspot area where billboards are placed. Chang HW predict the distribution of passenger demand in relation to time, weather and taxi location, and improve taxi operations through hot spot analysis through K-means, hierarchical clustering, and DBSCAN Manage. The B-Planner system uses taxi track data to extract passenger get-on and-off hotspots for night bus route planning in Hangzhou. In terms of visual analysis, Wang ZC designed a GPS-based one an interactive visualization system of trajectories of urban traffic congestion, used to explore and analyze traffic conditions in the city. Niu Dandan Visually analyzes passenger travel characteristics from the temporal and spatial dimensions by processing taxi trajectory data. Huang XK proposed visualization method Traj Graph, which stores and visualizes the traffic information recorded by taxi trajectories through graph structure, studies the traffic patterns of cities.

The application of big data technology can quickly and efficiently process large-scale spatial data. In recent years, many distributed computing frameworks for processing spatial data have emerged, such as Spatial Hadoop based on the Hadoop extension. With Hadoop-GIS, Spark-based extensions of Geo Spark, Location Spark and Simba et al. However, the Hadoop MapReduce computing model outputs

intermediate results to disk, generating a large number of I/O operations, making it difficult to achieve large-scale spatial data processing. The performance of the Spark framework is better than that of the Hadoop framework, and its memory-based parallel computing architecture performs better by using RDD.

Through literature research, it is found that the application of traffic big data in vehicle trajectory statistics, accompanying vehicle discovery, traffic flow prediction and fake deck vehicle screening is increasing, but the data needs to be processed before data analysis. With the development of distributed computing technology, big data technology has been widely used in the field of traffic data processing and analysis.

In the literature[2], a short-term traffic flow prediction based on MapReduce and K-neighbor search algorithm is proposed, which can accurately predict the traffic flow in the next time period in a short period of time. Literature [3] in Cao Bo. The distributed parallel data processing framework Spark was used to process and analyze the license plate recognition data generated by the road traffic data processing captured by the traffic camera, and the accompanying vehicles could be found quickly and efficiently based on the parallel FPGrowth algorithm. Literature[4] Based on real-time monitoring collected by traffic cameras. Data, using the parallel computing power of cloud infrastructure, can effectively process large-scale streaming license plate recognition data, instantly detect suspected accompanying vehicles and output the discovery results in real time. After a large number of studies on the literature, it is found that the current application of big data technology in the field of transportation mainly has two problems:

- (1) Most of the offline data is processed. At present, most of the processing of traffic data using big data technology is aimed at offline data, using the MapReduce programming model to abstract the data processing process. Map and Reduce operations, using the idea of "divide and rule" for distributed processing[5]. This processing method is aimed at static historical data, and in the face of low latency real-time data stream processing, its computing efficiency is relatively low, which cannot meet the requirements of real-time processing of stream data.
- (2) Processing for a single source of data. In the current literature, the use of big data technology in the field of urban transportation is mainly to process and analyze a single data source, such as video surveillance data or RFID data, which cannot be effectively processed by multiple data sources, and cannot comprehensively and effectively use the data collected by intelligent transportation systems, so as to make up for the limitations and defects of single

data.

Aiming at the above problems, this paper focuses on how to use big data technology to perform real-time computing and processing of RFID traffic data and video capture data obtained through dual base stations in intelligent transportation.

2 Framework and model research

2.1 Spark Distributed Computing Framework

Spark is an open source distributed computing framework like Hadoop that extends the widely used MapReduce computing model for building large, low-latency data analytics applications. Its main feature is the ability to read and write computations in memory, which improves computing performance. Resilient Distributed Dataset (RDD) is a basic data abstraction in Spark, representing a read-only, partitionable, parallel computable data collection. RDDs can be fully or partially cached in memory and reused in multiple calculations; By distributing tasks to all nodes in real time, the parallelism of computation can be guaranteed. RDD supports two types of manipulation operators: conversion actions and action actions. The conversion action generates a new RDD from an RDD calculation, and the action action is generated. RDD calculates a result and outputs the result to the Spark system.

Spark SQL is a module used by Spark to process structured data, which optimizes data storage in the form of column storage on the data store, which can more easily process taxi track data. The core programming abstraction of Spark SQL is Data Frame, a distributed dataset based on RDD that records structural information with data. At the same time, SQL statements are provided to manipulate and manage the data, and the results are returned in the form of Data Frame.

2.2 Dual base station technology

Dual base station, refers to each traffic data acquisition base station contains RFID traffic data acquisition system and video capture system two sets of data acquisition equipment, that is, when the vehicle passes through the base station, can get RFID passing record and video capture record at the same time[6]. The RFID

traffic data acquisition system is mainly composed of RFID electronic cards attached to the windshield of the vehicle, RFID readers installed on the road, communication transmission networks and information centers. The RFID reader can collect the information in the RFID tag carried by the vehicle, mainly including the record serial number, base station number, lane number, start card reading time, end card reading time, tag number, tag type, license plate number, license plate color, model, body color, etc. Video capture system.

