Towards a Cloud Computing Paradigm for Big Data Analysis in Smart Cities

[1] *Renzo Massobrio <renzom@fing.edu.uy>*
[1] *Sergio Nesmachnow <sergion@fing.edu.uy>*
[2] *Andrei Tchernykh <chernykh@cicese.mx>*
[3] *Arutyun Avetisyan <arut@ispras.ru>*
[4] *Gleb Radchenko <gleb.radchenko@susu.ac.ru>*
[1] *Universidad de la República, Montevideo 11300, Uruguay.*
[2] *CICESE Research Center, Ensenada , B.C. 22860, México*
[3] *Institute for System Programming of the RAS, Moscow, 109004, Russia*
[4] *South Ural State University, Chelyabinsk, 454080, Russia.*

Abstract. In this paper, we present a Big Data analysis paradigm related to smart cities using cloud computing infrastructures. The proposed architecture follows the MapReduce parallel model implemented using the Hadoop framework. We analyse two case studies: a quality-of-service assessment of public transportation system using historical bus location data, and a passenger-mobility estimation using ticket sales data from smartcards. Both case studies use real data from the transportation system of Montevideo, Uruguay. The experimental evaluation demonstrates that the proposed model allows processing large volumes of data efficiently.

Keywords: cloud computing; big data; smart cities; intelligent transportation systems.

1. Introduction

One of Smart City challenges is to use information and communications technologies to manage cities' assets with the goal of improving the quality and performance of urban services. By applying these techniques, it is possible to reduce infrastructure and operational costs, increase efficiency in the use of resources, and facilitate the interaction between citizens and authorities [1]. Such techniques are often applied to transportation services due to the central role they play in modern cities.

Nowadays, many complex activities, which impose serious challenges to the mobility of citizens are developed in modern cities [2]. Public transportation plays a major role in the city transit system, especially in dense urban areas. However, many public transportation systems are not able to cope with the growing mobility demand. In order to address this issue, authorities and decision-makers need to have

a deep understanding of the overall picture and details of the people mobility, including up-to-date information on the use of public transportation [3]. Unfortunately, due to the lack of financial and human resources, public administration often has scarce and outdated mobility data. Frequently, data are gathered but are not analyzed and used to improve public/private transportation infrastructure. For this reason, improving the decision-making processes related to urban mobility becomes mandatory in most modern cities. The adequate paradigm of smart cities allows to take advantage of data coming from a plethora of sources that can be processed to understand mobility in the cities.

Intelligent Transportation Systems (ITS) are a key component of smart cities. ITS are defined as those systems that integrate synergistic technologies, computational intelligence, and engineering concepts applied to transport systems in order to improve traffic congestion, safety, transport efficiency and environmental impact [4]. ITS allow gathering different data regarding transportation and mobility in the cities [5]. In big urban areas, ITS generate huge volumes of data that can be processed to extract valuable information about the mobility of citizens.

This article presents a framework to process ITS big data in an efficient manner, taking advantage of cloud computing techniques. The cloud computing includes a set of techniques that allows taking advantage of many computing resources in order to solve complex problems by applying the distributed computing approach. When facing large problems such as big data processing, pattern recognition, deep learning, etc., distributed computing comes to help researchers to get acceptable performance by applying a cooperative approach. The cooperation is achieved by splitting a big problem into many smaller subproblems to be solved in parallel in different computing resources, in order to speed up the processing [6]. In the last ten years, different frameworks have been developed to analyze big amounts of information applying distributed computing in cloud systems [7].

In this work, we apply and adapt a standard framework for big data in cloud systems to process large volumes of ITS data to improve public transportation in the context of smart cities. This proposal is our first step to provide an integrated and efficient tool for big data analysis in smart cities to benefit both citizens and city administrators.

Two case studies are presented: a quality-of-service assessment of public transportation system using historical location data, and a passenger-mobility estimation of ticket sales data using smartcards. Both case studies correspond to real ITS data from the city of Montevideo, Uruguay. Up to now, there is no automatic processing of ITS data to aid passengers or administrators in Montevideo, Uruguay. Thus, our proposal fulfills a specific need of the city with a concrete social value.

