




A Platform Service for Passenger Volume Analysis on Massive Smart Card Data in Public Transportation Domain

Weilong Ding^{1,2}(✉) , Zhe Wang^{1,2}, and Zhuofeng Zhao^{1,2}

¹ School of Computer Science and Technology,
North China University of Technology, Beijing 100144, China
dingweilong@ncut.edu.cn

² Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream
Data, Beijing 100144, China

Abstract. In current public transportation of modern cities, the passenger volume analysis counts the bus passengers in multiple perspectives, and it is significant to optimize the bus scheduling and evaluate transportation capacity. On the smart card data of passengers taking buses, traditional solutions have inherent limitations about long processing delay, inaccuracy result and poor scalability. In this paper, the spatio-temporal correlation with business restrictions is considered, and an effective platform service for passenger volumes analyses are proposed on massive smart card. Our service has been applied in practical usage for three types of passenger volume, and holds minute-level latencies on weekly data with nearly linear scalability in extensive conditions.

Keywords: Spatio-temporal data · Smart card data · Behavior analysis · Passenger volume · Platform service

1 Introduction

Nowadays, smart cards solutions have been adopted extensively in urban environment, and generate massive offline historical data [1]. The large data makes it possible for official governors to achieve intelligent analysis [2]. In public transportation domain, the *passenger volume* analysis counts the bus passengers in multiple perspectives. It is a significant indicator to find hot spots in a city, optimize the bus scheduling, and evaluate transportation capacity in the intelligent transportation system (ITS) [3].

Traditionally, the smart card data with typical spatio-temporal attributes are stored in data warehouse or relational database after necessary data cleaning [4, 5], and the passenger volume is achieved through statistic linear models. Most of those methods are done by interactive SQL (structural query language) or predefined store procedure on small data samples [4, 6]. However, it faces inherent limitations on massive spatio-temporal data. (1) First, the executive latency is intolerable when the involved data size is huge. With the simplified assumptions, traditional methods on small samples [7] only achieve short-term predictive values for limited locations [8] (e.g., prediction for given stations in a periodic five minutes). It suffers long time through database that large

volume of data has to be centralized loaded and scanned many times during the query execution. The I/O of such analyses even cost more time than that of computation itself [9]. As a result, the release of the routine report for passenger volumes is always delayed in metropolises. (2) Second, the analytical accuracy is low in practice because the business spatio-temporal correlation has not been fully considered. Traditional methods always focus on the holistic statistic characteristics to fit the historical observations, and neglect specific passenger behaviors in temporal or spatial perspective. In fact, how to describe those behaviors restricts the accuracy essentially [10]. (3) Third, the analytical scalability during calculation is extremely poor when the data size increases or the infrastructure updates. In either case, traditional methods have to pursue higher-level hardware or software in “scale-up” to redeploy the applications accordingly. It also implies a great deal of financial and man-power expenses.

In this paper, we propose a novel platform service for passenger volume analyses on massive smart card data. The contributions can be summarized as follows. (1) Three types of bus passenger volume, getting-on, getting-off and transfer, are defined by business spatio-temporal characteristics. It is the necessary condition for the accurate analyses. (2) On massive smart card data, each type of bus passenger volume is efficiently achieved with horizontal scalability. Modeled as Hadoop MapReduce jobs, the analyses procedure holds minute-level latencies on weekly data in extensive conditions. (3) Available in a practical project of public transportation domain, our work has brought benefits due to the visualization of the bus passenger volume.

This paper is organized as follows. Section 2 shows the background including motivation and related works. Section 3 elaborates the platform service and the volume analyses for multiple passenger behaviors. Section 4 quantitatively demonstrates effects from the experiment and case studies in various conditions. Section 5 summarizes the conclusion.

2 Background

2.1 Motivation and Assumption

Our work was initiated by *Passenger Big Data Analysis Platform* in Beijing. We cooperated with *E-hualu*, one of the leader Chinese companies in intelligent traffic domain, to deploy a bus scheduling system for more than 30 new night-bus lines in late 2014. The goal of this project is to optimize public transits through Big Data technologies to alleviate traffic jams, improve air quality, and bring regional integration with peripheral *Tianjin* city and *Hebei* province.

In Beijing until the end of 2015, more than 30 thousand buses of nearly one thousand lines adopted smart card readers and 44 million cards had been released to citizens, which would generate 15 thousand records with about tens of gigabyte data daily. Such data from buses has been accumulated day by day as massive historical data. As the data unit, a record of smart card data is typical spatio-temporal union, and contains 13 attributes in Table 1 including entities, timestamps and spatial attribute-groups.

The regular business reports for bus service are released from our system to evaluate the traffic management and passenger guidance in a macroscopic viewpoint.

Table 1. The record structure of smart card data.

Attribute	Notation	Type
<i>card_ID</i>	Identity of smart card	Entity
<i>line_ID</i>	Identity of bus line	
<i>bus_ID</i>	Identity of bus	
<i>begin_time</i>	Timestamp of getting-on	Time
<i>end_time</i>	Timestamp of getting-off	
<i>from_station_ID</i>	Identity of getting-on station	Space
<i>from_station_name</i>	Identity of getting-on station	
<i>from_station_longitude</i>	Longitude of getting-on station	
<i>from_station_latitude</i>	Latitude of getting-on station	
<i>to_station_ID</i>	Identity of getting-off station	
<i>to_station_name</i>	Name of getting-off station	
<i>to_station_longitude</i>	Longitude of getting-off station	
<i>to_station_latitude</i>	Latitude of getting-off station	

For example, in rush hours the early warnings and vehicle dispatching could be made for certain busy lines due to their too long departure frequency. Traditionally, such historical data was processed periodically through data warehouse and statistic models after its capture and storage, but the delay during the analysis is too long to endure. For instance, such a monthly report of Beijing usually requires more than half a month to complete. Accordingly, it is required to pursue efficient analyses solution, not only for optimizing bus departure intervals or passenger on-board time, but also for the better data management. That is the just motivation of our work.

