Distributed Big Data analysis for mobility estimation in Intelligent Transportation Systems

Enzo Fabbiani[1], Pablo Vidal[2], Renzo Massobrio[1], and Sergio Nesmachnow[1]

[1] Universidad de la República, Uruguay
{enzo.fabbiani, renzom, sergion}@fing.edu.uy
[2] CONICET – Universidad Nacional de la Patagonia Austral,
Caleta Olivia, Argentina
pjvidal@uaco.unpa.edu.ar

Abstract. This article describes the application of distributed computing techniques for the analysis of big data information from Intelligent Transportation Systems. Extracting useful mobility information from large volumes of data is crucial to improve decision-making processes in smart cities. We study the problem of estimating demand and origin-destination matrices based on ticket sales and location of buses in the city. We introduce a framework for mobility analysis in smart cities, including two algorithms for the efficient processing of large mobility data from the public transportation in Montevideo, Uruguay. Parallel versions are proposed for distributed memory (e.g., cluster, grid, cloud) infrastructures and a cluster implementation is presented. The experimental analysis performed using realistic datasets demonstrates that significant speedup values, up to 16.41, are obtained.

Keywords: Distributed computing; Big data; mobility analysis; Intelligent Transportation Systems

1 Introduction

Nowadays, many complex activities are developed in modern cities, which impose serious challenges to the mobility of citizens [7]. The main mobility issues in dense urban areas are related to public transport systems that are not capable of fulfilling the growing demand for transportation. In order to implement innovative solutions that address this issue, it is necessary to have access to updated information about the mobility of the citizens [2]. In most cities, the information available from public administrations is scarce and outdated, due to the lack of financial and human resources assigned to gathering and managing mobility data. In other cases, data are gathered but are not used for improving the mobility or optimizing public/vehicular transportation. In this scenario, developing improved decision-making processes related to urban mobility becomes mandatory. New smart city technologies are very helpful to offer high quality solutions for this kind of mobility problems.

The paradigm of smart cities proposes taking advantage of information and communication technologies to improve the quality and efficiency of urban services [4]. Intelligent Transportation Systems (ITS) are a key component of smart cities. ITS are defined as those systems integrating synergistic technologies, computational intelligence, and engineering concepts to develop and improve transportation. They are aimed at providing innovative services for transport and traffic management, with the main goals of improving transportation safety and mobility, and also enhancing productivity [18]. ITS allow gathering several data regarding transportation and mobility in the cities [5]. In big urban areas, they generate huge volumes of data that can be processed to extract valuable information about the mobility of citizens.

This article presents a framework to study mobility in the context of smart cities. In order to solve mobility and urban transportation optimization problems it is imperative to understand the mobility patterns of citizens and the demand distribution for public transport. This information is often represented using matrices: *i*) origin-destination (OD) matrices, indicate the amount of people moving from each point in the city in a given period of time [1]; *ii*) demand matrices, measure the number of bus tickets sold between each location in the city. These matrices are often built using data from surveys performed in-situ to passengers and drivers. However, surveys offer only a partial vision of the mobility patterns in the city, are expensive, and need to be performed regularly to get updated information. A different approach to build demand and OD matrices is based on processing real data from ITS, including tickets sold with and without smart cards, GPS data from buses, etc.

We introduce a specific methodology for generating OD and demand matrices using data from ITS to study mobility in smart cities. Taking into account the large computing time demanded when dealing with huge volumes of data, we propose applying distributed computing techniques to speed up the processing. A data-parallel approach is devised to process large datasets on distributed memory parallel architectures (e.g., cluster, grid, and cloud systems). Two specific algorithms are presented, based on real data from the ITS in Montevideo, Uruguay. The proposed approach can easily be extended to any other ITS-related information and scenarios.

The main contributions of the research reported in this article are: *i*) we introduce a methodology for OD matrix estimation using smart card information; *ii*) we design and implement specific algorithms for processing big data using distributed computing techniques; and *iii*) we report an experimental analysis which demonstrates that significant speedup values are obtained, allowing the efficient processing of large datasets.

The article is organized as follows. Section 2 introduces the problem of Big Data analysis for ITS. The proposal of applying distributed computing techniques to accelerate the processing of large volumes of ITS data is described in Section 3. The experimental evaluation of the proposed algorithms is reported in Section 5. Finally, Section 6 presents the conclusions of the research and formulates the main lines for future work.

2 Big Data analysis for Intelligent Transportation Systems

This section describes the problem of Big Data analysis for building demand and origin-destination matrices. A review of related work is also presented.