The remainder of the article is organized as follows. Section 2 outlines the basic concepts related to cloud computing, introduces the MapReduce paradigm and the Hadoop Framework. Section 3 presents the related literature on smart cities and ITS data, focusing on distributed and cloud processing of these data. Section 4 presents

the proposed model for ITS data analysis in the cloud. Section 5 presents the two case studies and experimental results. Finally, Section 6 presents the main conclusions of this article as well as the main lines of future work.

2. Cloud computing, MapReduce and the Hadoop framework

This section describes the main ideas behind the cloud computing concept and introduces the MapReduce model and its Hadoop implementation.

2.1. Distributed computing and cloud computing

Distributed computing is an umbrella term that defines a computing model and a set of programming strategies based on using many computing elements, connected through a network, to solve problems with large computing demands [8]. Specific distributed processes executing on those computing elements communicate, synchronize, and coordinate their actions in order to cooperatively solve a common problem, usually applying domain-decomposition or functional-decomposition programming techniques. In the last twenty years, distributed computing has been applied to efficiently solve complex problems with large computing demands in many application areas [6].

Cloud computing is a specific type of distributed computing model that makes use of geographically distributed computing resources and Internet-based communications to implement and provide services that are offered to users [9]. The main idea behind cloud computing is that the required services are used on demand, and users can access via many devices from any place on the globe, making use of a shared pool of configurable resources (e.g., computing power, storage, networking, applications, etc.) that underlies the model.

The cloud is largely used as the infrastructure for processing large volumes of data. One of the most popular models for big data processing in the cloud is the MapReduce model, which is described next.

2.2. MapReduce

MapReduce is a programming model for processing large volumes of data using parallel/distributed algorithms on cluster, grid, and cloud computing systems.

The MapReduce paradigm has two main simple operations. First one is based on applying a map function that performs (in parallel, using many computing resources) a set of computing tasks, including filtering, sorting, and/or computation. The second one is a reduce function that performs a summary operation based on the results of the map function [10].

Processes that use the MapReduce paradigm are called MapReduce jobs. A MapReduce job has two main phases: the map phase and reduce phase. The map phase is composed of four steps: record reader, mapper, combiner, and partitioner. The output of the map phase is a set of intermediate keys and values that are

grouped by key, which send to the reduce phase of the algorithm. The reduce phase is composed of four steps: shuffle, sort, reduce, and output format.

MapReduce libraries have been written in many programming languages. The most popular is Apache Hadoop, which is briefly described next.

2.3. Hadoop

Hadoop is currently one of the most popular frameworks to analyze large volumes of information using the MapReduce programming model. Hadoop is a distributed system that includes an open source distributed file system implementation, called Hadoop Distributed File System (HDFS) [11].

Hadoop provides a high level of abstraction for defining tasks. It allows users to implement distributed algorithms easily, even for developers, with little knowledge about distributed computing. These algorithms can be executed on many computing resources to analyze huge amounts of information in reasonable execution time. Hadoop includes a full ecosystem that provides extended capabilities for task management, distributed programming, database interfaces, and other features.

2.4. Hadoop MapReduce implementation

The most known and widely used implementation of the MapReduce programming model is the Hadoop implementation. Although the MapReduce paradigm is easy to understand and describe, it is not always easy to express an algorithm in terms of map and reduce functions.

In Hadoop, the mapper input is usually raw data saved in HDFS. The default format is a text, which specifies the lines of the raw file as values for the mapper, and the byte offset of the beginning of the line from the beginning of the file as key. A MapReduce job consists of the input data, MapReduce program, and configuration information. Hadoop runs the job by dividing it into map tasks and reduce tasks. The user can implement custom partitioners, record readers, input formats, and combiners depending on the specific needs [7].

A Hadoop cluster includes the master node and multiple slave nodes. The master node has several processes: JobTracker, TaskTracker, namenode, and datanode. A slave node has only datanode and TaskTracker processes. The JobTracker coordinates all jobs runned on the system by scheduling tasks on TaskTrackers. TaskTrackers run tasks and send progress reports to the JobTracker, which keeps a record of the overall progress of each job. If a task fails, the JobTracker can reschedule it to execute on a different TaskTracker. Both the namenode and the datanode belong to the HDFS cluster.

