EAI Endorsed Transactions
Smart Cities
Research Article EAI.EU

A distributed platform for big data analysis in smart cities: combining Intelligent Transportation Systems and socioeconomic data for Montevideo, Uruguay

Sergio Nesmachnow*, Sebastián Baña*, and Renzo Massobrio*

Universidad de la República, Herrera y Reissig 565, Montevideo, Uruguay

Abstract

This article proposes a platform for distributed big data analysis in the context of smart cities. Extracting useful mobility information from large volumes of data is crucial to improve decision-making processes in smart cities. This article introduces a framework for mobility analysis in smart cities combining Intelligent Transportation Systems and socioeconomic data for the city of Montevideo, Uruguay. The efficiency of the proposed system is analyzed over a distributed computing infrastructure, demonstrating that the system scales properly for processing large volumes of data for both off-line and on-line scenarios. Applications of the proposed platform and case studies using real data are presented, as examples of the valuable information that can be offered to both citizens and authorities. The proposed model for big data processing can also be extended to allow using other distributed (e.g. grid, cloud, fog, edge) computing infrastructures.

Received on 9 May 2017; accepted on 18 October 2017; published on 19 December 2017

Keywords: Smart cities, big data, distributed computing, Intelligent Transportation Systems

doi:10.4108/XX.X.X.XX

1. Introduction

The paradigm of smart cities proposes taking advantage of information and communication technologies to improve the quality and efficiency of urban services [1].

Modern cities are increasingly becoming sensed and instrumented. The embedding of smart devices into traditional city's physical systems together with the emergence of citizen sensors, such as mobile phones or "Internet of Things" (IoT) enabled domestic appliances, are generating vast volumes of data that present unprecedented opportunities as well as challenges. Extracting insights from these datasets is crucial to improve decision-making processes in cities and to achieve quality improvements and increase efficiency.

A particular sub-domain of a smart city are Intelligent Transportation Systems (ITS). ITS integrate synergistic technologies, computational intelligence, and engineering concepts to develop and improve transportation. ITS 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 [2]. ITS allow gathering large volumes of data by taking advantage of different sensors and devices present in current vehicles and infrastructure (e.g., passenger counters, GPS devices, video cameras, ticket vending machines). The development of smart tools that use data gathered by ITS infrastructure and vehicles has risen in the past years. These tools rely on efficient and accurate data processing (even in real-time), which poses an interesting challenge from the technological perspective. Furthermore, these data can be combined with more traditional data sources, such as sociodemographic data that are regularly and systematically collected by government agencies. This combined approach enables characterizing areas of study and helps answering questions such us how equitably the services delivered are and whether certain communities are disproportionally affected by poor service quality.

*Corresponding authors. Email: {sergion,sbana,renzom}@fing.edu.uy

In this context, applying distributed parallel computing and machine learning techniques arise as a promising methodology for processing large volumes of data to be used in services and applications targeting both citizens and authorities alike.

This article proposes a framework for capturing and processing large volumes of data in the context of the resolution of urban problems. The problems involve processing large volumes of data to offer real-time information to both citizens and transport authorities. The proposed framework applies distributed computing and big data processing methods to provide an easy-to-use and efficient solution. Furthermore, two specific applications of the proposed framework are presented: i) an analysis of public transportation in the city of Montevideo, Uruguay that uses historical geo-spatial data from vehicles combined with socioeconomic datasets to withdraw conclusions regarding both quality and equability of the services provided [3] and ii) the estimation of OD matrices and mobility patterns for the same public transportation system [4].

An experimental analysis is also reported, studying the computational efficiency of the proposed framework over both applications. The main results demonstrate the efficiency and scalability of the proposed solution, making it a promising approach to be applied in modern smart cities.

The article is organized as follows. Section 2 describes the generic framework proposed for distributed big data analysis for ITS in the context of the smart city paradigm and introduces the two practical applications studied. A review of related works on distributed big data analysis for smart city applications is presented in Section 3. Section 4 describes the proposed model for the distributed processing and the specific details of the implementations for the two cases of study. The computational efficiency analysis is reported in Section 5. Two examples of studies that generate useful statistics for the population are described in Section 6. Finally, Section 7 presents the conclusions and main lines of future work.

2. Big data processing for Intelligent Transportation Systems in smart cities

This section describes two problems related to processing big data from transportation systems applying distributed computing and computational intelligence.

2.1. Analysis of the quality and equability of the public transportation system

The first case of study proposes combining multiple datasets and performing both off-line and on-line analysis of GPS data and ticket sales information from buses.

Given a big set of data collected from GPS devices and ticket sales machines in buses, the problem consists in computing a number of important statistical values to assess the quality of the public transportation system.

The information collected by GPS devices includes the time and the coordinates for each bus, reported with a frequency of 10–30 seconds, which allow determining the location of each bus within its route. On the other hand, information from ticket sales include the information of every ticket sold on each bus, including: GPS coordinates, time, and date the ticket was sold. Additionally, if the ticket was paid for using a smartcard, a unique identifier for the smartcard is included, which allow identifying trips done by the same passenger.