The data attributes collected mainly include the serial number, base station number, lane number, crossing time, license plate number, license plate color, body color, location speed, violation type, image number, whether speeding, etc.

Through the research and analysis of the data collected by the above two kinds of equipment, these two kinds of data have the characteristics of continuous data arrival, real-time data arrival and data scale, so that the system needs to be processed in unit time. The amount of data has increased significantly, and the traditional data processing architecture can no longer meet the actual needs. To solve this problem, this article designs a distribution based on a combination of Spark Streaming and Apache Kafka Traffic data processing platform.

3 Platform architecture

3.1 Visualize the taxi platform architecture

Due to the advantages of high scalability, easy maintenance, and no need to install specific software, this paper adopts B/S software technology architecture for platform construction, and develops a "taxi trajectory processing and visualization platform" that integrates big data processing, visualization and interaction. In the big data processing stage, the platform processes raw data based on spark framework, and extracts high-efficiency taxi trajectory data after data preprocessing and benefit index model calculation. Calculate high-efficiency taxi characteristic data as a data source for subsequent visualization. The visualization stage consists of 3 parts: (1) Operational characteristics analysis visualizes the statistical calculation of high-efficiency taxi characteristic data in the form of charts; (2) Passenger hotspot visualization combines map data and passenger point data to allow users to use

cellular grids and DBSCAN algorithms to visualize hot spots for highly efficient taxi pick-up points at different times; (3) For trajectory query and visualization, the trajectory correlation factor of a single passenger trajectory of a highly efficient taxi is extracted and the trajectory visualization is performed on the map

The big data of taxi trajectory is processed using Spark RDD and Spark SQL, as shown in Figure 2, including data reading and encapsulation, data preprocessing, cost-effective taxi extraction, and visualization feature dataset calculation. Data reading and encapsulation, the taxi track data of a certain day is read into the RDD in the form of strings, the Taxi class is constructed for encapsulation, the RDD is extended to TaxiRDD, the initial RDD is converted into TaxiRDD. data preprocessing, taxiRDD is imported into the established Spark SQL data table, Spark SQL will automatically assign field names and data types according to TaxiRDD, and use SQL statements to preprocess the data.

Including culling abnormal data, data deduplication and other operations. High-efficiency taxi extraction, the pre-processed DataFrame is exported to TaxiRDD, the taxi benefit index is calculated and sorted by the conversion operator provided by Spark, the taxi track data is extracted and the high-efficiency taxi track data is extracted and output taxiRDD<Taxi><list >. Visualize the feature dataset calculation, and the taxi leddd < list for the high-efficiency taxi track data<Taxi> > trajectory points are calculated to obtain a visual characteristic dataset of highly effective taxis. The dataset is saved to a local folder as a data source for subsequent visual analysis.

The analysis of the operational characteristics of high-efficiency taxis can help the operation strategies of taxi drivers. Operational characteristics mainly include the distribution of passengers per hour of high-efficiency taxis and the length of a single passenger hunt

Distribution, distribution of single passenger carrying time, distribution of single passenger carrying distance, etc. The platform offers date range selection, allowing queries on the highly efficient taxi operation features of any date range. Use Spark to read the visual feature dataset of the selected date, perform statistical calculations, and calculate the results in JSON grids

transfer to the front end, using ECharts to implement interactive chart presentation in the browser interface

3.2 Passenger hotspot visualization function

This function provides date range selection and time range selection, and supports cellular grid and DBSCAN spatial clustering algorithm to calculate high-efficiency taxi passenger hotspots.

Grid structures are useful for analyzing large spatial datasets, while honeycomb structures are the best topology covering two-dimensional planes, and hexagons are also seamless polygons with the largest number of edges. This article uses a grid visualization of the cellular structure

The distribution of passenger points, but the grid-based calculation of passenger hotspots reduces the accuracy of hotspot calculation to a certain extent. Therefore, on this basis, the DBSCAN calculation is realized

Method calculation of passenger hotspots. DBSCAN is a density-based spatial clustering algorithm that can divide areas with high density into clusters and discover clusters of any shape in noisy spatial data. Setting the appropriate Eps neighborhood and minimum number of containing points for MinPts to cluster is DBSCAN

The core of the algorithm. According to the definition of the taxi hotspot area range in the literature, eps are set to 50 m. In the case of the Eps parameter determination, minPts values that are too small will produce too many class clusters, and conversely a large number of non-noisy objects will be ignored. Combined with the research data of this paper, after several trials, it was found

MinPts is best set to 20. In this paper, dbSCAN spatial clustering algorithm based on KD-Tree nearest neighbor search is implemented in Spark to improve the problem of long clustering time due to large data volume. Set the parameters Eps: 50 m, MinPts:20, After calculating the clustering results, each class center in the clustering result is used as the core clustering point, and the inverse address is resolved.