In this paper, there are two assumptions.

First, a record must contain the timestamps of both getting-on and getting-off. If one is excluded, it should be inferred by the other one before further processing [11, 12]. In Beijing, there are two charge kinds due to the smart card readers on buses. One is charged by travel counts and a card is read once in a trip at the getting-on time of passengers; the other is charged by travel distances and a card is read twice in a trip at both getting-on and getting-off times of passengers. All the data in this work was generated from the readers of the latter type. It is a sound assumption because such charging is popular progressively. For example, according to the official policies, all the buses in Beijing had updated their readers for distance charging since December 12th 2014.

Second, the fallacious records have been eliminated in advance. Due to the uncertain conditions of devices or storage, the raw data has two main defects. (1) The temporal attributes among the records are inconsistent. For example, we do not know whether 2001-01-01 or 2015-05-31 is the factual date if both values appear in the same record. (2) The missing or illegal records bring business confusion. For example, a bus may seem run too much time without any rest if the records in certain time are lost. Therefore, low data quality is the inevitable obstruct for data analysis [13]. Here, we employ dedicated data cleaning method [14] on massive spatio-temporal data to guarantee temporal consistency and semantic legality.

2.2 Related Work

As a hot topic in public transportation domain, the research about bus passenger volume can be classified in three categories according to the involved technology.

Database is the widely used technology for bus passenger volume analysis. The smart card data is maintained in the persistent storage, and the passenger volume is calculated by interactive SQL or predefined store procedure [4]. Through geographical database, Long et al. [6] uncovered the passengers' commuting pattern in Beijing, and compared their trips with the expensed time and geographical distance in different perspectives. On GTFS (General Transit Feed Specification) data in database, Tao et al. [10] demonstrated a multi-step method to examine the spatio-temporal dynamics of travel behaviors among bus passengers. But those works only concern the limited data instead of the holistic ones in wider time ranges. It would suffer long processing latency due to the heavy IO of loading and scan during analytical processing [9].

Statistic model is also adopted for bus passenger volume prediction. In some time intensive conditions, it can reduce the executive delay dramatically. During the short-time prediction for passenger volume, such models rely on the characteristic of samples. On smart card data, Ma et al. [4] built trip probability models of involved stations, and proposed DBSCAN joint algorithm to identify historical travel patterns and regularities. Zhou et al. [11] proposed OD (origin-destination) matrix to estimate public passenger volume in probability view. Zhang et al. [7] proposed a Kalman filter model to forecast short-term passenger volume on smart card data, vehicle location data and station video data. The accuracy can hold well, but benefits on limited data samples in practice. Moreover, as typical time series approaches, all these works above only exploit time characteristics without spatial consideration. It makes it impossible to exhaustively evaluate bus passengers' behaviors.

Big Data technology is popular nowadays especially on large volume data in scalable Cloud environment [15]. Hadoop ecology has been applied in the transportation domain, with the help of highly utilized virtual resources. Through Hadoop distributed file system, UrbanCPS [16] and coMobile [17] store data from heterogeneous sensors, and predict traffic speeds with human mobility in urban areas. Moreover, in a private Cloud, Xiong's work [3] integrates the transportation data in multiple perspectives. Through Hadoop MapReduce on smart card data and bus GPS data, Zhang et al. [1, 12] analyzed the passenger density to infer crowdedness and evaluate the vehicle scheduling. With analogous solutions, Wang et al. [18] estimated boarding stop time and bus arrival time. Moreover, SMARTBUS [19] shows a composite solution in multiple Hadoop layers. All those works prove their effectiveness in specific business, but none of them considers bus passenger volume analysis yet.

In brief, on massive data of smart card data, challenges still remain to analyze passenger volume efficiently. Therefore, we introduce platform service via Big Data technologies.

3 Bus Passenger Volume Analyses in the Platform Service

3.1 Methodology

We designed a platform service whose architecture is illustrated as Fig. 1.

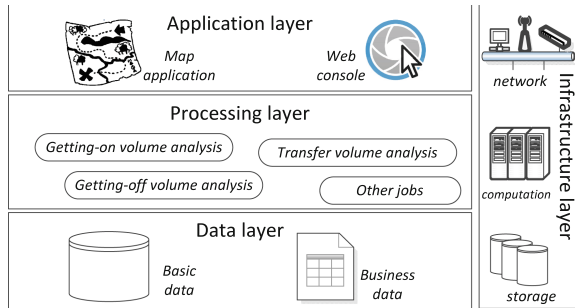


Fig. 1. The service architecture.

The *data layer* maintains the required data. The *basic data* is the essential auxiliary for analyses in the relational database, including locations of station, profile of bus lines and vehicle id, etc. The *business data* is the spatio-temporal records captured from the smart card readers, which would regularly load into the distributed file system after data cleaning. Compared with the basic data, the business data has much larger size (i.e., terabyte vs. megabyte level) with higher updates frequency (i.e., daily vs. monthly).

The *processing layer* provides run-time environment to calculate business analyses like bus passenger volumes. Such analyses are built and submitted by domain experts as calculative jobs, and run as parallel tasks.

The *application layer* shows the results of business analyses with pre-defined configurations. The *map application* visualizes the results in online maps of multiple perspectives. For bus passenger volumes, the results could be demonstrated in a map as the granularity of stations, bus lines, and road network. The *web console* sets the configurations of the whole service, manages the business analyses, and monitors the status of each layer.