The main goal of the data processing is to compute relevant metrics to assess the efficiency of the public transport system in Montevideo, for example: *i*) study the impact of traffic conditions and external events on the efficiency of the transportation system *ii*) analyze the real time that each bus takes to reach some important locations in the city (known as *control points* or *remarkable locations*), and *iii*) compute statistical information about the arriving times and delays for each remarkable location (maximum, minimum, mean, mean absolute deviation, and standard deviation).

The information to report must be classified and properly organized in order to determine accurate values according to different days of the week and hours in the day, which imply different passenger demands and different traffic mobility patterns.

The benefits of the proposed system for processing GPS data are twofold: *i*) from the point of view of the users, the system provides useful information from both historical data (monthly, yearly) and the current status of the public transportation in the city, to aid with mobility decisions (e.g., choose a certain bus, move to a different bus stop, consider using a different line); this information can be obtained via intelligent ubiquitous software applications and websites; *ii*) from the point of view of the city administration, the statistical information gathered is useful for planning long-term modifications in the bus routes and frequencies, and also to address specific bottleneck situations in the public transportation system.

A diagram of the proposed system for processing GPS data in ITS is presented in Figure 1. The system is based on buses that upload data reporting their current location (collected by the on-board GPS unit) and ticket sales to a server in the cloud and a historical database which includes transport and socioeconomic data from the past. The server applies big data and streaming analysis techniques to the collected data. The results are then exposed to be consumed by mobile applications for end-users and also to be used in monitoring applications for the city government authorities.

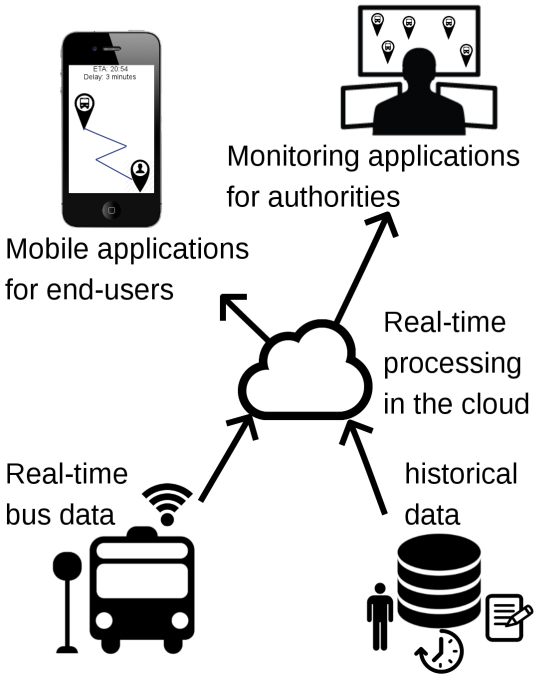

Figure 1. Architecture of the proposed big data analysis for ITS

The proposed system is useful for performing both on-line and off-line data processing. On the one hand, on-line processing applying streaming analysis techniques fulfill requirements in areas such as early warning and detection, and adaptive routing. The off-line analysis on the other hand, is focused on the study of trends that allow identifying mobility patterns and creating predictive models that can be later used in conjunction with sensors that stream data in real-time.

On-line ITS metrics and statistics. In order to handle real-time data the system relies on a streaming processing engine that enables high throughput ingestion from multiple concurrent data sources. The proposed system presents a set of expressive abstractions, such as join, map, and reduce, to apply transformations to the data before persisting the results into the storage subsystem.

Off-line ITS metrics and statistics. For computing off-line mobility metrics, the proposed system demands processing a large volume of data in short execution time, thus leading to a classic big data problem. The system applies a parallel/distributed model to perform the data processing, where the original data is split and distributed across different nodes to be processed independently. Finally, all the partial results from each node are combined to return the final solution. These data are used to derive metrics of the quality of the service that are later merged with socioeconomic indicators that are used to characterize the geographic area served by each particular bus line.

The number of bus lines, bus stops, and individual trips completed every day constitute a relatively large volume of data. Thus, even some of the more traditional geo-spatial analysis problems, such as deriving the isochrones for a fixed walking distance from the stops on a bus line–to determine the bus coverage area–becomes a candidate problem for parallel processing.

The social dimension of the analysis is crucial to evaluate whether the services are being fairly delivered, benefiting all the communities irrespectively from their location or demographic characteristics. The concept of "equitable city" is one of the promises of urban informatics and it is a central premise of our research.

2.2. Estimation of mobility patterns: demand and OD matrices

The second case of study proposes applying distributed computing techniques for estimating mobility patterns and OD matrices in ITS systems.

Origin-Destination (OD) matrices are often not directly observable, because sensors or GPS gadgets in buses typically measure traffic characteristics, which are the result of not just origin-destination trips, but also of route choices and traffic operations for certain types of vehicles. Thus, OD matrices have to be estimated from any available relevant data.

This is a relevant problem for implementing the smart city paradigm. Determining the mobility patterns to build demand and OD matrices is crucial for analyzing the transportation system and the resulting outcomes are key for city administrators to take decisions that improve the quality of the system.