The execution of a MapReduce job in Hadoop is as follows. First, a MapReduce job is created in the client node which is executed inside a Java Virtual Machine (JVM). Then, the JobClient submits a new job to the JobTracker, which centralizes all job executions and a new job identification is returned. After that, the execution file and distributed cache information needed for execution are copied to the nodes. Then,

the job is submitted and JobTracker, using the job ID, initializes the job and retries the input for the job. TaskTrackers communicate with the JobTracker sending information about availability and running capacity. The JobTracker assigns the job to a TaskTracker with availability to run the map or reduce tasks. The TaskTracker retrieves the resources to run the task. Finally, the TaskTracker launches a new JVM and executes the map or reduce task.

3. Related work

Several recent papers describe application of distributed and cloud computing approaches to process large volumes of data from different ITS sources. In this section, we present a brief review of related works.

The advantages of big data analysis for social transportation have been studied in the general review of the field by Zheng et al. [12]. The authors analyze several sources of information, including vehicle trajectories (e.g., GPS coordinates, velocity), incident reports, human mobility (using GPS and WiFi signals), social networking (e.g., textual posts, address), and web logs (e.g., user identification, comments). The advantages and limitations of using each type of data are commented. Several novel ideas to improve public transportation and implement the ITS paradigm are also reviewed, including crowdsourcing for collecting and analyzing real-time or near real-time traffic information, and data-based agents for driver assistance and analysis of human behavior. A conclusion of how to integrate all concepts in a data-driven social transportation system that contributes to the next generations of ITS for improving traffic safety and efficiency is also presented.

Computational intelligence techniques have been recently applied for ITS design. Oh et al. [13] proposed a sequential search strategy for traffic state prediction combining a Vehicle Detection System and K nearest neighbors (kNN). It outperforms a traditional kNN approach obtaining significantly more accurate results, while maintaining good efficiency and stability properties.

Shi and Abdel-Aty [14] applied the random forest data mining technique and Bayesian inference to process large volumes of data from a Microwave Vehicle Detection System. The main goal is to identify the factors contributing to crashes in real-time. A reliability model is included into the analysis during peak hours, higher volume/lower speed at upstream locations, and congestion index. The main conclusion is that congestion has the most impact on rear-end crashes.

Ahn et al. [15] applied a supervised learning approach using Support Vector Regression and a Bayesian classifier for building a real-time traffic flow prediction system. Initially, two stages of data preparation and noise filtering are applied over the raw data. These are common techniques used in related articles. We also apply them in our proposal. Then, a traffic flow model is proposed using a Bayesian framework. Regression techniques are used to model the time-space dependencies and relationships between roads. The performance of the proposed method is studied on traffic data from Gyeongbu, the Seoul-Busan corridor in South Korea.

The experimental results showed that the approach using SVR-based estimation outperform traditional linear regression methods in terms of accuracy.

Chen et al. [16] proposed a model to efficiently predict the traffic speed on a given location. It uses historical data from various sources including ITS data, weather conditions, and special events taking place in the city. To obtain accurate results the prediction model needs to be re-trained frequently in order to incorporate the most up-to-date data. The prediction model combines the kNN algorithm with a Gaussian Process Regression. Additionally, the results are computed using a MapReduce model, implemented under the Hadoop framework. The experimental evaluation was performed over a real scenario using data from the Research Data Exchange, a platform for ITS data sharing. The data corresponds to the I5N road section in San Diego, California, United States. The information includes speed, flow, and occupancy data measured using loop-detectors on the road, as well as visibility data taken from weather stations nearby. Experimental results showed that the proposed method is able to accurately predict traffic speed with an average forecasting error smaller than 2 miles per hour. Additionally, a 69% improvement on the execution time was achieved by using the Hadoop framework in a cluster infrastructure against running in a single machine.

Xia et al. [17] studied the real-time short-term traffic flow forecasting problem. To solve the problem, the K nearest neighbor algorithm is used in a distributed environment, following the MapReduce model using the Hadoop framework. The proposed solution considers the spatial-temporal correlation in traffic flow, i.e., current traffic at a certain road segment depends on past traffic (time dimension) and on traffic situation at nearby road segments (spatial dimension). These two factors can be controlled using weights in the proposed algorithm. The experimental analysis was performed using data of trajectories of more than 12000 GPS-equipped taxis in the city of Beijing, China, during a period of 15 days in November 2012. The data from first 14 days are used as the training set, and the last day is used for evaluating the computed results. The proposed algorithm allows to reduce the mean absolute percentage error by 8.5% to 11.5% on average over three existing techniques based on the k nearest neighbor algorithm. Additionally, the proposed solution achieved a computational efficiency of 0.84 in the best case.