The *infrastructure layer* supplies the virtualized resources from a private Cloud. The resources like computation, storage, and network are accommodated elastically on demand.

In fact, it is a typical architecture for Big Data analysis in the public transportation domain, where we focus on the bus passenger volume analyses in this following part.

From Table 1, the smart card data is formally defined first.

Definition 1: Smart card data. The smart card data is generated when a passenger's card is read by reader on the bus. A record as the unit can be represented as $r = (p, b, s_n, t_n, s_f, t_f)$. Here, p is a passenger's card, b is the taking bus, s_n (s_f) is the getting-on (getting-off) station, and t_n (t_f) is the getting-on (getting-off) timestamp at s_n (s_f).

For any station $s \in S$, its *passenger volume analyses* counts the number of passengers in any time slot δ_i , where S is the station set. Divided by the fixed interval length θ , δ_i , is the i^{th} time slot of a day. For example, when $\theta = 1$ h, $|i| = 24$, $\delta_0 = [0:00-1:00)$, $\delta_1 = [1:00-2:00)$, ..., $\delta_{23} = [23:00-24:00)$; the passenger volume would be a 24-dimensional vector, whose element is the count in a δ_i at s . In fact, a passenger must own one of three behaviors at a station: getting-on, getting-off and transfer. Accordingly, the bus passenger volume can be discussed accordingly.

3.2 Getting-On/Off Behavior and Its Passenger Volume Analysis

The getting-on and getting-off analyses are similar due to the symmetrical behaviors, so that only the former one would be fully discussed here.

Definition 2. Getting-on (getting-off) passenger volume. The getting-on (getting-off) passenger volume presented as nPF_s^δ (fPF_s^δ) counts passengers P who get on (off) any bus at station s in a time slot $\delta(t_l, t_r)$. Here, time $t_r > t_l$, \exists a record r of smart card data, $r.p \in P$, $r.s_n = s$ ($r.s_f = s$), $r.t_n \in \delta$ ($r.t_f \in \delta$). It is achieved periodically with a fixed interval length $\theta = |\delta| = t_r - t_l$. Usually, θ is set as 30 min, 1 h or 2 h in practice.

Referring certain location (i.e., station) at given time interval (i.e., time slot), those volumes as an aggregation evaluation reflect the hot degree of that station. The spatio-temporal continuity of individual passengers is kept in the records, but the moving of buses is required in the analysis. A certain bus always runs more than one round-trip in a day, and it is would stop and start at any station in each trip. The time when the bus started (stopped) brings the passengers' getting-on (getting-off) behaviors. However, only some passenger's times instead of that of buses are kept in the records. Therefore, the first difficulty comes from the gap of different temporal semantics. As Fig. 2, there are two trips of a certain bus at a station in a day. The getting-on timestamps of passengers would be gathered in clusters according to the time of bus start in each trip (i.e., t_{s1} and t_{s2} here).

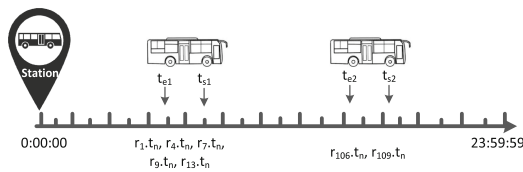


Fig. 2. The moving of a bus at a station.

Therefore, it remains three problems for getting-on (getting-off) passenger volume analysis: how to discriminate each trip of any bus; how to infer a bus's stop and start time at any station; how to sensibly build nPF_s^δ (fPF_s^δ) if waiting period of bus at a station overlaps two adjacent time slots (e.g., if a bus stopped when $t_e = 8:58$ and started when $t_s = 9:01$, the ridership during $[t_s, t_e)$ overlaps the time slots $\delta_7 = [7:00-8:00)$ and $\delta_8 = [8:00-9:00)$).

To solve those problems, we observe the characteristics of data, and propose the following Algorithm 1 with the symbols in definition 1. The algorithm is based the observations below. For a bus b at station s , a passenger's getting-on time must be earlier than b 's start time in a probability larger than 50%; that time of all getting-on passengers in the same trip must cluster according to b 's start time; in the same trip of b , all those times must be statistically positive-skewed because the median of those timestamps is smaller than their mean value. In the line 3–7, getting-on timestamps of different trips are clustered under auxiliary business conditions. In practice, we employ the empirical $\gamma = 420$ (i.e., 7 min), because we have learned from the official documents, a bus of Beijing spends at least 14 min for a round-trip. In the line 9–13, to infer the start time of bus b at station s , we have to alleviate the skewness of getting-on timestamps by logarithm transformation (i.e., function $\ln()$), where $\text{EXP}(x) = e^x$. For those positive-skewed timestamps, their logarithmic values roughly conform to the normal distribution. Due to the normality knowledge, we know 68% of g_i would not larger than $m_g + sd_g$. Therefore, t_s defined in line 12 is a sound approximation, because the start time of bus is larger getting-on timestamps of passengers in a probability more than a half.

Algorithm 1. Bus trips recognition

Input: records of smart card data in a bus ordered by the getting-on timestamp

Output: trips of the bus b at station s and the bus start time t_s in each trip

```

1  for a record  $r_i$ 
2     $b = r_i.b; s = r_i.s_n;$ 
3    if  $(r_i.t_n - r_{i-1}.t_n) < \gamma$ 
4      put  $r_i$  to the same trip with  $r_{i-1}$ ;
5    else
6      put  $r_i$  to a new trip;
7    end if
8  end for
9  for a trip of  $b$  at  $s$ 
10   for any  $r_i, g_i = \ln(r_i.t_n)$ ;
11   on those  $g_i$ , get their mean  $m_g$  and standard deviation  $sd_g$ ;
12    $b$ 's start time  $t_s = \text{EXP}(m_g + sd_g)$ 
13 end for

```

Through the Algorithm 1 above, the getting-on passenger volume analysis can be modeled as a two-step procedure in Fig. 3. Each step can be implemented as a Hadoop MapReduce job. Here, the vertical left part of either step works as a map task and the right one is a reduce task; each step requires only one-pass processing on the data.