3.3 Kafka Messaging System

Kafka was originally developed by LinkedIn as a high-throughput distributed message subscription and publishing system[7], whose throughput can increase linearly as the cluster expands, and is widely used in massive data processing fields. Figure 1 shows the overall architecture of Kafka, consisting of producers and proxies. (Broker) and Consumer (Consumer) three major components; Producer is responsible for collecting and pushing messages to the Broker, while brokers are responsible for

receiving messages and persisting them locally, and Consumer is the true consumer of messages, pulling messages from The Broker and processing them.[8]

Figure 1 Kafka overall structure diagram

A topic can be thought of as a queue, and each message must specify a Topic. In order to increase the throughput rate of Kafka linearly, each topic can be divided into one or more partitions, each of which corresponds to a folder at the physical storage level, including this under the folder partition all messages and index files. All partitions of a Topic are distributed across multiple servers in a Kafka cluster, and the Kafka instance on each server is responsible for reading and writing messages in the partition on that server. When the message arrives, it is appended directly to the partition, which is a sequential write disk operation, so it is very efficient; When consumers consume messages, Kafka reserves an offset for each consumer. Information offset, which is controlled by the consumer, under normal circumstances Consumer increments the offset sequentially after consuming a message, not required Kafka uses a locking mechanism to mark which messages have been consumed by which consumers, thus providing a strong guarantee for Kafka's high throughput rate. Through this partitioning mechanism, the efficiency of message preservation/consumption is ensured, and the throughput rate of Kafka is effectively improved.

For high availability Kafka introduced a replication mechanism, each partition is backed up to multiple servers, based on this scheme, it means that multiple backups of the partition need to be scheduled; Each partition has a server as the lead, responsible for all read and write operations; Other servers act as followers, just synchronize messages. If the dominant server fails, the other follower servers will take over as the new dominant, so that only one server is responsible for reading and writing operations, making the system simpler and more efficient. However, as the dominant server will carry all the request pressure, for better load balancing, Kafka will try to distribute all partitions evenly across the entire cluster, thus ensuring the overall stability and efficiency.

Kafka provides message persistence with $O(1)$ time complexity, high throughput, high reliability, and easy scalability, and can support both offline data processing and

real-time data processing. Based on this, Kafka is fully suitable for distributed traffic big data processing systems.

3.4 Spark Streaming

As a major model in the field of big data computing, the current mainstream stream computing frameworks include Twitter's Storm, Yahoo's S4, Microsoft's Timestream and UC Spark Streaming developed by Berkeley AMPLab.[9]. Among them, Spark Streaming is a distributed streaming computing framework based on the DStream model and built on the Spark computing engine, which can realize high-throughput, fault-tolerant real-time streaming data processing.

DStream (Discretized Stream) is a key program abstraction in Spark Streaming, representing a continuous input data stream that can be created based on input data streams such as Kafka[10]. Internally, a DStream is composed of a set of elastic distributed datasets that are continuous in time RDD sequences; Each RDD contains data streams over time intervals and is an immutable distributed recalculable data set, as shown in Figure 2. Since Spark Streaming is based on the Spark processing engine, its calculation process is actually to divide the input data into a segment of DStream, each DStream is converted into an RDD for Spark, and then the RDD is manipulated into an intermediate result stored in memory through the Spark engine, and finally the intermediate result is overlaid or stored to an external device according to the needs of the business.

Figure 2 Spark Streaming process diagram

Spark Streaming supports fetching data from a variety of data sources, such as TCP Socket, Flume, Kafka, Twitter, Zero MQ, etc. For the Kafka messaging system, Kafka's simple consumer API Create Direct Stream is directly called during system development to read data and create the same number of RDD partitions as Kafka partitions to correspond one-to-one, thereby improving system performance.

On the one hand, Spark Streaming was chosen because the data flow system can be processed in real time and the results can be obtained, and the framework has good scalability, fault tolerance and throughput, which is suitable for application scenarios where data is continuously generated and real-time processing is required. On the

other hand, because it is based on the Spark batch processing engine, it can be placed in the same software stack as batch processing and instant query, realizing seamless sharing of data, thereby reducing development and maintenance costs, and also providing a foundation for offline data processing and mining services in the later stage of the system.

3.5 Platform System Architecture

The system architecture of the distributed traffic flow data processing platform designed in this article using the Kafka and Spark Streaming frameworks is shown in Figure 3, including the data receiving part, the Kafka message subscription system, the data handler, and the data storage part.