2.1 Problem description

The main challenge faced when generating demand and OD matrices using data from tickets sales is that passengers validate their smart cards when they board but not when they alight the bus. Therefore, while the origin of each trip is known with certainty, it is necessary to estimate the destination. Furthermore, some passengers do not use smart cards to pay for their ticket. Therefore, there are sale records which do not provide information to be used to track several trips made by a single passenger. Specific big data processing algorithms must be designed and implemented for each case.

In this article we focus on generating demand and OD matrices for the city of Montevideo, Uruguay. The city government in Montevideo introduced in 2010 an urban mobility plan to redesign and modernize urban transport in the city [9]. Under this plan, the Metropolitan Transport System (Sistema de Transporte Metropolitano, STM) was created, with the goal of integrating the different components of the public transportation system together. One of the first improvements in STM was to include GPS devices on buses and allow passengers to pay for tickets using a smart card (STM card). Additionally, the complex system of fares was simplified to allow only two different type of tickets: i) "one hour" tickets, allowing up to 1 transfer within an hour of taking the first bus; ii) "two hours" tickets, allowing unlimited transfers within 2 hours from the moment the ticket is purchased. However, it is not compulsory to use the STM card to buy bus tickets. Passengers may pay with cash directly to the driver. In this case, the ticket is only valid for that trip and no transfers are allowed.

The bus companies that operate in Montevideo are obliged to send bus location and ticket sale data daily to the city authorities. The bus network in Montevideo is quite complex, consisting of 1383 bus lines and 4718 bus stops. In this article we consider the complete dataset of ticket sales and bus location for 2015, comprising nearly 200 GB of data. Bus location data contain information about the position of each bus, sampled every 10 to 30 seconds. Each location record holds the following information:

- *lineID*, the unique bus line identifier;
- *tripID*, the unique trip identifier for each single trip for a given lineID;
- *latitude* and *longitude*,
- *vehicle speed*;
- *timestamp* of the location; and
- *stopID*, the identifier for the nearest bus stop to the current bus location.

Ticket sales data contain information about sales made with and without STM cards. Each sale record has the following fields:

– *tripID*, as in location data;
– *latitude* and *longitude*,
– *stopID*, as in location data;
– *number of passengers*, since it is possible to buy tickets for multiple passengers at once; and
– *timestamp* of the sale.

Additionally, tickets payed with STM cards have the following fields: unique STM card identifier (cardID) which is hashed for privacy purposes, number of transactions made with that STM card (transactionID), and the last payed transaction (payedID). This allows identifying when a passenger transfers between buses, since transactionID will increment while payedID will remain unchanged. The number of transfers is equal to transactionID−payedID.

2.2 Related work

Many articles in the related literature have proposed applying statistical analysis for estimating OD matrices and computing several other relevant statistics for ITS. Some approaches applying parallel and distributed computing techniques have also been proposed recently. A review of the main related works is presented next.

A detailed literature review on the use of smart cards in ITS was presented by Pelletier et al. [14]. The review covers the hardware and software needed for the deployment of smart card payment solutions in urban transportation systems as well as the privacy and legal issues that arise when dealing with smart card data. Additionally, the authors identify the main uses for these data, including: long-term planning, service adjustments and performance indicators of the transportation systems. Finally, examples of smart card data usage around the world are reviewed.

Trépanier et al. [20] presented a model to estimate the destination for passengers boarding buses with smart cards, following a database programming approach. Two hypotheses are considered, which are commonly used in many related works: *i*) the origin of a new trip is the destination of the previous one; *ii*) at the end of the day users return to the origin of their first trip of the day. Based on these assumptions, the authors propose a method to follow the chain of trips of each user in the system. Those trips for which chaining is not possible (e.g., only one trip in the day for a particular user) are compared with all other trips of the month for the same user, in order to find similar trips with known destination. The experimental evaluation was conducted using real information from a transit authority in Gatineau, Quebec. Two datasets were used, with 378,260 trips from July 2003 and 771,239 trips from October 2003. Results showed that a destination estimation was possible for 66% of the trips. Most trips where destination could not be estimated take place during off-peak hours, where more atypical and non-regular trips are performed. Considering only peak hours, the percentage of trips with their destination estimated improves to 80%. However, the estimation accuracy could not be assessed due to lack of a second source of data (e.g., automatic passenger count) for comparison.