The first step as the upper part of the Fig. 3 is to achieve the getting-on passenger volume of every single bus. Here, each record would be extracted by its attributes. The timestamp of getting-on is divided to *date* and *time*. After grouping by the composition of station id, bus id and date, Algorithm 1 is invoked. As a result, the passenger volume of each bus are achieved and ordered by the station id. For example, when θ is set as 1 h, and a output could be $\langle 3, 00028294, 20151208, 0, 0, 0, 0, 0, 0, 13, 0, 0, 25, 0, 0, 0, 18, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 \rangle$. It means the bus 0028294 at station 3 on Dec.8th 2015 had three trips when the counts were 13 in [6:00, 7:00), 25 in [9:00, 10:00) and 18 in [13:00, 14:00).

The second step as the lower part of Fig. 3 is to achieve the getting-on passenger volume of all the buses. In this step, the inputs are achieved from the first step. After grouping by the composition of station id, bus id and date, the final result is the vector addition in respective time slot. For example, when θ is set as 1 h, and an output could be $\langle 3, 20151208, 0, 0, 0, 0, 0, 64, 85, 105, 128, 256, 204, 230, 242, 189, 205, 143, 145, 252, 286, 259, 235, 102, 82, 35 \rangle$. It means the count for station 3 at each time slot of one hour on Dec.8th 2015.

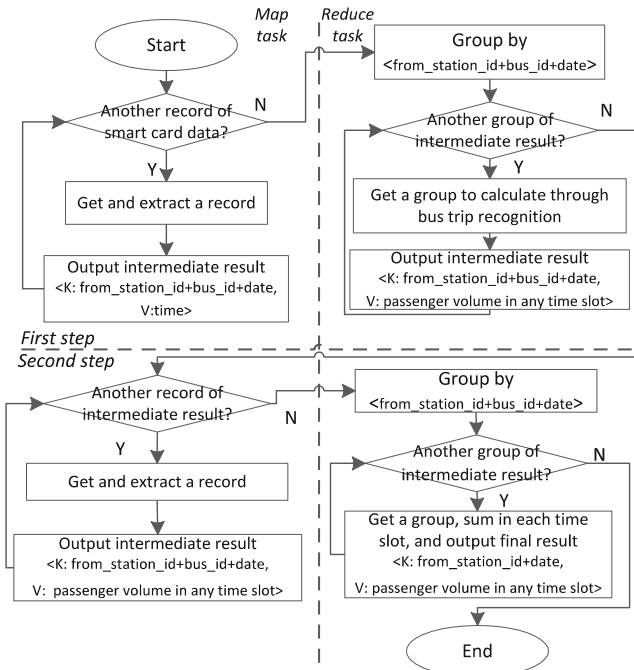


Fig. 3. Getting-on passenger volume analysis.

Here, considering some waiting periods of bus at a station overlap two adjacent time slots δ_i, δ_{i+1} , we regard the ridership belongs to either δ_i if $t_s \in \delta_i$ or to δ_{i+1} if $t_s \in \delta_{i+1}$. It is sound because the getting-on behaviors of passengers depend on the start of a bus.

Therefore, all three problems mentioned have been solved. Analogously, the **getting-off passenger volume** could be achieved, where the getting-off timestamp (i.e., r_i, t_f) of passengers and the stop time of buses are focused instead. With the similar logarithm transformation, the inferred bus stop time $t_e = \text{EXP}(m_g + sd_g)$ with the same confidence 68%. It is sound because the getting-off timestamps are also positive skewed and smaller than stop time in a probability more than a half.

3.3 Transfer Behavior and Its Passenger Volume Analysis

Then, the third type of passenger volume is discussed.

Definition 3. Transfer passenger volume. The transfer passenger volume presented as xPF_s^δ in any time slot $\delta(t_l, t_r)$ counts passengers P who gets off any bus b_j at s_f and gets on another bus b_k at s_n within a time duration no more than ε . Here, time $t_r > t_l, \exists$ two records r_f, r_n of smart card data, $r_f.p = r_n.p \in P, r_f.s_f = s_f, r_n.s_n = s_n, r_f.t_f \in \delta, r_n.t_n \in \delta$. It is achieved periodically with a fixed interval length $\theta = |\delta| = t_r - t_l$. Usually, θ is set as 30 min, 1 h or 2 h in practice, and the threshold $0 < \varepsilon < \theta$. The station s_f and s_n are either geographical neighbors or the same one.

Referring a location pair (i.e., getting-off and getting-on stations) at given time duration (i.e., time slots), the transfer passenger volume faces analogous condition as that of getting-on passenger volume. Because transfer behavior consists of a getting-off behavior and a successive getting-on one like Fig. 4, it could be achieved directly from the method in Sect. 3.2. However, it would be inefficient to merge two independent analyses for integral results. Accordingly, we focus on the dedicated method in this section.

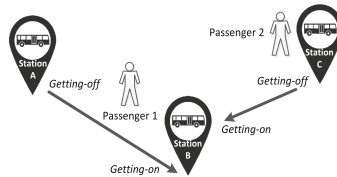


Fig. 4. The transfer behaviors of passengers.