The data receiving section is used to receive data sent from the front-end data collector, and to ensure the efficiency and stability of data transmission with the client. Netty should be a high-performance, asynchronous event-driven network Using the program framework [11], the use of multi-threaded or IO multiplexing technology can simultaneously process thousands of client access requests, which is widely used in the field of big data distributed computing, the data processing platform designed in this paper uses Netty as a communication server. Front-end data collection in this system. The integrator acts as the client, and the Netty server acts as the server to collect data request messages sent from the client. Netty data receiving servers, which are considered message producers, are responsible for writing received data to the Kafka messaging system through the producer interface created, the number of different types of traffic. According to the different topics sent to Kafka.

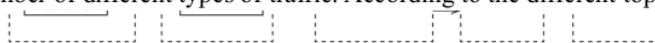


Figure 3 Overall architecture diagram of the system

The Kafka messaging system aggregates data from RFID capture devices and video capture devices into a data stream. The Kafka messaging system spreads different types of data across different topics; Depending on the business processing requirements, the data is forwarded to the Spark Streaming cluster differently. Processing is carried out in the processing module.

The data handler, which runs on a Spark Streaming cluster, is the core module of the system; The Kafka message system is used as a data input stream for real-time

processing. Spark Streaming continuously extracts data from the Kafka message queue in a time window and then processes it for business processing, including both direct business handler and indirect business handler. Direct business processing procedures in this article mainly refer to traffic statistics; The indirect business processing. process first performs data splicing processing, and then processes it according to the business rules, which mainly includes trajectory data statistics and abnormal data extraction. With the continuous expansion of the business, other business processing procedures can be developed as needed for convenient and quick deployment on the cluster.

Data storage section, the result of the processing data is stored. When the data is spliced, statistical and abnormal processed by the data processing program, it is necessary to use different storage methods for different processing results, the main. There should be relational databases, in-memory databases, distributed databases, etc. The vehicle trajectory processing results are stored in HBase, which is divided into multiple different domains, and stored with the reverse order string of the license plate number and time string as the key. The traffic statistics of each collection point are stored in the in-memory database Redis; Store exception data in the relational database MySQL.

3.6 Handle the internal structure of the platform

Figure 4 shows the internal structure of the traffic data processing platform. A Spark Streaming cluster consists of multiple worker nodes, each running multiple Spark Executors, running related business processors on spark Executor. At the same time, the distribution is installed on each worker node. database systems HBase, in-memory redis, and relational databases MySQL; HBase is used to store vehicle trajectory data; MySQL is used to store exception data records; Redis uses to store traffic data from collection points. In order to ensure the scalability of the system, the three storage methods selected in this article, HBase, Redis, and MySQL, can be deployed using a clustering scheme.

Data from video capture devices is written using topic 1 to Kafka messaging system, and data from RFID data collectors is written to Kafka messaging system using Topic2. In order to ensure the throughput of the system, Kafka themes need to be partitioned to improve the system's concurrent read and write capabilities. In the

internal structure of the system, each Spark Executor runs a direct or indirect business handler. At the same time, each message partition of Topic1 has a data consumer (indirect business handler); Topic2 has two data consumers per message partition: one is a direct business handler, which refers to a process traffic statistics handler based on a Spark Streaming cluster; The other is the indirect business handler, which first stitches the video data with Topic1 according to the rules, and then follows the business rules. Real-time data processing, including the handling of vehicle trajectory and abnormal data. One or more business processors form a message consumer group, while ensuring that the data of each partition will be received by at least one consumer in each consumer group.



Figure 4 System Internal Structure Diagram

Figure 4 depicts the internal structure of the system consisting of a Spark Streaming cluster of two worker nodes, each running three Spark Executors, and one Spark Executor responsible for a business handler; Among them, there are two Spark Executors responsible for indirect business processors and one Spark Executor responsible for direct business processing procedures. In the figure, each message topic is divided into four partitions, and the indirect business processor consumes the data of a Topic1 message partition and a Topic2 message partition. The direct business handler is responsible for consuming the data of Topic2's two message partitions. To ensure system performance, the number of partitions of the topic is best set. For multiples of consumers, it is best to ensure that the number of message partitions that each consumer is responsible for processing is equal.

Data splicing programs, which belong to the primary module of indirect business processing procedures. The data splicer continuously extracts data from Kafka according to the time window according to the Kafka message system, and from the time interval set. Continuous Kafka distributed message queue to obtain RFID traffic data and capture data, each cumulative acquisition of data within the specified time period for splicing processing, and then the processing results for other processing procedures, the specific splicing processing as shown in Figure 5. First of all, the RFID data is captured and captured. Individually encapsulated as corresponding RDDs; Then the two types of data RDDs are converted to obtain RDDs in the form of key-value pairs to facilitate the comparison of connection operations, where the keys are strings composed of three fields: license plate number, time, and base station

number. Finally, the RDDs of both kinds of data are based on the key Values are compared and concatenated.

In addition to the data splicing program, the indirect business processor also runs a number of real-time data processing procedures, including vehicle trajectory handlers and abnormal data handlers. Each real-time processor is processed according to the results of the data splicing program and then processed according to the business rules, and the results of the data processing are stored.