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

Data from GPS, sensors, and traffic gadgets are gathered in many formats and with different granularity. This case of study focuses on a study of the ITS for the city of Montevideo, Uruguay. Thus, the types of input data considered in this article for origin-destination estimation and prediction are the GPS and ticket sales data from buses in Montevideo.

The city government in Montevideo introduced in 2010 an urban mobility plan to redesign and modernize urban transport in the city [5]. 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 boarding 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, as 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.

Using historical information gathered in the context of the STM transport system of Montevideo, the goal is to accurately estimate demand and OD matrices from GPS bus location and ticket sales data (considering tickets payed with and without smartcards). The computed results are of significant value to the authorities at the city government in Montevideo, since there is a serious lack of mobility information. Traditional methods (e.g., passenger surveys, visual inspections) have proven to be expensive and offer outdated information while novel methods based on information already gathered by the ITS have not been explored by the city authorities yet.

3. Related works

This section reviews the related works on the two main topics addressed in this article: applying big data and distributed computing approaches for processing data from ITS and related systems in the context of smart cities, and processing data to estimate demand and OD matrices.

3.1. Distributed computing for processing traffic data

Several articles have proposed applying distributed computing approaches to process large volumes of traffic data with diverse goals. A brief review of related works is presented next.

The advantages of using big data analysis for social transportation have been studied in a thorough manner in the general review of the field by Zheng et al. [6]. The authors analyzed using several sources of information, including vehicle mobility (e.g., GPS coordinates, speed data), pedestrian mobility (e.g., GPS and WiFi signals from mobile devices), incident reports, social networking (e.g., textual posts, address), and web logs (e.g., user identification, comments). In the review, the advantages and limitations of using each source of data are discussed. Several other novel ideas to improve public transportation and implement the ITS paradigm are also reviewed, including applying *crowdsourcing* techniques for collecting and analyzing real-time or near real-time traffic information, and using *data-based agents* for driver assistance and analyzing human behavior. A conclusion on how to integrate all the previous concepts in a data-driven social transportation system that improves traffic safety and efficiency is also presented.

Several other computational intelligence techniques have been recently applied to process ITS data in order to help the decision-making processes in smart cities. Oh et al. [7] proposed a sequential search strategy for traffic state prediction combining a Vehicle Detection System and the k nearest neighbors (kNN) non-parametric method for classification. An experimental evaluation was performed considering data from the performance measurement system from State Route 78 highway in California, United States. The results demonstrated that the proposed system outperformed a traditional kNN approach, computing significantly more accurate results while maintaining good efficiency and stability properties.

Shi and Abdel-Aty [8] applied the random forest data mining technique and Bayesian inference to process large volumes of data from a microwave vehicle detection system, with the main goal of identifying the contributing factors to crashes in real-time. Rear-end crashes were studied because they have a straightforward relation with congestion. The experimental evaluation of the proposed computational intelligence approach was performed considering traffic data from State Routes 408, 417, and 528 in Central Florida, United States. A reliability model was also included in the analysis. The main results allowed the authors to conclude that peak hour, higher volume and lower speed at upstream locations, and high congestion index at downstream detection point significantly increased the probability of crashes.

Ahn et al. [9] applied Support Vector Regression (SVR) and a Bayesian classifier for building a real-time traffic flow prediction system. Data preparation and noise filtering are applied to raw data, and 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 outperformed a traditional linear regression methods in terms of accuracy.

Chen et al. [10] proposed a model that aims to efficiently predict traffic speed on a given location using 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 Map-Reduce 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 used corresponds to the Interstate 5 Higway in San Diego, California, United States. The processed information included 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 was able to accurately predict traffic speed with an average forecasting error smaller that 2 miles per hour. Additionally, a 69% improvement on the execution time was achieved by using the Hadoop framework in a cluster infrastructure when compared with a sequential algorithm running in a single machine.

Xia et al. [11] studied the real-time short-term traffic flow forecasting problem. To solve the problem, the authors proposed using the k nearest neighbor algorithm in a distributed environment, following the Map-Reduce model implemented over the Hadoop framework. The proposed solution considered 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 first 14 days of data are used as the training set and the last day is used for evaluating the computed results. The proposed algorithm allows reducing the mean absolute percentage error by 8.5% to 11.5% on average over three existing techniques based on the kNN algorithm. Additionally, a computational efficiency of 0.84 is reported for the best case.

3.2. Estimation of demand and OD matrices

The estimation of demand and OD matrices is a well-known problem in the field of public transportation. This problem has had a renewed interest with the increasing availability of large volumes of data from modern ITS systems.

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.

An analysis of the literature about using smart cards in ITS was presented by Pelletier et al. [12]. The review covered all the details about hardware and software needed for deploying smart card payment solutions in urban transportation systems. In addition, privacy and legal issues that arise when dealing with smart card data were also reviewed. The authors identified the main uses for smart card data, including: long-term planning, service adjustments, and performance indicators of the transportation systems. Finally, the review described several examples of smart card data utilization around the world.