Wang et al. [21] proposed using a trip-chaining method to infer bus passenger OD from smart card transactions and Automatic Vehicle Location (AVL) data from London, United Kingdom. In the studied scenario, authors needed to estimate both origin and destination of trips. Origins are accurately estimated by searching for the timestamp of each smart card transaction in the AVL records to determine the bus stop of each trip. To estimate destinations, the authors used a similar methodology to that presented by Trépanier et al. [20], chaining trips when possible to infer destinations. Results were compared against passenger survey data from Transport for London, performed every five to seven years for each bus route and including the number of people boarding and alighting at each bus stop. The analysis show that origins can be estimated for more than 90% of the trips while origin and destinations can be estimated for 57% of all trips. When compared to the survey data, the difference on the estimated destinations were below 4%. Finally, two practical applications of the results are presented. The first one consists of studying the daily load/flow variation to identify locations along each bus route where passenger load is high, as well as underutilized route segments. The second application consists of an transfer time analysis, evaluating the average time that users need to wait for transferring between buses, based on the alighting stop and the AVL data.

Munizaga et al. [12] presented a similar approach to estimate OD matrices in the multimodal transportation system of Santiago, Chile, where passengers can use their smart cards to pay for tickets at metros, buses or bus stations.

Several proposals have applied distributed computing approaches to process large volumes of traffic data, but few works have dealt with the estimation of demand or OD matrices.

Pioneering works on this topic applied distributing computing to gather traffic data. Sun [17] proposed a client-server model developed in CORBA for collecting traffic counts in real time, to be used for dynamic origin/destination demand estimation. The proposed solution included a CORBA client to extract data from the traffic network, and a CORBA server for storing data in a centralized repository. All the information is processed to be later used in Dynamic Traffic Assignment strategies for the traffic network studied, for the estimation of dynamic OD matrices applying a bi-level optimization framework.

Toole et al. [19] propose combining data from many sources (call records from mobile phones, census, and surveys) to infer OD matrices. The authors combine several existing algorithms to generate OD matrices, assign trips to specific routes, and to compute quality metrics on road usage. Furthermore, a web application is introduced to give simple visualizations of the computed information. The authors mention that computations are performed in parallel, but no parallel model is described and no performance metrics are reported.

Also using mobile phone data, Mellegård [11] proposed a Hadoop implementation to generate OD matrices while keeping users' privacy. However, the experimental analysis is done on synthetic data due to the difficulties on getting real data from mobile operators. Furthermore, no performance metrics are reported, so the advantages of the Hadoop implementation are not clear.

Huang et al. [8] proposed a methodology for offline/online calibration of Dynamic Traffic Assignment systems via distributed gradient calculations. An adaptive network decomposition framework is introduced for parallel computation of traffic network metrics and for parallel simulation, in order to accelerate the computations. Parallel OD demand estimation is proposed as a line for future work, in order to deal with large-scale traffic networks with huge number of OD pairs and sensors.

Our recent work [10] presented a preliminary analysis on using distributed computing techniques to process GPS data from buses. We introduced a Map-Reduce approach for processing historical data to study relevant metrics to assess the quality-of-service of the transportation systems in Montevideo, Uruguay. We used the strategy to compute the average speed of buses and to identify troublesome locations, according to the delay and deviation of the times to reach each bus stop. The parallel implementation scaled properly when processing large volumes of input data.

In the present article, we extend our preliminary approach [10] to solve a more complex and computing intensive problem: the estimation of demand and OD matrices for public transportation. Up to our knowledge, this approach has not been previously proposed in the related literature.

3 Mobility estimation from ITS smart card data

This section presents the proposed methodology for estimating demand and OD matrices taking into account the two kind of transfer trips in Montevideo (explained in Section 2). We introduce two models for estimation: the first one for one-way transfer trips and the second one for multiple trips. We present a description of our sequential algorithm for estimating demand and OD matrices. Finally, the parallel implementation with all its components is described.

3.1 Models for demand and OD estimation

The model used for estimating OD matrices is based on reconstructing the trip sequence for passengers that use a smart card, following a similar approach to that applied in the related literature [20, 21, 12]. We assume that each smart card corresponds to a single passenger, so we use the terms *card* and *user* in an indistinct manner. The proposed approach is based on processing each trip, retrieving the bus stop where the trip started, and identifying/estimating the stop where the passenger alighted the bus from the information available.

We identify two different ways of estimating destinations from the data used: *transfer trips* and *direct trips*. The main details for each case are presented next.