Figure 5 The Data Stitching Process

(1) Vehicle trajectory data statistics procedure

According to the formatted RDD of the stitching processing, the vehicle trajectory handler uses the comparison function provided by Spark Streaming to reverse the order processing of the license plate number and time string, and uses the license plate color and traffic time field to filter the data row to obtain the vehicle trajectory data. Taking the reverse order string of the license plate number and time string as the key, the vehicle trajectory data is stored in HBase, and the HBase is divided into several different domains, and the reverse order string of the license plate number and time string is stored as the key

(2) Exception data extractor

According to the formatted RDD of the splicing process, the abnormal data extraction program uses the judgment rules of the abnormal data to filter the RFID passing data and the capture data respectively, extract the abnormal data, and perform the abnormal data

Statistics. The types of data anomalies are divided into missing data, missing license plate color field, capturing image link indeed, number plate inconsistency, and color inconsistency. According to the data anomaly type, first determine whether the RFID data is missing, if there is RFID data, then determine whether the color field in the RFID data exists, whether the capture picture link in the capture data exists, if the field is complete, then determine whether the number plate and color in the RFID data and the capture data are consistent, and finally, the extracted abnormal data is stored in the MySQL database, and the exception type is identified. The relational database MySQL is used for storage.

(3) Traffic flow statistics procedure

The process flow statistics program converts the RFID transit data into key-value

pairs with the collection point field and the time string accurate to the hour as the key; According to the principles of Spark Streaming distributed big data processing, data records with the same key are counted, and then each collection point is counted. The statistical results are summed at the set time intervals to obtain the traffic flow records of each collection point in the corresponding time period. An in-memory database is used to store the traffic flow of each collection point.

4 Experimental results and analysis

4.1 Taxi data

The dataset used in the experiment was the operation trajectory of 13 000 taxis in Chengdu from August 17 to 23, 2014, with data recording time 6:00–24:00 per day, and a total data volume of 18.3 GB. The dataset is stored in days and in txt format, and each track point contains the following attributes: taxi number, latitude, longitude, time, passenger status. Passenger status

State is whether the taxi is carrying a passenger tag (1 for passengers, 0 for empty).

After the efficiency index model calculation, a total of 807 267 trajectories of the top 20% taxis were selected as high-efficiency taxis, and the visual characteristics dataset of high-efficiency taxis was extracted, and the operation rules of high-efficiency taxis were analyzed accordingly

4.2 Analysis of the operational characteristics of high-efficiency taxis

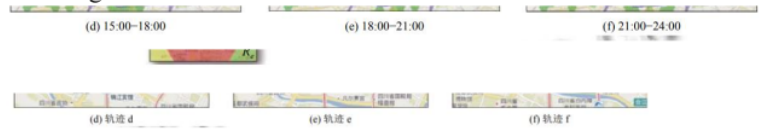
Operational characteristics analysis of a 7-day high-efficiency taxi. The passenger capacity in each hour is counted, and the characteristics of the passenger load of high-efficiency taxis at different times of the day are analyzed. Loads increase dramatically during the 6:00–9:00 hours, reaching the morning peak at 10 o'clock and peaking demand at noon at 13 o'clock. There was a slow downward trend from 13:00–16:00, with a rapid increase in passenger capacity again at 19:00–22:00, and late at 21:00 Peak. In addition, the distribution of passenger capacity on weekdays and rest days is slightly different. On weekdays, there is a downward trend in passenger capacity from 13:00–18:00, with the lowest number of passengers at 18:00; On closed

days, 13:00–18:00 are evenly distributed. The 7-day data is averaged to get a high-efficiency taxi. Analysis results of passenger search time, passenger travel time and passenger carrying distance. The results of the distribution of high-efficiency taxi search time in a single trip are shown, and it is found that the high-efficiency taxi search time is relatively short, and 60% of drivers will find passengers within 2 minutes, indicating that high-efficiency taxi drivers are very familiar with the distribution of passenger sources and road conditions. The results of the distribution of a single passenger carrying time of a high-efficiency taxi are shown, and the distribution results of the single passenger carrying distance are shown. The main service duration of the high-efficiency taxi is within 20 minutes, of which 15 minutes The number of passengers carried within the board accounted for 66.2% of the total, and it was a high-efficiency taxi owner. To serve the time domain. The single passenger carrying distance of 76.1% of high-efficiency taxis is mostly concentrated within 8 km, which is the main service radius of high-efficiency taxis.