Algorithm 2. Transfer behaviors recognition and counting

Input: records of smart card data group by passenger

Output: the transfer behavior of each passenger

```

1  for two successive records  $r_{i-1}, r_i$  of passenger  $p$ 
2   $r_{i-1}.p = r_i.p = p; b_f = r_{i-1}.b; b_n = r_i.b; s_f = r_{i-1}.s_f; s_n = r_i.s_n; t_f = r_{i-1}.t_f; t_n = r_i.t_n;$ 
3  if ( $b_f \neq b_n$ )
4    if ( $\text{distance}(s_f, s_n) < \eta$ ) and ( $t_n - t_f \leq \Gamma$ )
5      a transfer behavior of  $p$  appears at station  $s_n$  and at the time  $t_n$ ;
6    end if
7  end if
8  end for
    
```

Besides the same problems with the getting-on volume analysis, the transfer passenger volume analysis remains other two ones. One is how to define the spatial neighborhood of any station. The transfer behavior only makes sense at the getting-on station. The other is how to depict temporal closeness of any passenger for a transfer between his getting-off behavior and the successive getting-on one. Too long delay would not appropriately reflect the passengers' real intention.

To solve those problems, we propose the Algorithm 2 with the symbols in definition 1. The algorithm to find a transfer behavior is based the definition of spatial neighborhood and temporal closeness in the line 3–5. For a transfer behavior of a passenger, the getting-off bus must be different with the getting-on one; while the getting-off station could be same with the getting-on one. Threshold η of spatial neighborhood restricts the max cartographic distance between two stations, and threshold Γ of temporal closeness implies the longest period duration two behaviors. Here, we regard the count of transfer behavior as the result at the time t_n with the station s_n .

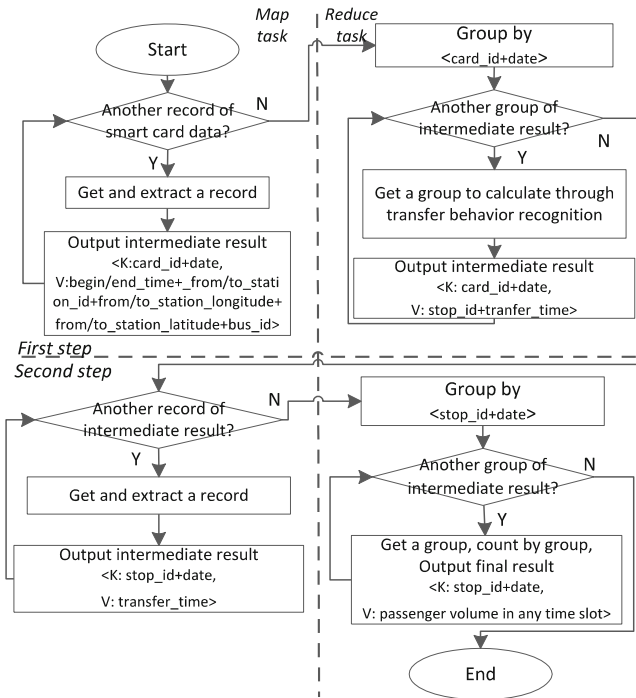


Fig. 5. Transfer passenger volume analysis.

Through the Algorithm 2, the transfer passenger volume analysis can be modeled as two-step procedure in Fig. 5. Each step can be implemented as a Hadoop MapReduce job. Here, the vertical left part of either step works as a map task and the right one is a reduce task; each of which requires only one-pass processing on the data.

The first step as the upper part of the Fig. 5 is to find the transfer behaviors of passengers. Here, each record would be extracted by its attributes. The timestamp of either getting-on and or getting-off is divided to *date* and *time*. The date of getting-on and getting-off are identical. After grouping by the composition of card id and date, the algorithm 2 is i. The thresholds could be set empirically: $\eta = 1000$ m and $\Gamma = 20$ min. It comes from the facts in the official documents: in Beijing for a bus transfer, 95% passengers expect to wait less than 20 min; only 16% passengers would endure to walk

more than 1000 m to the next getting-on station. As a result, if θ is 1 h, an output could be $\langle 3, 00000370A80456014EBE774FE6D150C1, 20151208, 8, 1 \rangle$. It means a passenger using that card had a transfer behavior at station 3 at the 8th time slot, i.e., [7:00, 8:00) on Dec.8th 2015.

The second step as the lower part of Fig. 5 is to achieve the transfer passenger volumes of all the passengers. In this step, the inputs are achieved from the first step. After grouping by the composition of station id and date, the final result is the vector addition in respective time slots. For example, if θ is set as 1 h, a output could be $\langle 3, 20151208, 0, 0, 0, 0, 0, 26, 52, 75, 82, 126, 124, 123, 130, 98, 105, 93, 75, 82, 96, 89, 75, 82, 68, 23 \rangle$. It shows the transfer passenger volume at station 3 in 24 time slots on Dec.8th 2015.

4 Evaluations

4.1 Settings

The executive performance and practical effects are evaluated respectively by extensive experiments and case studies in this section.

In our private Cloud as the service infrastructure layer, four Acer AR580 F2 rack servers via Citrix XenServer 6.2 are utilized in the infrastructure layer, each of which own 8 processors (Intel Xeon E5-4607 2.20 GHz), 64 GB RAM and 80 TB storage. Six virtual machines are used to build our platform service, each of which owns 4 cores CPU, 4 GB RAM and 1.2 TB storage with CentOS 6.6 x86_64 operating system. As the Fig. 1, the data layer consists of MySQL 5.1 and HDFS; the processing layer is the customized Hadoop MapReduce 2.6.0; the application layer would be further exploited in the Sect. 4.3.