Transfer trips. In a transfer trip, passengers pay for their ticket when boarding the first bus by using a smart card identified by its cardID. Later, they can take one or more buses within the time limits permitted by the ticket, as explained in Section 2.1. For each ticket sold, the number of transactions (transactionID)

made with that cardID and the last payed transaction (payedID) are recorded. This allows detecting whether a smart card record corresponds to a new trip (payedID is equal to transactionID) or to a transfer between buses (transactionID is higher than payedID). We assume that passengers avoid excessive walking in transfers; we consider that a passenger finishes its first leg at the nearest bus stop to the bus stop where he boards the second leg, and so on. The boarding bus stop for the second leg is recorded in the system, thus we estimate the alighting point from the first bus by looking for the closest bus stop corresponding to that line.

A transfer example is presented in Figure 1. At 07:42 the passenger takes bus number 1 (green line) at bus stop 12. At this point, we have no information about the destination of the passenger. However, at 08:12 the passenger boards bus number 2 (blue line) by using a transfer, without paying a new ticket. Therefore, we can confidently estimate that the passenger alighted from bus number 1 at bus stop number 1-5 which is the closest bus stop to bus stop 23, where the passenger does the transfer. We have no information about the destination of the second trip. This issue is addressed with the direct trip estimation, described next.

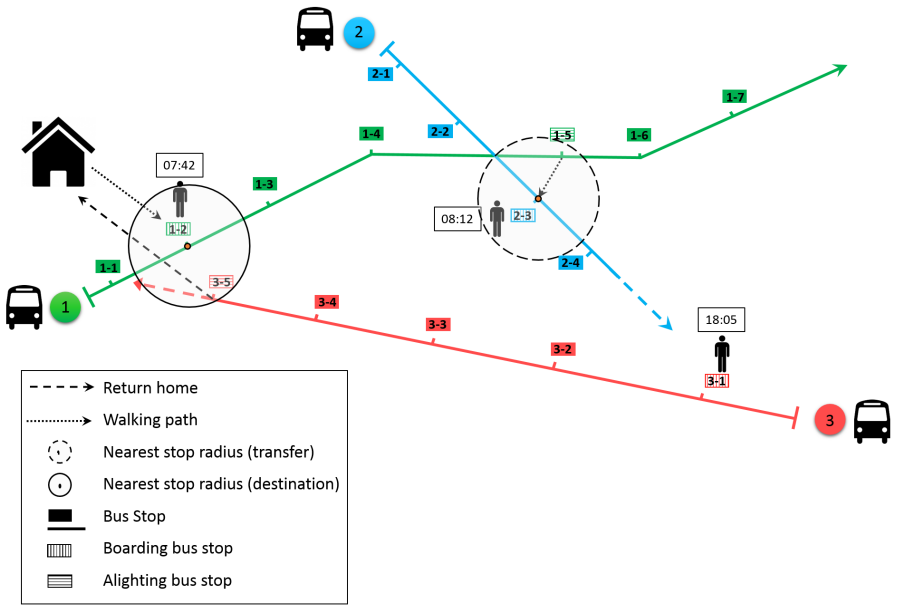

Fig. 1. Demand and OD estimation for transfer and direct trips

Direct trips. We consider direct trips as those that have no bus transfers. We also consider the last leg of a trip with one or more transfers as a direct trip. In both cases, the difficulty lies in accurately estimating a destination point for these trips.

To estimate the destination points we consider two assumptions, which are commonly used in the related literature: *i*) passengers start a new trip at a bus stop which is close to the destination of their previous trip; *ii*) at the end of the day, passengers return to the bus stop where they boarded the first trip on the same day.

In order to estimate destinations it is necessary to chain the trips made by each passenger on a single day. A preliminary study performed on the sales dataset showed that the best option is to consider each day starting at 04:00, since the lowest number of tickets are sold at that time. This allows considering passengers with different travel patterns, such as those who commute to work during the day and those who work at night.

The model for chaining direct trips of a specific passenger works as follows. We iterate through all the trips done in a 24 hour period (from 04:00 to 04:00 on the following day). For each new trip, we try to estimate the alighting point by looking for a bus stop located in a predefined range from the boarding bus stop of the previous trip.

In the example shown in Fig. 1, the passenger takes bus number 3 (red line) at 18:05, to return home. In order to estimate the destination of this trip, we look for the closest bus stop of bus number 3 located within a given search radius from stop number 12, which is the origin of the previous trip. In the example, stop number 35 is the only stop within that radius, so it is chosen as the alighting point for the trip. When no bus stop is found on that radius, the procedure is repeated using a larger radius (two times the original one) to search for bus stops. If no bus stop is found using the larger radius, the origin of the trip is recorded, in order to report the number of unassigned destinations.