Trépanier et al. [13] proposed a model for estimating the destination for passengers boarding buses with smart cards, following a database programming approach. Two hypotheses are considered, which are also 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 the previous two assumptions, the authors proposed 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 exists 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 the 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. It is worth noting that most trips for which its destination could not be estimated with the proposed approach 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 real 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. [14] proposed using a trip-chaining method to infer bus passenger origin-destination 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. [13], 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% on the worst case. Finally, two practical applications of the results are presented. The first one consists of studying the daily load/flow variation in order to identify locations along each bus route where passenger load is high, as well as underutilized route segments. The second application consists of a 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.

Later, Munizaga et al. [15] presented a similar approach to the one applied by Wang et al. [14] for estimating OD matrices in the multimodal transportation system of Santiago, Chile. The scenario considered in the article by Munizaga et al. is more general, because passengers can use their smart cards to pay for tickets at metros, buses, and bus stations. The proposed approach is evaluated using smart card datasets corresponding to two different weeks, with over 35 million transactions each. The origin of the trip is accurately determined for nearly every transaction while the destination and time of alighting was estimated for over 80% of the transactions. After extrapolating and post-processing, an estimated OD matrix is presented to visualize the computed results at any given timespace disaggregation.

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.

Early works on this topic applied distributing computing to gather traffic data. Sun [16] 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. [17] 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 [18] 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 unclear.

Huang et al. [19] 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 origin-destination demand estimation is proposed as a line for future work, in order to deal with large-scale traffic networks with huge number of origin-destination pairs and sensors.

3.3. Summary of related works

The analysis of related works allows identifying several proposals for using big data analysis and computational intelligence methods to design improved ITS. Computational intelligence and 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. Furthermore, few works focus in the social justice, integrating into the analysis elements that provide insights into the fairness with which the service is delivered. In this context, the research reported in this article contributes with specific proposals to monitor and improve the public bus transportation, considering the point of view of both users and administrators, and providing objective metrics on the way communities receive these services in Montevideo, Uruguay.

Regarding demand and OD matrices estimation, previous works have addressed the problem of estimating the destination by chaining trips where the destination is assumed to be near the origin of the following trip. This article expands that idea by also considering transfers between bus lines, which are specifically recorded in the smart card dataset used for the evaluation. Additionally, a novel parallel/distributed computing approach is presented to allow solving a more complex and computing-intensive data processing problem. To the best of our knowledge, this approach has not been previously proposed in the related literature.

4. The proposed solutions

This section describes the proposed solutions for the two cases of study presented in this article.

4.1. Processing GPS data from buses in the public transport system

Historical GPS data processing applying Map-Reduce over Hadoop. The first case of study is described next.

Design and architecture. The problem is decomposed in two sub-problems: *i*) *pre-processing* to properly prepare the data to be used as input for the processing in the next phase, and *ii*) *statistics computation* of the public transportation system, using a parallel/distributed approach.

A master-slave parallel model is used to define and organize the control hierarchy and processing. Figure 2 presents a conceptual diagram of the proposed solution.

In the pre-processing phase, the master process filters the data, in order to select only that information that is useful to compute the statistics. The data processing phase applies a data-parallel domain decomposition strategy for parallelization. The available data resulting from the previous phase is split in chunks to be handled by several processing elements. The master process is in charge of controlling and monitoring the system, performing the data partition, and sending the chunks to slaves for processing. Each slave process receives a subset of the data from the master. The group of slaves processes collaborate in the data processing, generating the expected statistical results. Each slave performs the same task; therefore, a single program multiple data (SPMD) parallel model is applied.

Strategy for data processing: algorithmic description. The input data correspond to the GPS coordinates sent by each bus in operation, during each trip. Every line in the input file represents a new position recorded by a certain bus during a given route and for a particular instant of time. The first column of the input file corresponds to the *line number* field, which is a unique identifier for each bus line. In turn, to distinguish different trips of the same bus line, the file has a self-generated numeric field, *trip number*, which identifies a particular trip of a bus line.

Pre-processing stage. The main goal of pre-processing is data preparation. This stage filters input data that do not contain useful information for the statistics to compute, and classifies/orders useful records.

Three phases are identified in the data preparation:

1. *Filtering*: This phase filters the data according to the statistics to compute. Two relevant cases are considered: *i*) *discarding non-useful data*, as the raw data files include information that is not useful for computing the statistics (e.g., when computing the accuracy of buses to reach remarkable locations, the GPS information not related to remarkable locations is not needed); and *ii*) *filtering ranges*, as the system receives a time range as input and computes the statistic for that given period of time.

2. *Time range characterization*: This phase identifies the time range of the timestamp of each record containing useful information. Since traffic patterns vary significantly throughout the day, the time range must be taken into account when computing and analyzing the generated statistics. to compute and analyze statistics generated due to variations of this factor determines very different values to be processed. We consider three time ranges in the study:

   - *morning*, between 04:00 and 12:00,
   - *afternoon*, between 12:01 and 20:00, and
   - *night*, between 20:01 and 4:00.