We employ the smart card data of Beijing on eight days in 2013, which contains 24263142 records on 7349 buses of 233 lines involving 3581 stations. All the data was generated from the readers charging by travel distance, and each record contains 13 attributes as Table 1. The data has been cleaned in advance through dedicated method [14], and has been divided into eight parts by the original dates as the experiment inputs.

To analyze the passenger volume, two different methods have been implemented in our platform service for comparison. One is our method termed as BD (Big Data). The counterpart is a statistic estimation [11] termed as ODE (Origin-destination Estimation) in the current productive environment.

4.2 Experiments

We evaluate the performance of two methods to analyze passenger volume in the experiments below.

Experiment 1. The data of different size is used as the input. The getting-on and transfer passenger volume are executed through both BD and ODE, and note their average executive times in each condition. The result is showed as Fig. 6 where the left is the getting-on passenger volume analysis and the right is the transfer one.

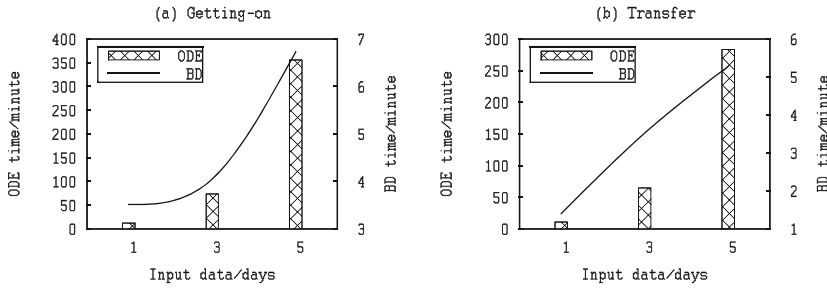


Fig. 6. The passenger volume analyses through two methods

When the input size increases, the executive time of both methods rises, but ODE has much longer time than BD in two orders of magnitude for either analysis. In average, the input record size of one day is about 1 million. The executive time through ODE grows sharply when the input is more than 5 million, while that of BD rises almost linearly. On the input of 3-day data, BD costs minute-level time, while the ODE requires more than 5 h. The lower latency of BD comes from the parallel execution of two-step procedure in either passenger volume analysis. But through ODE, the analysis requires multiple passes to sort data, and has to run on a single machine without parallelism. As a result, ODE only suits small size data, while BD has much lower latencies on massive data.

In the following parts, only BD is evaluated for its efficiency and scalability.

Experiment 2. The data of one day is appended to the input in each test, and the executive times for getting-on passenger volume analysis through BD are noted. For the comparison on each input size, the interval length θ is set as 10-min, 1-h and 4-h respectively. The result is presented in Fig. 7(a). The average executive time on fixed one million records in each test can be deduced as Fig. 7(b).

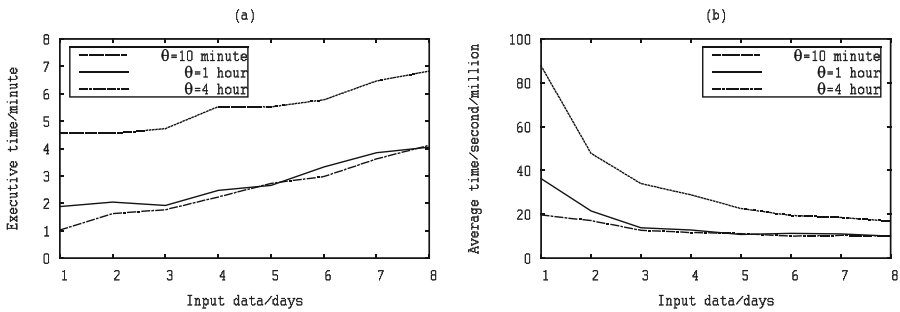


Fig. 7. The getting-on passenger volume analyses at 3 intervals on 8 inputs.

The getting-on analysis through BD method is proved scalable on the increasing data. On the one hand, when input scales at any interval length, the increment of executive time surpasses the linearity. In Fig. 7(a), the time is kept minute-level and not

doubled even when the input size grows eight folds. That trend can be demonstrated clearly in another perspective of Fig. 7(b), where the average executive time on fixed input size declines to the steadiness about 10 s. It shows that the processing capacity of BD method is stable and horizontally scalable. On the other hand, on the input of the same size, the executive time varies by interval lengths. The longer interval length implies lower latencies, because our analysis relies on the interval length: shorter interval length implies more dimensions in result vector due to definition 2, and requires more calculation delay. It is interesting that when input scales, the capacity on fixed size converges at any interval length like Fig. 7(b), which also proves its horizontal scalability.

In a similar way, the transfer passenger volume analysis is evaluated next.

Experiment 3. The data of one day is appended to the input in each test, and the executive times for transfer passenger volume analysis through BD method are noted. For the comparison on each input size, the interval length θ is set as 10-min, 1-h and 4-h respectively. The result is presented in Fig. 8(a). The average executive time on fixed one million records in each test can be deduced as Fig. 8(b).

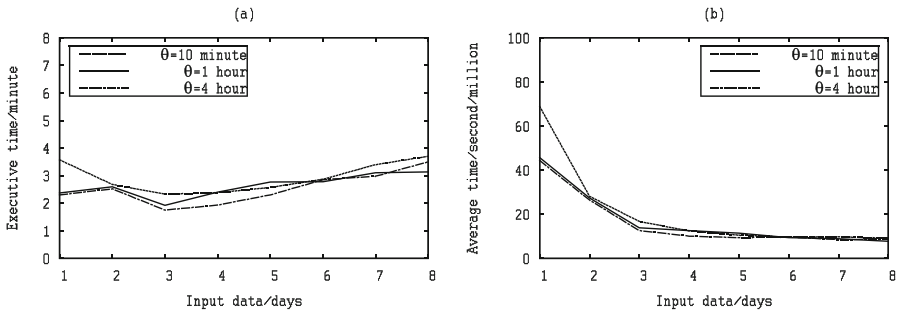


Fig. 8. The transfer passenger volume analyses at 3 intervals on 8 inputs.