3.2 Algorithm for demand and OD estimation

We propose a specific methodology for reconstructing the trip sequence for passengers, by estimating the destination points from the information available. The algorithm for estimating trip destinations is described in the flowchart in Fig. 2.

Three phases are identified in the proposed algorithm, which are relevant for building the estimated demand and OD matrices:

Pre-processing. The pre-processing phase prepares the data, filtering those records with incoherent information and classifying records by month per passenger. The algorithm receives as input an unstructured dataset containing raw GPS positions and ticket sales data. Initially, the algorithm discards those sales records that have invalid GPS coordinates; which are not processed for the demand and OD matrices estimation. A sale record has an invalid location when its coordinates are not within the route of the bus corresponding to the sale, with a tolerance of 50 meters.

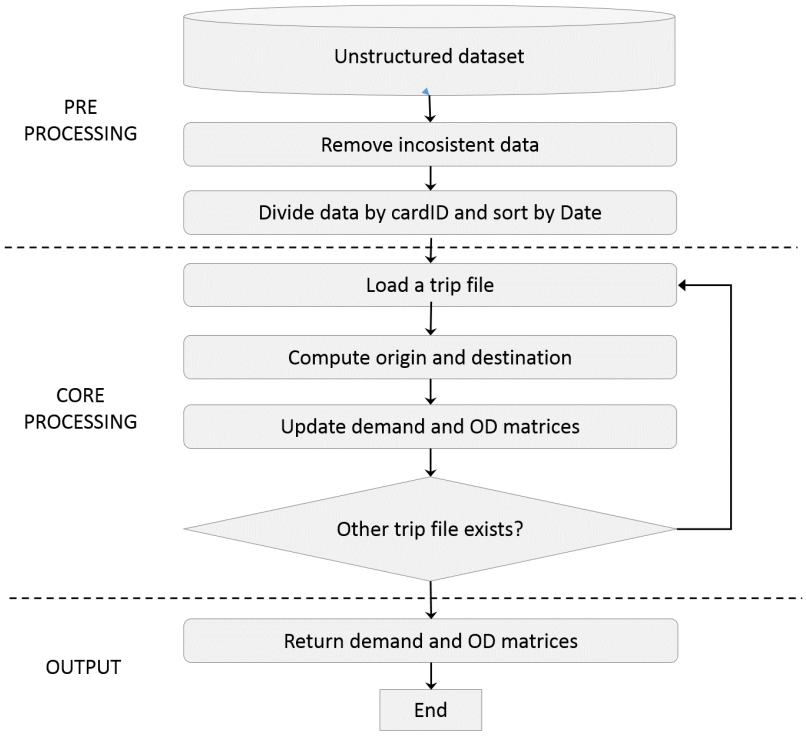

Fig. 2. Flowchart of the algorithm for estimating trip destinations.

Finally, trip records with consistent location information are separated into different files, according to their cardID and then ordered according to their date field. This allows processing the trips of each passenger independently.

Core processing. In this phase the sales data are processed in order to generate demand and OD matrices. Data are iteratively processed: for each passenger, trips are analyzed considering 24 hour periods starting and finishing at 04.00. First, the origin of the trip is recorded. In order to estimate the destination, the models defined in Section 3.1 are applied, depending on whether the trip corresponds to a transfer or to a direct trip. Once the origin and the destination are computed, the corresponding values are updated in the demand and OD matrices. The process is repeated until all trip records are processed. In our study, we consider a distance of 500m for the search radius used when estimating destination of direct trips, as described in Section 3.1.

Output After all records are computed the demand and OD matrices are returned.

4 A parallel algorithm for demand and OD matrices estimation

The capability of a traditional sequential algorithm for the estimation of demand and OD matrices is limited by the computational capacity of a single computing element (*node*). High performance computing architectures based on distributed parallel processing principles can achieve a large computational efficiency as well as high scalability for solving complex problems [6].

In this section, we present the parallel model for processing a dataset consisting of many trip files and estimating the demand and OD matrices using a parallel/distributed system.