More recently, Peña et al. [23] propose using a multiobjective cellular genetic algorithm to address the scheduling of a bus fleet with muliple types of vehicles with different sizes.

We identify several proposals for using big data analysis and computational intelligence methods to process large volumes of ITS data. Machine learning methods, such as regression, kNN and Bayesian inference are often used to identify traffic patterns and provide useful information for planning. However, there are few works focusing on improving the public transportation systems, especially considering the point of view of the users. In this context, our paper contributes state of the art with a specific proposal of a framework to analyze ITS data on the cloud to monitor and improve public bus transportation systems.

4. Proposed model

In this section, we describe the proposed framework for processing ITS data in the cloud to improve public transport systems.

4.1. Design and architecture

The problem is decomposed into two sub-problems: i) a pre-processing stage to properly prepare the input data for the next phase, and ii) the parallel/distributed processing ITS data from the public transportation system. A master-slave model is used to define and organize the control hierarchy and processing. Fig. 1 presents a conceptual diagram of the proposed system.

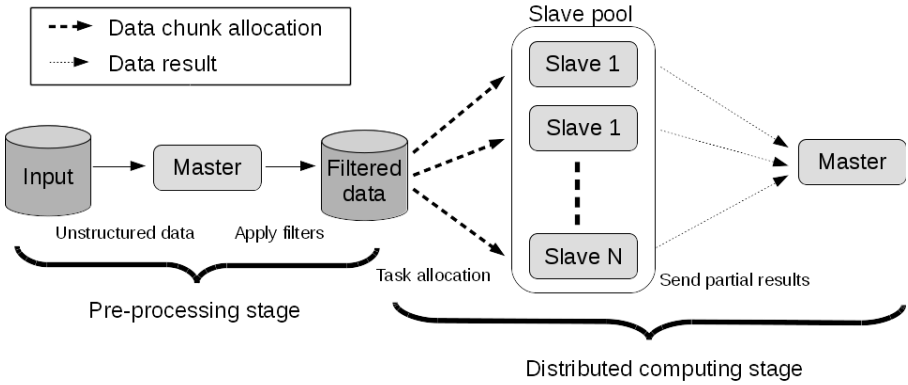

Fig. 1. Conceptual diagram of the proposed framework

During the pre-processing phase, the master prepares the data, filtering those records that do not contain useful information to be processed and classified, and orders the records for the processing stage. The filtering stage is dependent on the problem being solved. It is discussed in the case studies presented in Section 5. The data processing phase applies a data-parallel domain decomposition strategy for parallelization. After the pre-processing stage, the filtered and sorted data are split in chunks and distributed to several processing elements. The master process is in charge of controlling this task, partitioning the data and assigning each chunk to a slave for processing. Each slave receives a subset of the data from the master. The slaves processes collaborate in the data processing, following a single program multiple data (SPMD) model.

4.2. Implementation of the MapReduce application

The proposed parallel/distributed system for ITS data analysis is implemented using a MapReduce approach under the Hadoop framework. The system suits the

MapReduce model, since no inter-slave communications are required, and master-slave communications are limited to data distribution and results gathering. The standard MapReduce engine in Hadoop is applied, using one master node and several slave nodes. The master node uses the JobTracker process to send jobs to different TaskTracker processes associated to the slave nodes. Once the slaves finish their assigned processing task, each TaskTracker sends the results back to the JobTracker in the master node.

A fault tolerance mechanism is already included in Hadoop. Additionally, some features are activated to improve fault tolerance when processing ITS data: i) the possibility of discarding corrupt input lines, for cases where a record has corrupt information; and ii) the native replication mechanism in HDFS, in order to keep data replicated in different processing nodes.

The proposed implementation can be used following a Software as a Service (SaaS) paradigm in a real cloud computing environment, providing a useful service both citizens and authorities.

5. Case studies

This section describes the application of the proposed framework for processing two different kinds of ITS data. In both cases, the experimental evaluation is performed using real data corresponding to the ITS in Montevideo, Uruguay.