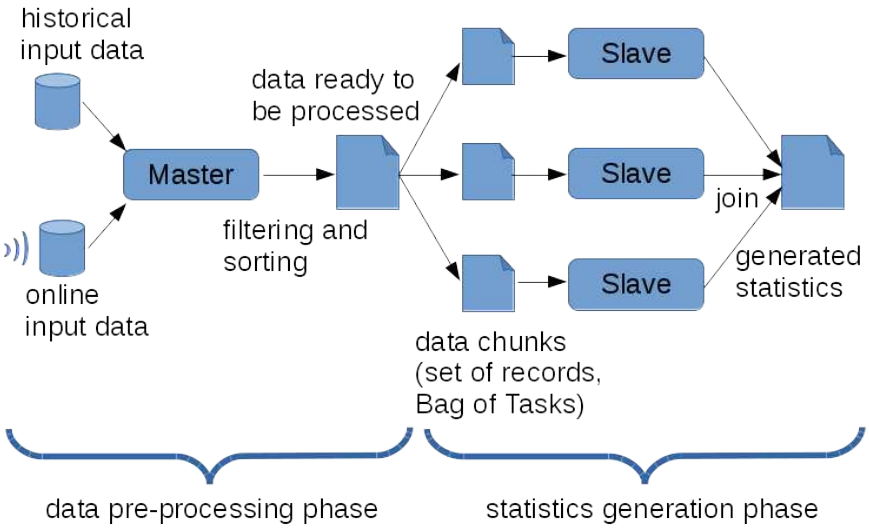

Figure 2. Conceptual diagram of the proposed application

3. *Sorting*: This phase sorts the records according to the bus line identification (*line number*) and the timestamp of the record. These fields define a processing key, which is needed to compute the time differences between the departing time for each bus and the time taken to reach each of the remarkable locations.

After applying the pre-processing stage, the master process has the filtered data to be used as input data for the processing to be performed by each slave process. The data consist in a set of records containing the following fields: *line number*, *trip number*, *timestamp*, *timerange*, and *bus stop*.

Statistics generation stage. The statistics generation stage is organized in four phases:

1. *Data partitioning and distribution*: The master process divides the dataset of useful GPS records and distributes the resulting subsets among the slave processes. Each of the resulting data subsets includes a group of records associated with the same line number, sorted according to the criteria applied in the pre-processing stage. Statistics associated with the same line number are processed in the same slave.
2. *Computing temporal distances*. In this phase, each slave processes the subset assigned by the master process, splitting each record into different fields, to create new data. For each bus line number, the distances between different points on the bus route are calculated iterating through a date-ordered list containing the distance values. The start of each trip is defined by the first new occurrence of a trip number found in the subset containing the GPS data handled by each slave process. Each slave uses that initial time to track each remarkable location or bus stop, by computing the time between the timestamp and the initial time (i.e., the *relative time*). The computed relative times are then filtered by timerange and by remarkable location. The generated results are stored in memory, grouped by the fields mentioned above, to be available for the next phase.
3. *Statistics generation*. In the third phase, data are reduced into results and finally statistics are computed. For each occurrence, an iteration over the calculated distances is performed to compute several metrics, including:
   - the maximum differences between times (*max time difference)*;
   - the minimum differences between times (*min time difference*);
   - the average time (*time average*); and
   - the standard deviation of time values (*time standard deviation*).

   These metrics are computed considering the relative time (accumulated time of the trip from the starting location) to reach each bus stop or remarkable location for each bus line, and filtering by the different time ranges considered. The output values are grouped and ordered by line number, remarkable location, and timerange.
4. *Return results to the master process*. The slaves return the partial results to the master, who groups and prompts the final results to the user.

Implementation details. The proposed parallel/distributed system for traffic data processing is implemented using a Map-Reduce approach in Hadoop. The application fits in the Map-Reduce model because no communications are required between slave processes and the only communications between master and slaves are performed for the initial phase of data distribution and the final phase to report the results.

The Map-Reduce 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. When the slaves finish processing, each *TaskTracker* sends the results back to the *JobTracker* in the master node. The details for each stage are presented next.

Pre-processing stage. The pre-processing stage involves the traditional phases usually applied when using the Hadoop framework: *splitting*, *mapping*, and *shuffling and sorting*. All these tasks are performed by the Hadoop master process:

- *Splitting*. The splitting phase assigns records to the master process. Two instances of the `FileInputFormat` and `RecordReader` classes in Hadoop were implemented to filter useful data and generate the input data to be used by the mapper process. After that, all selected records are converted to appropriate datatypes to be used in the statistics generation stage (e.g., numerical data are converted to `long` or `int`, dates are converted to `timestamp`, etc.).
- *Mapping*. In the mapping phase, each mapper receives a subset of data (*data block*) to process. The number of mappers on execution is defined by X/b, where X is the size of the data to process and b the size of the data blocks.