4.1 Parallel Model

Processing large volumes of data is needed to accurately estimate demand and OD matrices from ticket sales and bus location information. Initial experiments on a reduced portion of the dataset suggest that processing only one month of sales data demands over 18 days of computational time, when using a sequential algorithm in a regular desktop computer (Core i5 x2, 6 GB of RAM, Ubuntu 14.04). Therefore, we propose to run the estimation algorithms in parallel, making use of several computing units. The basic idea of the proposed parallel algorithm is to apply a data-parallel approach, dividing the dataset of sales and GPS records in chunks, following the Bag-of-Tasks (BoT) paradigm [3]. In our case, the BoT corresponds to a set of user trip files. Since each set of trip files are independent, they can be assigned to different compute nodes (slaves) for processing. Using a master-slave architecture seems appropriate since the slave processes do not need to share information with each other.

Fig. 3 shows the proposed master-slave model for processing trip data. Initially, the master collects the data to be processed and filters inconsistent records. Next, the master sends the BoT to each slave in the slave pool in order to perform the assigned computation task. Then, each slave node runs the designated estimation procedure. Finally, the master receives the partial results and store them to join and create the final demand and OD matrices.

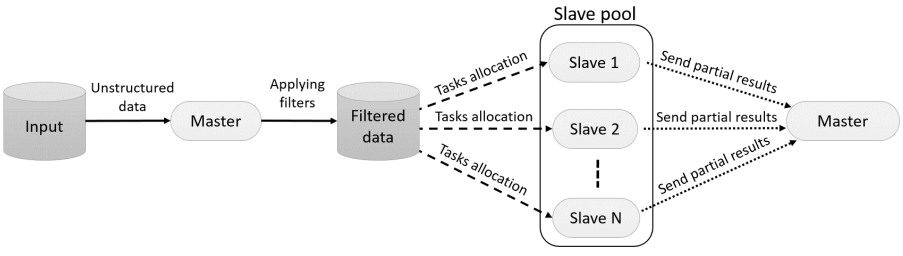

Fig. 3. Diagram of the proposed parallel model for demand and OD matrices estimation

Two variants of the proposed algorithm were implemented, one for each of the two different estimation procedures presented in Section 3.1. Both variants follow the same general approach which is specified in Section 4.2.

4.2 Implementation details

The implemented algorithms were designed using Python 2.7.5. The cross-platform open-source geographic information system QGIS [16] was used to manage geographic information corresponding to bus location and bus stops data.

We applied dispy [15] for creating and distributing parallel tasks among several computer nodes. dispy is a Python framework that allows executing parallel processes, supporting many different distributed computing infrastructures. The main features of the framework include tasks distribution, load balancing, and fault recovery. The dispy framework provides an API for the user to define clusters and schedule jobs to execute on those clusters. Creating a cluster in dispy consists of packaging computation fragments (code and data) and specifying parameters that control how the computations are executed, such as which nodes can execute each computation. Listing 1 presents the main methods used with dispy to create jobs and assign them to slave nodes in order to estimate demand and OD matrices.

Listing 1 dispy script for job creation and distribution

```
# function 'estimationOD'
def estimationOD(tripFile):
    #code for estimating origin-destination
    return pairsOD
#main
if __name__ == '__main__':
    # modules imported, are not available in job computations
    import dispy
    cluster = dispy.JobCluster(estimationOD,depends=[settings,
                               data, visual,...],...more parameters)
    jobs = []
    #Create tasks related with each trip file
    for tripFile in os.listdir(settings.path_to_trip_dataset):
        job = cluster.submit(tripFile)
        jobs.append(job)
        for job in jobs: #Execute jobs into the nodes availables
            job()# waits for job to finish and returns results
            pairsOD = job.result
    for pairs in pairsOD:
        #OD matrix is reconstructed with partial results
        #stored in variable 'pairsOD'
```

A list of parameters are needed to set a dispy cluster. First of all, the program to execute in each node must be indicated (in our case, the O-D matrix estimation procedure). In addition, the list of nodes available to execute the jobs, and a list of dependencies needed for computation must be specified (in our application the only dependency is the availability of the QGIS software).

Once a cluster is created, jobs can be scheduled to execute at a certain node. dispy will execute the job on an available processor in the defined cluster. When a submitted job is called, it returns that job's execution result, waiting until the job is finished if it has not finished yet. After a job is finished the information about the pairs origin-destination found is used to build the OD matrices.

Each slave keeps track of the index of the last file or line processed. Therefore, in case of a system failure it is possible to resume the execution from the last processed record, without the need of starting the process from the beginning.