5.1. Quality of service metrics for public transportation using bus location data

This case study presents the problem of computing quality-of-service (QoS) metrics for public transportation system using data collected from on-board GPS devices installed in buses [24]. The data holds the information of the time and location for each bus, reported with a frequency of 10–30 seconds. The main objective is to compute relevant metrics to assess the efficiency of the public transport system, such as: i) the real travel time of each bus to reach previously defined remarkable locations in the city and ii) historical statistical information about the delays for each bus line along their route, to identify possible bottlenecks. The data must be classified and properly organized to allow reporting information corresponding to different days of the week and hours in the day, passenger demands and traffic mobility patterns vary with time.

There are at least two target groups that would benefit from the information computed in the proposed system: passengers and city authorities. From the point of view of public transport users, the information computed from historical and real-time data can help with their mobility decisions (e.g., prefer a certain bus line over others, decide to transfer between buses).

This information can be offered via intelligent ubiquitous mobile applications and websites. From the point of view of the city authorities, the statistical information is

useful for planning long-term modifications of the bus routes, timetables, bus stops locations, as well as to identify specific bottleneck situations.

A diagram of the proposed system is presented in Fig. 2. Buses send their current geographic location to a server in the cloud. The server performs the MapReduce processing of the collected GPS data from the different buses in real time. The results from this processing are published to be consumed by a mobile app for end-users and by a monitoring application for city authorities, following the traditional SaaS model for cloud computing.

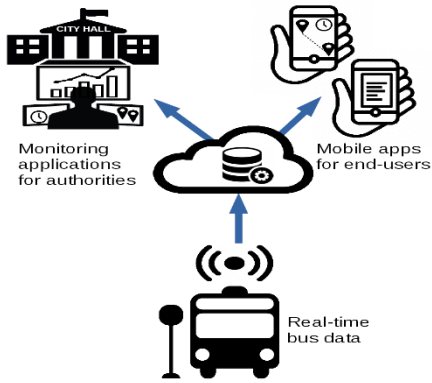

Fig. 2. System architecture to allow processing ITS data on the cloud

During the pre-processing stage, the master prepares the data by filtering records that hold non-useful data (e.g., corrupt GPS records). Additionally, the data is filtered to consider only those records that fall in the time range specified by the user. Finally, the records are sorted according to the bus identification number to allow efficient processing during the distributed phase.

During the distributed computing phase, the master process divides the dataset of useful GPS records that were generated in the previous phase and distributes the resulting subsets among the slave processes. Each subset includes GPS and time records associated with the same bus line. This allows each slave to do the processing independently, with no inter-slave communication, thus improving the computational efficiency. Each slave computes the travel time between each remarkable location along the bus route. During the reduce phase, the computed results are used to compute the relevant QoS statistics that will be presented to the user. The results are ordered by bus line and remarkable location. Finally, each slave return the partial results to the master, which is in charge of group them and outputs the final results to the user.

The experimental evaluation is performed over the cloud infrastructure provided by Cluster FING, Universidad de la República, Uruguay [18], using AMD Opteron 6172 Magny Cours (24 cores) processors at 2.26 GHz, 24GB RAM, and CentOS

Linux 5.2 operating system. For the experimental analysis, we use real historical ITS data provided by the city authorities of Montevideo, Uruguay. 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 case study, we consider the complete dataset of ticket sales and bus location for 2015, comprising nearly 200 GB of data. Bus location data contains information about the position of each bus, sampled every 10 to 30 seconds.

The complete dataset is divided into smaller ones that are used to define different scenarios conceived to assess the performance of the proposed system, including different input file sizes, different time intervals, and different number of map and reduce processes. We work with datasets containing 10GB, 20GB, 30GB, and 60GB of ITS data, and consider different time intervals for computing statistics (3 days, 1, 2, 3, and 6 months). We apply the traditional metrics to evaluate the performance of parallel algorithms: speedup and efficiency.

Table 1 reports the computational efficiency for the proposed systems under different scenarios, varying the following parameters: the size of the input data in GB (#I), the number of days considered for the statistics (#D), the number of mappers (#M), and reducers (#R).

For each scenario, the execution time (in seconds) using 1 (T1) and 24 cores (TN), speedup (SN), and efficiency (EN). Reported values correspond to means computed over five independent executions.