  Data filtering is applied by each mapper, using `RecordReader<KeyBusCompound,BusInfo>` objects. `KeyBusCompound` and `BusInfo` are used as <key,value> pairs for records sent to mappers. `KeyBusCompound` is a compound key including the fields needed to identify a bus trip (line number, trip number, and timestamp). `BusInfo` includes values for remarkable locations, timerange, trip number, and timestamp, needed to apply a secondary sorting (see *Sorting*).

The Hadoop framework creates a number of filtering instances according to the number of input splits on the input file. Each instance p operates on a data subset $\{L\}_p$, where $\{L\}$ is the set of input lines received in the splitting phase.

Several methods were implemented to define the filtering logic to return the next register to be processed by mappers. For each record on the input, an integrity check is performed to discard records without the expected format and those not included in the range to process. A two-field record <KeyBusCompound, BusInfo> is added to the resulting set for each record not discarded. The output is a list containing $\{T\}_p$ records, with the format <(line_number, trip_number, timestamp), info>.

- *Shuffling and sorting*. Sorting, shuffling, and partitioning are applied after the map stage on a typical Map-Reduce application. The common practice is sorting keys, but we decided to apply a secondary sorting [20] to deliver ordered values to each reducer to compute temporary distances in the proposed system. The secondary sorting is needed to sort both keys and values (i.e., the well-known *value-to-key* conversion procedure).

Statistics generation stage. This stage is performed by Hadoop reduce processes, which correspond to slave processes in the conceptual algorithmic description. Each process has three phases:

- *Data partitioning and distribution*. This phase corresponds to the data sent from map to reduce processes. By default, the Hadoop framework distributes keys to different reducers applying a hashmap partitioning. This distribution mechanism does not guarantee an appropriate load balancing, because reducers do not receive equally-size subsets to process. Furthermore, the results produced by reducers will not be ordered, as it is desirable for the reports to be delivered to the users of the proposed application. For these reasons, we implemented a specific partitioning method using a TreeMap [21] hash, so the reducer sends those records associated to a specific bus line number. The TreeMap structure is dynamically generated in the main program, taking into account the number of reducers and a CSV file containing ordered unique keys (line number).
- *Reduce*. According to the procedure in the previous phase, each reducer receives records with the format: <(line_number$_i$), [info$_{i1}$, ..., info$_{iN}$]>, related to a given bus line number, and all values associated to keys are ordered. A reduce function is executed on each key (line_number) on set {T}: the initial times are determined for each bus trip and the relative times between remarkable locations in the trip are computed. Data is temporarily stored in a TreeMap structure, using KeyStatistics (line_number$_i$, control_point$_i$, timerange$_i$) as key and LongWritable representing the time differences, as values.
- *Statistics generation*. A second function on the reducer receives each <KeyStatistic, LongWritable> pair and computes the statistic values from the previous partial results. In the cases of study reported in this article, we compute the maximum, minimum, arithmetic mean, mean absolute deviation, and standard deviation of times for each bus line number, control point, and timerange. The reducers output is <(line_number$_i$, timerange$_i$, control_point$_i$), (min_time$_i$, max_time$_i$, mean$_i$, mean_deviation$_i$, standard_deviation$_i$)>. These pairs are represented as text, key, and value, to prompt results to user.

Fault tolerance. The proposed implementation applies the automatic fault tolerance mechanism included in Hadoop. Additionally, some features are activated to improve fault tolerance for the ITS application developed: *i*) the feature that allows discarding corrupt input lines is enabled, to be used in those cases where a line cannot be read (the impact of discarding corrupt lines is not significant, because the system is oriented to compute statistics and estimated values); and *ii*) the native replication mechanism in HDFS was activated, to keep data replicated in different processing nodes.

Characterizing the bus service zones using socioeconomic indicators. The main details on the procedures for defining and characterizing the bus service zones are presented next.

Data and methods. This part of the solution relies on a combination of geo-spatial analysis and traditional statistics. The bus service areas are based on isochrones which define equal travel times for a walking distance of ten minutes from each of the stops of a particular bus line. The bus lines and bus stops geometries to derive the isochrones were obtained from the open data available on the Geographic Information System (GIS) site of Montevideo city government [22]. These shapefiles were fetched using Python scripts and manipulated using Python packages including geopandas [23], fiona [24], and shapely [25].

The socioeconomic indicators were obtained from the National Institute of Statistics (Instituto Nacional de Estadísticas, INE) in Uruguay. For the purpose of this study, we used the continuous household survey (Encuesta Contínua de Hogares, ECH) [26]. This is a cross sectional survey, conducted uninterruptedly since 1968, that delivers the official indicators for

employment and income (for both household and individuals). INE also makes available shapefiles for the areas of analysis (the census sections and census segments) [27]. Merging all these datasets and shapefiles we developed a combined geo-spatial and socioeconomic view of each area. Figure 3 illustrates this approach presenting all the census sections and segments with an overlapped bus service area.