The transfer analysis through BD method is also proved scalable on the increasing data. On the one hand, when input scales at any interval length setting, the executive time of the analysis keeps steadily. In Fig. 8(a), that time fluctuates between one and four minutes even when the input grows eight folds. Compared with that of getting-on analysis, the execution here is relatively faster, because transfer behavior appears much fewer than getting-on and requires lesser time. As Fig. 8(b), the executive time on fixed input size declines to a steadiness about less than 10 s. The same value with that of getting-on analysis shows the scalable capacity of BD method again. On the other hand, on the input of the same size, the executive time has little difference at three interval lengths setting because the intermediate results in the experiment are too small to manifest their variance. The capacity on fixed input size also converges at any interval length when input scales as Fig. 8(b), which proves its scalability either.

With the three experiments above, our analysis in platform service proves minute-level latencies on weekly historical data with horizontal scalability.

4.3 Case Studies

We evaluate the practical effects in the application layer of service by case studies next. The getting-on passenger volume is exhibited first.

Case 1. The jobs of getting-on passenger volume are submitted successively to *web console* to execute. After their completion, the visual results are available in *map application*. As an example, when a station named *Beijing West Railway Station* is selected in the map, the profile emerges including id, name and GPS location, and the hyperlink of getting-on passenger volume is also enabled in a pop-up window.

The analysis jobs are governed full life-cycle in our service via the *web console*. In this case as the Fig. 9(a), the second job of the getting-on passenger volume analysis is being submitted to the console. After the code package has been uploaded, the console would check source code to assist the job configuration (e.g., the executive entrance from the candidate Java main classes is auto-prompting), and then assigns to the processing layer. For any station, the passenger volume would be visualized in the *map application*. As In this case as the Fig. 9(b), the station *Beijing West Railway Station* is a transportation hub and close to a railway station. Some buses are in 7 * 24 h service around there, and most bus passengers go there to take trains. The getting-on passenger volume of this station would display in the map at all the time slots in a given day at one hour interval by default.

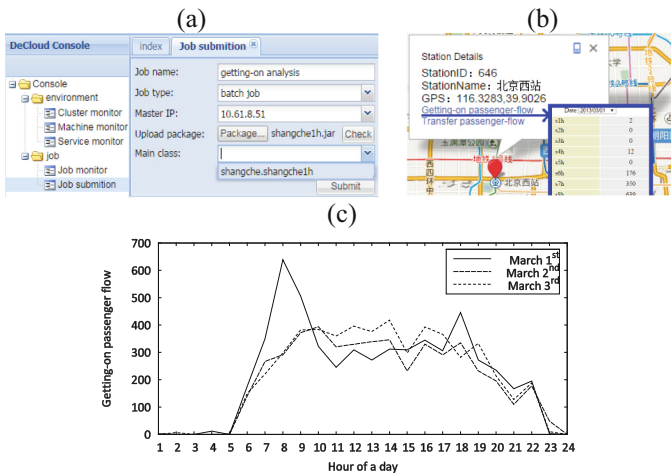


Fig. 9. The getting-on passenger volume of station Beijing West Railway Station.

Moreover, the synthetic views can be generated in our platform service. In this case, such a view as the Fig. 9(c) demonstrates the comparison of getting-on passenger volume on three successive days. We found the result on any of three days keeps steadily high since 8:00 to 19:00 just when the trains are usually busy. All those exactly match the real situations from the local official statistics.

Next, the effects of transfer passenger volume are evaluated in another case.

Case 2. The jobs of transfer passenger volume analysis are submitted successively to *web console*. After their completion, the visual results are available in *map application*. As an example, when a station named *Chinese Academy of Agricultural Science* is selected in the map, the profile emerges, and the hyperlink of transfer passenger volume is also enabled in a pop-up window.

The jobs' status is available in JSON (JavaScript Object Notation) format in our service and also visualized in *web console*. In this case as Fig. 10(a), the newly submitted jobs are noted as successfully completed in a history table. Even during the jobs' run-time, the console provides graphical interface to suspend or kill them. In this case as the Fig. 10(b), the station *Chinese Academy of Agricultural Science* lies in a junction of three trunk roads where more than 12 bus lines pass-by around two adjacent stations. It also can reach to two subway stations within 800 m. Therefore, many bus passengers go there for a transfer. The transfer passenger volume would display in the map at all the time slots in a day at one hour interval by default.

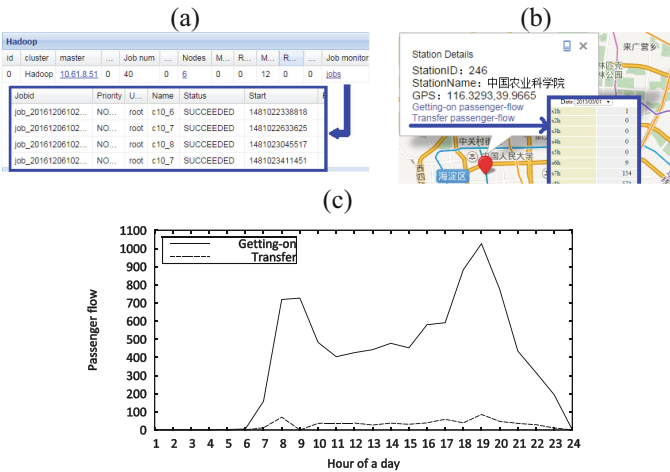


Fig. 10. Two kinds of passenger volume of station Chinese Academy of Agricultural Science.