Table. 1. Computational efficiency results

#I	#D	#M	#R	T_1(s)	T_N(s)	S_N	E_N		#I	#D	#M	#R	T_1(s)	T_N(s)	S_N	E_N
10	3	14	8	1333.9	253.1	5.27	0.22		30	30	22	22	5052.9	281.1	17.98	0.75
10	3	22	22	1333.9	143	9.33	0.39		30	60	14	8	5927.9	383.4	15.46	0.64
10	30	14	8	2108.6	178	11.84	0.49		30	60	22	22	5927.9	311.4	19.04	0.79
10	30	22	22	2108.6	187.3	11.26	0.47		30	90	14	8	7536.9	416.6	18.09	0.75
20	3	14	8	2449	351.1	6.98	0.29		30	90	22	22	7536.9	349.2	21.58	0.9
20	3	22	22	2449	189.8	12.9	0.54		60	3	14	8	7249.6	944	7.68	0.32
20	30	14	8	3324.5	275.6	12.06	0.5		60	3	22	22	7249.6	362.1	20.02	0.83
20	30	22	22	3324.5	238.8	13.92	0.58		60	60	14	8	10037.1	672.6	14.92	0.62
20	60	14	8	4762	300.8	15.83	0.66		60	60	22	22	10037.1	531.4	18.89	0.79
20	60	22	22	4762	264.7	17.99	0.75		60	90	14	8	11941.6	709.6	16.83	0.7
30	3	14	8	3588.5	546.9	6.56	0.27		60	90	22	22	11941.6	648.9	18.4	0.77
30	3	22	22	3588.5	179.6	19.99	0.83		60	180	14	8	19060.8	913.7	20.86	0.87
30	30	14	8	5052.9	359.6	14.05	0.59		60	180	22	22	19060.8	860.3	22.16	0.92

The results in Table 1 show that the distributed approach allows significantly improving the efficiency over the sequential version, especially when processing large datasets. Speedup up to 22.16 (when using 24 cores) is obtained, corresponding to a computational efficiency of 0.92. The distributed implementation allows reducing the execution time from about 6 hours to 14 minutes when processing the largest dataset. This performance enhancement is essential for authorities to provide a fast response to specific bottleneck situations and to analyze different metrics and scenarios for both users and administrators.

Using 22 mappers and 22 reducers allows obtaining the best efficiency, improving up to 15% the execution time (9% in average) over the ones demanded when using 14 mappers and 8 reducers. When dealing with small datasets, the data is split in small pieces, thus generating low loaded processes and not improving notably over the execution time of the sequential algorithm.

5.2. Origin-destination matrix generation using historical smartcard data

In order to solve urban transportation optimization problems, it is imperative to understand the mobility patterns of citizens and demand distribution. This information is often represented using matrices: i) origin-destination (OD) matrices that indicate the amount of people moving from each point in the city in a given period of time [19]; ii) demand matrices that measure the number of bus tickets sold between each location in the city.

Traditionally, these matrices are built using data from surveys performed to passengers and drivers. However, surveys offer provide only a partial vision of the mobility patterns, require large funds from authorities to be performed, and do not contain up-to-date information.

In our case study, we propose a different approach for building demand and OD matrices [25]. We calculate and update them based on processing real data from ITS, considering tickets sold with and without smart cards, and location of buses. Taking into account the large computing time demanded when dealing with huge volumes of ITS data, we propose applying the distributed computing model described in Section 4 to speed up the processing.

The main challenge of generating demand and OD matrices using data from tickets sales is that passengers validate their smart cards only when they boarding but not when they get off the bus. Therefore, while the origin of each trip is known, it is necessary to estimate the destination. Furthermore, passengers are not forced to use smart cards to pay for their ticket (they can pay by cash to the driver). Therefore, some sale records do not hold complete information linked to any specific individual and do not allow to track several trips made by a single passenger.

The model used for estimating OD matrices is based on the trip sequence reconstructing for passengers that use smart cards, following a similar approach to

that applied in the related literature [20, 21, 22]. We assume that each smart card corresponds to a single passenger. 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 get off the bus. We identify two distinct cases for estimating destinations: transfer trips and direct trips. The main details of each case are presented below.

Transfer trips. In a transfer trip, passengers pay for their ticket when boarding the first bus by using a smart card uniquely identified by its cardID. Later, they can take other one or more buses within the time limits permitted by the ticket, without buying a new ticket validating their smartcard on each new bus they board.