4.3 Efficient taxi trajectory query and visualization

visualize multiple trajectories in the start buffer and end buffer DB by plotting the start buffer OB and the end buffer DB on the map, and extract the trajectory correlation factors. Select the data from August 17th and 23rd, set the OB near Chunxi Road subway station, the buffer radius is 155 m, and the DB is set near changshun upper street in the wide and narrow alley, and the buffer radius is 260 m. A total of 31 trajectories were queried, and some of the trajectory visualization results were obtained. Among them, trajectories a, b, d, e frequently choose paths for high-efficiency taxis, and trajectories c and f appear only once, which is a special path. Trajectory visualization combined with trajectory correlation factors can be used to understand the road traffic conditions in the current area and analyze the driver's path selection behavior. Trajectory a trajectory correlated factors at different time periods, it was found that a2 had the longest operating hours and a3 had the shortest operating hours, indicating that a path had large traffic flow at noon, was more congested, had less traffic flow at night, and drove faster. Trajectories d, e, f selected the trajectories of different paths for the same time period, listed the trajectory correlation factors, and found that the trajectory d had the shortest driving distance but the longest operating time, and the trajectory f had the longest driving distance but the shortest operating

time. It can be found that the d path is more congested in the current period, and if you want to reach the destination in a shorter time, you need to choose to detour to avoid the congested area.



4.2 System experimental testing

In order to ensure the data processing capability of the platform, consider the scalability and load balancing of the system, this paper sets up the test environment as shown in the figure. The test environment used vSphere to build a cluster of 6 virtual machines, each configured with 8 cores of CPU, 16 GB of memory, and 80 G of disk, as shown in Table 1. In the experimental process, load runner tool was used to simulate RFID collector and video capture equipment to collect data; Netty acts as a data communication receiver for receiving data sent from LoadRunner and writing it to the Kafka message subscription system; Zookeeper is responsible for coordinating services for distributed applications. Measuring a system's real-time processing power is usually the time it takes to test a system to process a piece of data. The shorter the time for the system to process each piece of data, the stronger the real-time processing power of the system. During testing, will the concurrency of the LoadRunner tool is set to 2000, 4000, 6000, 8000, 10 000, respectively, which means that the data collected by the analog 2000, 4000, 6000, 8000, and 10 000 acquisition devices, respectively, the acquisition frequency is 1/s. Start the system for calculation and processing, observe and record an RFID data and video capture device data from transmission to processing into the time required.

—系列1	0.759	0.761	0.748	0.757	0.763
------	-------	-------	-------	-------	-------

Figure 6 Experimental results

The abscissa in Figure 6 represents the number of concurrency of the data acquisition device; The ordinate represents the calculation processing time. As can be seen from the figure, when the concurrency amount is between 2000 and 10 000, the processing time is 0.748 ms and 0.763 ms fluctuates, and the amplitude of change is small, and its processing time is close to 0.7 ms. According to experimental results, the time required for a piece of data from transmission to processing is milliseconds,

indicating that the system has good real-time processing capabilities, and can completely in large-scale high concurrent data streams application in the scene.

Spark Streaming and Kafka are used in combination, when the amount of data is small, the default configuration of many parameters can meet the application situation, but when the amount of data is large, it is necessary to optimize the system performance from many aspects, on the one hand, try to choose high-performance operators in the development process, such as using map Partitions instead of ordinary maps, making foreach partitions instead of foreach, etc. On the other hand, the parameters are adjusted experimentally according to the actual situation, such as Spark. Streaming's batch time batch duration, Kafka pull volume max Rate Per partition, and partition number of partitions are carried out experimental tuning.

5 conclusion

The proposed traffic data processing platform based on Spark Streaming and Kafka makes full use of the most advanced real-time big data stream processing technology, and the system architecture and internal structure are studied and designed according to the characteristics of traffic data stream. Through research and analysis, the real-time traffic data platform designed in this paper has the following advantages:

- (1) With efficient data processing ability. The system adopts Spark Streaming, an advanced Streaming framework, and Kafka messaging system with high reliability and throughput as the core framework. Because the data processing process of Spark Streaming is implemented based on memory, it has strong real-time processing ability, and its throughput rate is 100 times higher than the processing method of traditional architecture. In this paper, the Spark Streaming technology is used to obtain the RFID traffic data and capture data from the continuous Kafka distributed message queue at a set time interval. The data within the set time period is accumulated each time, and then splicing and processing are carried out according to the data. Using the time window mechanism, the data waiting problem can be effectively solved and the data processing ability of the platform can be improved.
- (2) It has strong scalability. The Kafka messaging system, Spark Streaming cluster, data storage (HBase, Redis and MySQL) and Netty data communication server can all adopt distributed cluster architecture with strong scalability. Therefore, in the actual

application, it is only necessary to design the system deployment scheme according to the current processing requirements, without too much consideration of the future business expansion requirements, so as to reduce the design burden and construction cost of the initial deployment of the project.