In our approach, the master creates a set of BoT where each task corresponds to all the trip records of a single passenger. Then, each BoT is distributed using Dispy across the different slaves to execute the estimation algorithms. It is important to choose the amount of passengers' trip records to assign to each slave in order to optimize the execution time, avoiding costly communications between the slaves and the master. This parameter is configured in the experimental analysis presented in Section 5. Finally, the master node distributes tasks to slaves on demand, and obtains the results computed by each slave to gather them to return the final solution

5 Experimental analysis

This section describes the experimental analysis performed to assess the efficiency of the parallel algorithms developed. The computational platform used and the problem instances are detailed. Finally, the computational efficiency results are reported and commented.

5.1 Computational platform

The experimental analysis was performed on an AMD Opteron 6172 Magny Cours processors with 24 cores at 2.26 GHz, 72 GB RAM, with CentOS Linux 5.2 operating system from Cluster FING, the high performance computing infrastructure at Facultad de Ingeniería, Universidad de la República, Uruguay [13].

5.2 Problem instance and metrics

Problem instance. For the experimental analysis, the complete dataset corresponding to the ITS in Montevideo for January 2015 was processed, including ticket sales and bus location data. This dataset holds the mobility information for over half a million smart cards (corresponding to more than 13 million trips).

Computational efficiency metrics. In order to evaluate the computational efficiency of the proposed parallel algorithm we evaluate the *execution time* and the *speedup*. If we denote T_m the execution time for an algorithm using m processors, then the speedup is the ratio between the (larger) execution time on one processor T_1 and the (smaller) execution time on m processors T_m. This ratio value is presented in Eq. 1

$$S_m = \frac{T_1}{T_m} \tag{1}$$

5.3 Results and discussion

In the proposed master-slave parallel model it is necessary to define the size of the BoT assigned to each slave to compute, in order to have an appropriate load balance and avoid excessive communication between the slaves and the master. Several executions were performed varying the size of the BoT as well as the number of cores used. The experimental results are reported on Table 1. The number of cores (#cores) and the size of the BoT (#BoT) used in each experiment are indicated. Then, for each combination of these values, the best (i.e., minimum), average, and standard deviation of execution time and speedup values are reported for both direct and transfer trips. Execution times are reported in minutes and the results correspond to 5 independent executions of the algorithm using each configuration of #cores and #BoT.

#cores	*#BoT*	*direct trips*		*transfer trips*	
		avg. time$\pm\sigma$ (best)	speedup	avg. time$\pm\sigma$ (best)	speedup
1	1	25920	-	30240	-
16	5000	2092.1±3.4 (2089.6)	12.40	2648.9±3.2 (2645.5)	11.43
16	10000	2372.4±1.8 (2371.1)	10.92	3068.8±3.5 (3063.2)	9.87
24	5000	1582.7±2.4 (1579.4)	16.41	2371.1±2.5 (2368.1)	12.76
24	10000	1858.2±2.1 (1855.9)	13.96	2617.9±3.3 (2614.3)	11.56

Table 1. Execution time results and performance analysis

The experimental results obtained suggest that the parallel approach is an appropriate strategy for significantly improving the efficiency of the data processing for demand and O-D matrices estimation. Promising speedup values were obtained, up to **16.41** for the direct trips processing and using a BoT of 5000 trips and executing in 24 nodes. These results confirm that the proposed master/slave parallel model allows improving the execution time of the computational tasks by taking advantage of multiple computing nodes.

Furthermore, the computational efficiency results indicate that the size of the BoT (i.e., the amount of passengers' trip data given to each slave to process at once) has a significant impact on the overall execution time of the algorithm. Execution times were reduced when using the smallest size for the BoT (5000). Further experiments should be performed to assess if using a smaller size for the BoT is still more efficient, and to determine the tradeoff value before the communications between the slaves and the master become more expensive and have a negative impact on the execution time.

Using 24 cores and tasks with the trip data corresponding to 5000 passengers, the proposed strategy allows improving in up to 54.4% the efficiency when compared to using 12 cores and a BoT size of 5000, and up to 57.9% against a sequential algorithm running on a single computing node. This efficiency allows processing the full information of GPS and trip data for one year (more than 130 GB) in 33 days, a significant improvement over the 468 days demanded by a sequential algorithm.

6 Conclusions and future work

In this work, we proposed and implemented an efficient estimation method for obtaining demand and origin-destination matrices from real smartcard data of bus ticket sales. The proposed procedure demands estimating the alighting stops, since passengers do not validate the smartcard when getting off the bus. Two different approaches are proposed depending on whether the trip record correspond to a direct trip or to a bus transfer. For the estimations we considered similar assumptions to other works in the related literature.