This allows detecting whether a smart card record corresponds to a new trip (i.e., a new ticket sale) or to a transfer between buses (i.e., by smart card validating). We assume that passengers avoid excessive walking during transfers, and 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 boarding (changing) points from one bus to another one by looking for the closest bus stop corresponding to the line.

Direct trips. We consider direct trips as those that do not involve a bus transfer. Additionally, we consider the last leg of a series of bus transfers as a direct trip as well. In both cases, the challenge consists in accurate estimating the 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 located near 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 of that same day. In order to estimate destinations, we try to "chain" the trips made by each passenger on a single day. For this purpose, we iterate through all the trips done by each passenger in a 24 hour period. For each new trip, we try to estimate the alighting point by looking for a bus stop located in a predefined distance range from the bus stop of the previous trip. We also keep a log of those trips that could not be accurately chained in order to report statistics regarding the efficacy of the method.

In this case study, the pre-processing stage of the model presented in Section 4 filters those sale records that hold incoherent information. We consider that a GPS measure is inaccurate, if it falls further than 50 meters away from the route the corresponding bus. The filtered data are split in chunks based on the ID of the smartcard and distributed to the slaves. Thus, each slave has the information of all the trips of a single user, avoiding inter-slave communications. In order to improve performance, at the end of the pre-processing stage, records are sorted by date, so they can be sequentially processed by each slave.

For the experimental analysis, the complete ITS dataset, corresponded to 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).

In the proposed master-slave parallel model it is necessary to calibrate the size of the Bag of Tasks (BoT) assigned to each slave. An appropriate BoT size offers a good load balance, avoiding excessive communication between the slaves and the master. Five independent executions were performed varying the size of the BoT as well as the number of cores used.

The obtained experimental results suggest that the proposed distributed model allows significantly improve the efficiency of the mobility data processing over a sequential one. Promising speedup, up to 16.41, was obtained for direct trips using a BoT of 5000 trips and 24 computing nodes. These results confirm that the proposed distributed model allows improving the execution time of the computational tasks by taking advantage of multiple computing nodes.

Furthermore, results indicate that the size of the BoT has a significant impact on the overall execution time of the algorithm, with smaller BoT achieving better results. Further experiments should be carried out to establish a trade off value when the amount of master-slave communications does not become too expensive and have a negative impact on the execution time.

6. Conclusions and future work

In this work, we propose and implement a distributed computing model to process large volumes of ITS data in the cloud using MapReduce over the Hadoop Framework. A review of the related literature is presented, with discussion of previous attempts of using distributed computing to process ITS data in the context of smart cities. The efficiency of the proposed model is depicted using two case studies: i) computing QoS metrics of public transportation based on bus location data; ii) estimating origin-destination matrices from ticket sales data. In both case studies, the proposed distributed model allows to significantly reduce the execution times needed to process large volumes of historical data. The experimental analysis of both cases is performed using real ITS data from the city of Montevideo, Uruguay.

The main lines of future work include: i) evaluating different cloud solutions to deploy the proposed data processing algorithm including Spark, Pivotal GreenPlum, Amazon Redshift, etc.; ii) incorporating different ITS and sources of data; iii) developing applications for citizens to access the computed information in a user-friendly manner; iv) applying the obtained information to solve optimization problems such as bus route design, bus stop location, bus timetabling, etc.

Acknowledgment. The work of S. Nesmachnow and R. Massobrio is partly funded by ANII and PEDECIBA, Uruguay. This work is partially supported by Government of the Russian Federation, Act 211, contract № 02.A03.21.0011, and

CONACYT (Consejo Nacional de Ciencia y Tecnología, México), grant no. 178415. Datasets used in this paper are from Intendencia de Montevideo.