Moreover, a synthetic view as the Fig. 10(c) is generated in the platform service to compare two kinds of passenger volume on a given date March 4th. We found the value of transfer volume is about 1/8-1/5 than that of getting-on and owns the same rush hours. It reflects the fact that transfer behavior is much smaller than getting-on one but shares the similar temporal trends. In either type of passenger volume in this workday, we can find the morning rush hours are from 7:00 to 9:00 and evening ones are from 17:00 to 20:00. All those exactly match the traffic report published by Beijing Traffic Commission.

With the two case studies above, our method achieves convenient effects and effective results in practical situations.

5 Conclusion

In public transportation domain, we propose a novel platform service to analyze multiple passenger volumes on massive smart card data. For any analysis, our service can hold minute-level latency on historical weekly data and keep nearly linear scalability in extensive conditions. It also shows practical effects and exact results in authentic cases. In the future, we would introduce more domain analyses in our service, such as bus arrive time prediction at given station and bus transit speeds prediction between two directly adjacent stations.

Acknowledgements. This work was supported by the Youth Program of National Natural Science Foundation of China under Grant 61702014, the General Program of Beijing Natural Science Foundation under Grant 4192020, and Top Young Innovative Talents of North China University of Technology under Grant XN018022.

References

1. Zhang, J., Zheng, Y., Qi, D., Li, R., Yi, X., Li, T.: Predicting citywide crowd flows using deep spatio-temporal residual networks. *Artif. Intell.* **259**, 147–166 (2018)
2. Chen, M., Mao, S., Liu, Y.: Big data: a survey. *Mob. Netw. Appl.* **19**, 171–209 (2014)
3. Xiong, G., et al.: A kind of novel ITS based on space-air-ground big-data. *IEEE Intell. Transp. Syst. Mag.* **8**, 10–22 (2016)
4. Ma, X., Wu, Y.-J., Wang, Y., Chen, F., Liu, J.: Mining smart card data for transit riders' travel patterns. *Transp. Res. Part C: Emerg. Technol.* **36**, 1–12 (2013)
5. Tang, N.: Big data cleaning. In: Chen, L., Jia, Y., Sellis, T., Liu, G. (eds.) *APWeb 2014. LNCS*, vol. 8709, pp. 13–24. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11116-2_2
6. Long, Y., Zhang, Y., Cui, C.: Identifying commuting pattern of Beijing using bus smart card data (in Chinese). *Acta Geogr. Sin.* **67**, 1339–1352 (2012)
7. Zhang, C., Song, R., Sun, Y.: Kalman filter-based short-term passenger flow forecasting on bus stop (in Chinese). *J. Transp. Syst. Eng. Inf. Technol.* **11**, 154–159 (2011)
8. Zhou, J., Sun, Y., He, L.: Multi-model hybrid traffic flow forecast algorithm based on multivariate data. In: Sun, Y., Lu, T., Xie, X., Gao, L., Fan, H. (eds.) *ChineseCSCW 2018. CCIS*, vol. 917, pp. 188–200. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-3044-5_14
9. Dugane, R.A., Raut, A.: A survey on big data in real time. *Int. J. Recent Innov. Trends Comput. Commun.* **2**, 794–797 (2014)
10. Tao, S., Rohde, D., Corcoran, J.: Examining the spatial–temporal dynamics of bus passenger travel behaviour using smart card data and the flow-comap. *J. Transp. Geogr.* **41**, 21–36 (2014)
11. Zhou, X., Yang, X., Wu, X.: Origin-destination matrix estimation method of public transportation flow based on data from bus integrated-circuit cards (in Chinese). *J. Tongji Univ. (Nat. Sci.)* **40**, 1027–1030 (2012)
12. Zhang, J., Yu, X., Tian, C., Zhang, F., Tu, L., Xu, C.: Analyzing passenger density for public bus: inference of crowdedness and evaluation of scheduling choices. In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC 2014)*, pp. 2015–2022. IEEE, (Year)

13. Carey, M.J., Jacobs, S., Tsotras, V.J.: Breaking BAD: a data serving vision for big active data. In: Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems, pp. 181–186. ACM, Irvine (2016)
14. Ding, W., Cao, Y.: A data cleaning method on massive spatio-temporal data. In: Wang, G., Han, Y., Martínez Pérez, G. (eds.) Advances in Services Computing: 10th Asia-Pacific Services Computing Conference, APSCC 2016, Proceedings, pp. 173–182. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-49178-3_13
15. Pelletier, M.-P., Trépanier, M., Morency, C.: Smart card data use in public transit: a literature review. *Transp. Res. Part C: Emerg. Technol.* **19**, 557–568 (2011)
16. Zhang, D., Zhao, J., Zhang, F., He, T.: UrbanCPS: a cyber-physical system based on multi-source big infrastructure data for heterogeneous model integration. In: Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems, pp. 238–247. ACM, Seattle (2015)
17. Zhang, D., Zhao, J., Zhang, F., He, T.: coMobile: real-time human mobility modeling at urban scale using multi-view learning. In: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 1–10. ACM, Bellevue (2015)
18. Wang, Y., Ram, S., Currim, F., Dantas, E., Saboia, L.A.: A big data approach for smart transportation management on bus network. In: 2016 IEEE International Smart Cities Conference (ISC2), pp. 1–6. IEEE (2016)
19. Ram, S., Wang, Y., Currim, F., Dong, F., Dantas, E., Saboia, L.A.: SMARTBUS: a web application for smart urban mobility and transportation. In: Proceedings of the 25th International Conference Companion on World Wide Web, pp. 363–368. International World Wide Web Conferences Steering Committee, Montreal (2016)