(3) With good business function expansion ability. In the current design, this paper only provides three real-time data processing functions: vehicle trajectory data statistics, traffic statistics and abnormal data extraction, which can be used in the future

In addition, this paper implements the taxi trajectory big data processing and visualization platform, and designs the benefit index model to extract high-benefit taxis for visualization analysis. The analysis of operation characteristics is of great significance to the study of high-benefit taxi operation mode and the improvement of taxi benefits. The analysis of the hot spot of passenger carrying is conducive to the rational allocation of urban public transport and the improvement of passenger carrying efficiency. Trajectory query and visualization can be used to analyze urban road traffic and study the influence of trajectory related factors on path selection behavior. In order to

The taxi trajectory data of Chengdu City is used as a research example to verify the effectiveness of the platform. In future research, we will continue to improve the platform functions, such as adding three-dimensional visualization, providing passenger finding recommendation functions, and realizing real-time taxi data analysis services. At the same time, we hope that the platform can be applied to different regions to compare the similarities and differences of taxi operation modes and passenger carrying strategies in different regions.

References

- [1] 陆化普, 孙智源, 屈闻聪. 大数据及其在城市智能交通系统中的应用综述. 交通运输系统工程与信息, 2015, 15(5): 45–52. [doi: 10.3969/j.issn.1009-6744.2015.05.007]
- [2] Chen C, Liu Z, Lin WH, Distributed modeling in a map reduce framework for data-driven traffic flow forecasting. IEEE Transactions on Intelligent Transportation Systems, 20

-
- 13, 14(1): 22–33. [doi: 10.1109/TITS.2012.2205144]
- [3] 曹波, 韩燕波, 王桂玲. 基于车牌识别大数据的伴随车辆组发现方法. 计算机应用, 2015, 35(11): 3203–3207. [doi: 10.11772/j.issn.1001-9081.2015.11.3203]
- [4] 朱美玲, 王雄斌, 张守利, 等. 基于大规模流式车牌识别数据的即时伴随车辆发现. 中国科学技术大学学报, 2016, (1): 47–55.
- [5] 朱继召, 贾岩涛, 徐君, 等. SparkCRF: 一种基于 Spark 的并行 CRFs 算法实现. 计算机研究与发展, 2016, 53(8): 1819–1828.
- [6] 于影, 王静, 宁丹, 等. 基于双基站交通数据采集技术的假套牌车辆识别方法. 交通信息与安全, 2017, 35(4): 76–83. [doi: 10.3963/j.issn.1674-4861.2017.04.010]
- [7] Ranjan R. Streaming big data processing in datacenter clouds. IEEE Cloud Computing, 2014, 1(1): 78–83. [doi: 10.1109/MCC.2014.22]
- [8] 韩德志, 陈旭光, 雷雨馨, 等. 基于 Spark Streaming 的实时数据分析系统及其应用. 计算机应用, 2017, 37(5): 1263–1269.
- [9] 崔星灿, 禹晓辉, 刘洋, 等. 分布式流处理技术综述. 计算机研究与发展, 2015, 52(2): 318–332.
- [10] 易佳, 薛晨, 王树鹏. 分布式流数据加载和查询技术优化. 计算机科学, 2017, 44(5): 172–177.
- [11] 魏晓辉, 李聪, 李洪亮, 等. 支持大规模流数据处理的在线 MapReduce 数据传输机制. 吉林大学学报(理学版), 2015, 53(2): 273–279.
- [12] Makino H, Tamada K, Sakai K. Solutions for urban traffic issues by ITS technologies. IATSS Research, 2018, 42(2): 49–60. [doi: 10.1016/j.iatssr.2018.05.003]
- [13] Liu Y. Big data technology and its analysis of application in urban intelligent transportation system. Proceedings of 2018 International Conference on Intelligent Transportation, Big Data & Smart City. Xiamen, China. 2018. 17–19.
- [14] Gohar M, Muzammal M, Rahman AU. SMART TSS: Defining transportation system behavior using big data analytics in smart cities. Sustainable Cities and Society, 2018, 41: 114–119. [doi: 10.1016/j.scs.2018.05.008]
- [15] Liu L, Andris C, Ratti C. Uncovering cabdrivers' behavior patterns from their digital traces. Computers, Environment and Urban Systems, 2010, 34(6): 541–548. [doi: 10.1016/j.compenvurbsys.2010.07.004]
- [16] eng JC, Zhai YQ, Zhao XJ, et al. Floating car data based. taxi operation characteristics analysis in Beijing. Proceedings of the 2009 WRI World Congress on

Computer Science and

Information Engineering. Los Angeles, CA, USA. 2009.508–512.

[17] Zhang DQ, Sun L, Li B, et al. Understanding taxi service strategies from taxi GPS traces. *IEEE Transactions on Intelligent Transportation Systems*, 2015, 16(1): 123–135.[doi: 10.1109/TITS.2014.2328231]

[18] Liu DY, Weng D, Li YH, et al. Smart AdP: Visual analytics of large-scale taxi trajectories for selecting billboard locations. *IEEE Transactions on Visualization and Computer Graphics*, 2017, 23(1): 1–10.