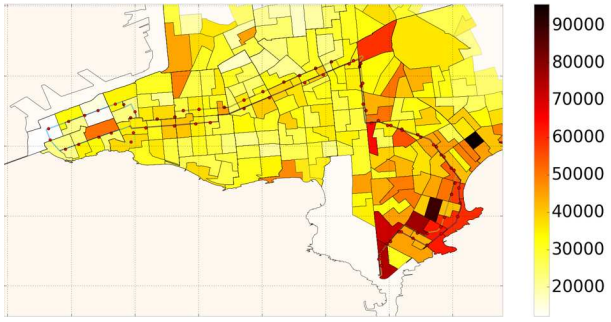

Figure 3. Choropleth map of the serviced area for bus line 185 in Montevideo. The divisions are census segments and the color represents household income (in uruguayan pesos per inhabitant)

The bus line shapefiles are used exclusively to represent the bus trajectory on the maps. All the bus lines geometries are contained on a single shapefile that the GIS publishes. From these files, the two relevant attributes for our analysis were: The bus stops shapefiles, which are used both on the visual representations and as input to derive the isochrones for the bus service zones. The stops are also contained on a single shapefile with similar attributes. In this case, the geometry contains the latitude and longitude for the POINT object that represents the stop. Figure 4 displays the bus lines and the corresponding isochrones of their service areas for the city of Montevideo.

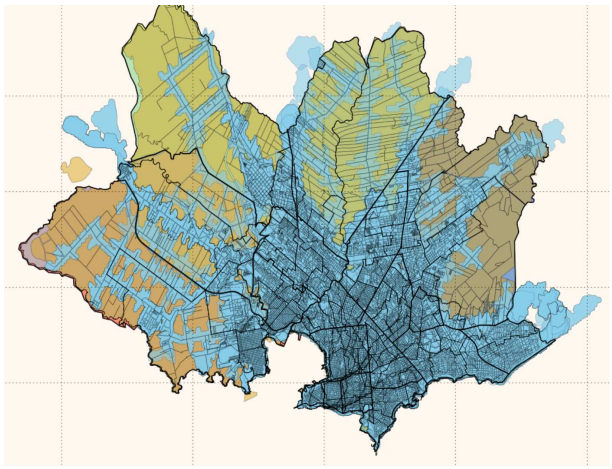

Figure 4. Map of Montevideo with isochrones representing the bus coverage areas (light blue shapes). The map displays large areas of the city outside of the service coverage.

The ECH datasets and shapefiles are accessible from the INE websites [26, 27]. We created local copies of these datasets and used the geopandas package to perform the mapping and statistical manipulations. Basic data wrangling was performed to merge the datasets in order to produce a consolidated geopandas geo dataframe containing both the relevant geometries and the indicators selected for the analysis (geometry, median household income, and number of inhabitants per household). We used the most granular census unit: the census segment. Data extraction, manipulation, and generation of the datasets was scripted using python. We intend to update the computed results yearly, as new data from the ICH is published.

4.2. Mobility patterns and demand/OD matrices estimation

Design and architecture. Initial experiments confirmed that processing only a small portion of the ticket sales dataset requires a large computation time: studying only one month of ticket sales data demands over 18 days when using a sequential algorithm in a regular desktop computer (Intel Core i5 x2 processor with 6 GB RAM and Linux Ubuntu 14.04 operating system). Therefore, applying a parallel/distributed approach is fully justified to reduce the execution times. Our proposal is based on executing the algorithms for demand and OD matrices estimation in parallel, making use of several computing units.

The main idea of the proposed parallel algorithm is to apply a data-parallel approach. The datasets of ticket sales and GPS records are divided in chunks, following the bag-of-tasks paradigm [28]. In this case of study, the bag-of-tasks corresponds to a set of user trips records. Since each set of trip records is independent, as they hold information of different citizens, the bag-of-tasks can be assigned to different slaves for processing.

Using a master-slave model for organizing the processes is an appropriate choice for implementing the estimation algorithm, since the slave processes do not need to share information with each other. A set of slave nodes are created and organized in a *slave pool*, to be used on demand. This decision reduces the overhead of thread creation and destruction, as every thread is used many times while there are records left to be processed.

Initially, the master process collects all the data to be processed and applies a pre-processing stage to filter inconsistent records. After that, the master builds the bags-of-tasks and sends the corresponding bags to each slave in the slave pool, which will perform the assigned computation task. Afterwards, each slave node executes the destination estimation procedure. Finally, the master receives the partial results, persists them, and join them together to create the final demand and OD matrices.

Strategy for data processing: algorithmic description. The main details of the proposed algorithm are presented next.

Data description. The bus companies that operate in Montevideo are required to send bus location and ticket sales data to the city authorities. The bus network in Montevideo is quite complex, including 1383 bus lines and 4718 bus stops. The case study described in this section considers the dataset of ticket sales and bus locations for January 2015, comprising about 200 GB of data.

Bus location data contains 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*, the unique trip identifier for each single trip for a given lineID;
- *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*) that is hashed for privacy purposes, number of transactions for that STM card (*transactionID*), and the last payed transaction (*payedID*). These data allow identifying when a passenger transfers between buses: transactionID increments while payedID remains unchanged. The number of transfers is equal to transactionID−payedID.