References

[1]. Deakin, M., & Al Waer, H. (2011). From intelligent to smart cities. Intelligent Buildings International, 3(3), 140-152.
[2]. Grava, S. (2003). Urban transportation systems. Choices for communities.
[3]. Chen, C., Ma, J., Susilo, Y., Liu, Y., Wang, M.: (2016). The promises of big data and small data for travel behavior (aka human mobility) analysis. Transportation Research Part C: Emerging Technologies 68, 285–299.
[4]. Sussman, J. S. (2008). Perspectives on intelligent transportation systems (ITS). Springer Science & Business Media.
[5]. Figueiredo, L., Jesus, I., Machado, J. T., Ferreira, J., & de Carvalho, J. M. (2001). Towards the development of intelligent transportation systems. In Intelligent transportation systems (Vol. 88, pp. 1206-1211).
[6]. Foster I. (1995). Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
[7]. White T. (2009). Hadoop: The Definitive Guide (1st ed.). O'Reilly Media, Inc..
[8]. Attiya H. & Welch J. (2004). Distributed Computing: Fundamentals, Simulations and Advanced Topics. John Wiley & Sons.
[9]. Buyya R., Broberg J., & Goscinski. A. M. (2011). Cloud Computing Principles and Paradigms. Wiley Publishing.
[10]. Dean J. & Ghemawat S. (2008). MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (January 2008), 107-113.
[11]. Shafer, J., Rixner, S., & Cox, A. L. (2010). The hadoop distributed filesystem: Balancing portability and performance. In *IEEE International Symposium on Performance Analysis of Systems & Software* (pp. 122-133).
[12]. Zheng, X., Chen, W., Wang, P., Shen, D., Chen, S., Wang, X., ... & Yang, L. (2016). Big data for social transportation. IEEE Transactions on Intelligent Transportation Systems, 17(3), 620-630.
[13]. Oh, S., Byon, Y. J., & Yeo, H. (2016). Improvement of Search Strategy With K-Nearest Neighbors Approach for Traffic State Prediction. IEEE Transactions on Intelligent Transportation Systems, 17(4), 1146-1156.
[14]. Shi, Q., & Abdel-Aty, M. (2015). Big data applications in real-time traffic operation and safety monitoring and improvement on urban expressways. Transportation Research Part C: Emerging Technologies, 58, 380-394.
[15]. Ahn, J., Ko, E., & Kim, E. Y. (2016). Highway traffic flow prediction using support vector regression and Bayesian classifier. In 2016 International Conference on Big Data and Smart Computing (BigComp) (pp. 239-244). IEEE.
[16]. Chen, X. Y., Pao, H. K., & Lee, Y. J. (2014). Efficient traffic speed forecasting based on massive heterogenous historical data. In Big Data (Big Data), 2014 IEEE International Conference on (pp. 10-17). IEEE.
[17]. Xia, D., Wang, B., Li, H., Li, Y., & Zhang, Z. (2016). A distributed spatial–temporal weighted model on MapReduce for short-term traffic flow forecasting. Neurocomputing, 179, 246-263.

[18]. Nesmachnow S. (2010). Computación científica de alto desempeño en la Facultad de Ingeniería, Universidad de la República. Revista de la Asociación de Ingenieros del Uruguay 61 (1), 12-15.
[19]. Yang H., Sasaki T., Iida Y., Asakura Y. (1992). Estimation of origin-destination matrices from link traffic counts on congested networks, Transportation Research Part B: Methodological, Volume 26, Issue 6, Pages 417-434.
[20]. Trépanier, M., Tranchant, N., & Chapleau, R. (2007). Individual trip destination estimation in a transit smart card automated fare collection system. *Journal of Intelligent Transportation Systems*, *11*(1), 1-14.
[21]. Wang, W., Attanucci, J. P., & Wilson, N. H. (2011). Bus passenger origin-destination estimation and related analyses using automated data collection systems. *Journal of Public Transportation*, *14*(4), 7.
[22]. Munizaga, M. A., & Palma, C. (2012). 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.
[23]. Peña D., Tchernykh A., Nesmachnow S., Massobrio S., Drozdov A. Y., Garichev S. N. (2016). Multiobjective vehicle type and size scheduling problem in urban public transport using MOCell. IEEE International conference Engineering & Telecommunications, Moscow, Russia.
[24]. R. Massobrio, A. Pías, N. Vázquez, & S. Nesmachnow (2016). Map-Reduce for Processing GPS Data from Public Transport in Montevideo, Uruguay. In 2do Simposio Argentino de Grandes Datos.
[25]. E. Fabbiani, P. Vidal, R. Massobrio, & S. Nesmachnow (2016). Distributed Big Data analysis for mobility estimation in Intelligent Transportation Systems. In Latin American High Performance Computing Conference.