[19] Chang HW, Tai YC, Hsu JYJ. Context-aware taxi demand hotspots prediction. *International Journal of Business Intelligence and Data Mining*, 2010, 5(1): 3–18. [doi:10.1504/IJBIDM.2010.030296]

[20] Chen C, Zhang DQ, Li N, et al. B-Planner: Planning bidirectional night bus routes using large-scale taxi GPS traces. *IEEE Transactions on Intelligent Transportation Systems*, 2014, 15(4): 1451–1465. [doi: 10.1109/TITS.2014.2298892]

[21] Wang ZC, Lu M, Yuan XR, et al. Visual traffic jam analysis. based on trajectory data. *IEEE Transactions on Visualization and Computer Graphics*, 2013, 19(12): 2159–2168. [doi:10.1109/TVCG.2013.228]

[22] 牛丹丹, 段宗涛, 陈柘, 等. 城市出租车乘客出行特征可视化分析方法. *计算机工程与应用*, 2019, 55(6): 237–243. [doi:10.3778/j.issn.1002-8331.1712-0019]

[23] Huang XK, Zhao Y, Ma C, et al. TrajGraph: A graph-based visual analytics approach to studying urban network centralities using taxi trajectory data. *IEEE Transactions on Visualization and Computer Graphics*, 2016, 22(1): 160–169.[doi: 10.1109/TVCG.2015.2467771]

[24] Eldawy A, Mokbel MF. Spatial hadoop: A map reduce framework for spatial data. *Proceedings of the IEEE 31st International Conference on Data Engineering*. Seoul, Republic of Korea. 2015. 1352–1363.

[25] Aji A, Sun XL, Vo H, et al. Demonstration of Hadoop-GIS: A spatial data warehousing system over MapReduce. *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. Orlando, FL, USA. 2013. 528–531.

[26] Yu J, Zhang ZS, Sarwat M. Spatial data management in Apache Spark: The Geo Spark perspective and beyond. *Geo Informatica*, 2019, 23(1): 37–78. [doi: 10.1007/s10707-018-0330-9]

-
- [26] Tang MJ, Yu YY, Malluhi QM. Location Spark: A distributed in-memory data management system for big spatial data. *Proceedings of the VLDB Endowment*, 2016,9(13): 1565–1568. [doi: 10.14778/3007263.3007310]
- [27] Xie D, Li FF, Yao B. Simba: Efficient in-memory spatial analytics. *Proceedings of the 2016 International Conference on Management of Data*. San Francisco, CA,USA. 2016. 1071–1085.
- [28] 谭亮, 周静. 基于 Spark Streaming 的实时交通数据处理平台. *计算机系统应用*, 2018, 27(10): 133–139. [doi: 10.15888/j.cnki.csa.006592]
- [29] 段宗涛, 陈志明, 陈柘, 等. 基于 Spark 平台城市出租车乘客出行特征分析. *计算机系统应用*, 2017, 26(3): 37–43. [doi:10.15888/j.cnki.csa.005617]
- [30] Mao B, Yu ZC, Cao J. Large scale spatial temporal data visualization based on Spark and 3D volume rendering. *Proceedings of 2016 International Joint Conference on Neural Networks*. Vancouver, BC, Canada. 2016. 1879–1882.
- [31] 李格非, 马蔚吟, 李力. Spark 平台下的凸包问题研究. *计算机工程与应用*, 2018, 54(22): 67–73, 112. [doi: 10.3778/j.issn.1002-8331.1708-0293]
- [32] Qin GY, Li TN, Yu B. Mining factors affecting taxi drivers' incomes using GPS trajectories. *Transportation Research Part C: Emerging Technologies*, 2017, 79:103–118. [doi: 10.1016/j.trc.2017.03.013]
- [33] 孙飞, 张霞, 唐炉亮, 等. 基于 GPS 轨迹大数据的优质客源时空分布研究. *地球信息科学学报*, 2015, 17(3): 329–335.
- [34] Li DQ, Mei HH, Shen Y. ECharts: A declarative framework for rapid construction of web-based visualization. *Visual Informatics*, 2018, 2(2): 136–146. [doi: 10.1016/j.visinf.2018.04.011]
- [35] 陈丽璐, 聂文惠. 基于出租车数据的载客热点与打车热点研究. *计算机系统应用*, 2019, 28(4): 32–38. [doi: 10.15888/j.cnki.csa.006863]
- [36] Lu M, Lai CF, Ye TZ, et al. Visual analysis of multiple route choices based on general GPS trajectories. *IEEE Transactions on Big Data*, 2017, 3(2): 234–247. [doi: 10.1109/TBDATA.2017.2667700]

FINAL GRADE

/100

GENERAL COMMENTS

Instructor

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