Due to the large volume of data to be processed, we designed and implemented a parallel version of the estimation algorithm, following the master/slave parallel model. When compared to a sequential algorithm, the proposed parallel model reduces execution time from 56120 to 3954 minutes and achieves speed up values of 16.41 when using 24 cores in the best case.

We identify three main lines of future work: i) validate the computed demand and OD matrices using other sources of data, such as surveys; ii) incorporate machine learning techniques to infer destinations with high accuracy, for example by identifying recurrent destinations of a single passenger; iii) take advantage of the computed mobility data to address optimization problem that arise in most modern ITS, such as bus route design, bus stop location, bus timetabling, etc.

References

1. Bell, M.: The estimation of an origin-destination matrix from traffic counts. Transportation Science 17(2), 198–217 (1983)
2. Chen, C., Ma, J., Susilo, Y., Liu, Y., Wang, M.: The promises of big data and small data for travel behavior (aka human mobility) analysis. Transportation Research Part C: Emerging Technologies 68, 285–299 (2016)

3. Cirne, W., Brasileiro, F., Sauvé, J., Andrade, N., Paranhos, D., Santos-Neto, E.: Grid computing for bag of tasks applications. In: Proceedings of the 3^{rd} IFIP Conference on E-Commerce, E-Business and EGovernment (2003)
4. Deakin, M., Waer, H.: From Intelligent to Smart Cities. Taylor & Francis (2012)
5. Figueiredo, L., Jesus, I., Machado, J.T., Ferreira, J., de Carvalho, J.M.: Towards the development of intelligent transportation systems. In: Intelligent transportation systems. vol. 88, pp. 1206–1211 (2001)
6. Foster, I.: Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1995)
7. Grava, S.: Urban Transportation Systems. McGraw-Hill Education (2002)
8. Huang, E., Antoniou, C., Lopes, J., Wen, Y., Ben-Akiva, M.: Accelerated on-line calibration of dynamic traffic assignment using distributed stochastic gradient approximation. In: 13^{th} International IEEE Conference on Intelligent Transportation Systems. pp. 1166–1171 (2010)
9. Intendencia de Montevideo: Plan de movilidad urbana: hacia un sistema de movilidad accesible, democrático y eficiente (2010)
10. Massobrio, R., Pias, A., Vázquez, N., Nesmachnow, S.: Map-Reduce for Processing GPS Data from Public Transport in Montevideo, Uruguay. In: 2^{nd} Argentinian Symposium on Big Data (AGRANDA) (2016)
11. Mellegård, E.: Obtaining Origin/Destination-matrices from cellular network data. Master's thesis (2011)
12. Munizaga, M.A., Palma, C.: Estimation of a disaggregate multimodal public transport origin–destination matrix from passive smartcard data from santiago, chile. Transportation Research Part C: Emerging Technologies 24, 9–18 (2012)
13. Nesmachnow, S.: Computación científica de alto desempeño en la Facultad de Ingeniería, Universidad de la República. Revista de la Asociación de Ingenieros del Uruguay 61, 12–15 (2010)
14. Pelletier, M.P., Trépanier, M., Morency, C.: Smart card data use in public transit: A literature review. Transportation Research Part C: Emerging Technologies 19(4), 557–568 (2011)
15. Pemmasani, G.: dispy: Distributed and parallel computing with/for python. http://dispy.sourceforge.net/, [Online; accessed July 2016]
16. QGIS Development Team: QGIS Geographic Information System. Open Source Geospatial Foundation (2009), http://qgis.osgeo.org, [Online; accessed July 2016]
17. Sun, C.: Dynamic origin/destination estimation using true section densities. Tech. Rep. UCB-ITS-PRR-2000-5, University of California, Berkeley
18. Sussman, J.: Perspectives on Intelligent Transportation Systems (ITS). Springer Science + Business Media (2005)
19. Toole, J.L., Colak, S., Sturt, B., Alexander, L.P., Evsukoff, A., González, M.C.: The path most traveled: Travel demand estimation using big data resources. Transportation Research Part C: Emerging Technologies 58, Part B, 162–177 (2015)
20. Trépanier, M., Tranchant, N., Chapleau, R.: Individual trip destination estimation in a transit smart card automated fare collection system. Journal of Intelligent Transportation Systems 11(1), 1–14 (2007)
21. Wang, W., Attanucci, J., Wilson, N.: Bus passenger origin-destination estimation and related analyses using automated data collection systems. Journal of Public Transportation 14(4), 131–150 (2011)