Methodology. The proposed methodology for estimating demand and OD matrices takes into account the two kinds of transfer trips existent in Montevideo (detailed in Section 2.2). The proposed model 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 [13–15]. 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. Therefore, two models for estimation are proposed: one for direct trips and one for trips including transfers:

- *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. For each ticket sold, *transactionID* and *payedID*) are recorded. These values 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.
- *Direct trips*. Direct trips are 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. When no bus stop is found on that radius, the procedure is repeated using a larger radius (twice 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.

Estimation algorithm. We propose a specific methodology for reconstructing the trip sequence for passengers, by estimating the destination points from the information available.

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

1. *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 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.

   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.

2. *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. The trip destination is estimated depending on whether it is a transfer or 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 previously described.

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

Two variants of the proposed algorithm were implemented, one for each of the two different estimation procedures presented in Section 2.2. Both variants follow the same general parallel approach previously described. The main implementation details are presented next.

Implementation details. The proposed algorithms were implemented using Python 2.7.5. The cross-platform open-source geographic information system QGIS [29] was used to manage geographic information corresponding to bus location and bus stops data.

The `dispy` [30] software package was used for creating and distributing parallel tasks among several computing 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 defining 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 to execute the computations (e.g., which nodes can execute each computation).

A number of parameters are needed to set a `dispy` cluster, including the program to execute in each node must, the list of nodes available to execute the jobs, and a list of dependencies needed for computation must be specified (in the proposed application there is only one dependency: the availability of the QGIS software).

Once a cluster is created, jobs can be scheduled to execute at a certain node. `dispy` executes the job on an available processor in the defined cluster. After a job finishes, the information about the origin-destination pairs computed is used to build the OD matrix.

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 bag-of-tasks where each task corresponds to all the trip records of a single passenger. Then, each bag-of-tasks 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. Computational efficiency evaluation

This section describes the experimental evaluation of the proposed system for generating statistics of public transportation based on ITS data. The setup for the experimental evaluation is described, including the computational platform used and the problem instances generated from the historical data. After that, the computational efficiency results are reported. Finally, sample studies are presented from the data processed for a specific bus line.

5.1. Platform, instances, and metrics

The main setup for the experimental analysis is described next.

Computational platform. The experimental evaluation was performed over the cloud infrastructure in Cluster FING, the high performance computing facility at Universidad de la República, Uruguay [31]. The analysis was performed using AMD Opteron 6172 Magny Cours (24 cores) processors at 2.26 GHz, 24 GB RAM, and CentOS Linux 5.2 operating system.

Problem instances and data. The problem instances considered in each case of study are described next.

GPS and ticket sales data. Several datasets are used to define different scenarios conceived to test the behavior of the system under diverse situations, including different input file sizes, different time intervals, and using different number of map and reduce processes. We work with datasets containing 10 GB, 20 GB, 30 GB, and 60 GB, and also different time intervals (3 days, and 1, 2, 3, and 6 months), with real GPS data from buses in Montevideo, provided by the local administration *Intendencia de Montevideo*. The input data file to use in each test of the experimental evaluation was stored in HDFS. To better exploit the parallel processing, more mappers than HDFS blocks must be used when splitting the file. Considering an input file of size X MB and HDFS blocks of size Y MB, the algorithm needs using at least X/Y mappers. Hadoop uses the input file size and the number of mappers created to determinate the number of splits on the input file.

Demand and origin-destination matrix estimation. For the experimental analysis of demand and OD matrices estimation, the dataset corresponding to the ITS in Montevideo for January 2015 was processed, including ticket sales and bus location information. This dataset holds the mobility information for over half a million smart cards (corresponding to more than 13 million individual trips). The total size of the dataset is 120GB.

Computational efficiency metrics. Several metrics have been proposed in the related literature to evaluate the performance of parallel and distributed algorithms [32]. In the experimental analysis reported in this article we focus on two traditional metrics for performance evaluation: the *speedup* and the *efficiency*.

The speedup evaluates how much faster a parallel algorithm is compared to its sequential version. It is defined as the ratio of the execution times of the sequential algorithm (T_1) and the parallel version executed on N computing elements (T_N) (Equation 1). The ideal case for a parallel/distributed algorithm is to achieve linear speedup ($S_N = N$). However, the common situation for parallel algorithms is to achieve sublinear speedup ($S_N < N$), due to the times required to communicate and synchronize the parallel/distributed processes. The efficiency is the normalized value of the speedup, regarding the number of computing elements used for execution (Equation 2). The linear speedup corresponds to $E_N = 1$, and in usual situations $E_N < 1$.

$$S_N = \frac{T_1}{T_N} \qquad (1) \qquad\qquad E_N = \frac{S_N}{N} \qquad (2)$$

5.2. Experimental results

The results of the computational efficiency analysis for the two cases of study is presented next.

GPS data processing. We evaluated the computational efficiency of the proposed distributed solution and also the correctness to produce useful information for users and administrators.

Table 1 reports the computational efficiency results for the proposed application when varying the size of the input data (#I), days (#D), number of mapper (#M) and reducer (#R) processes. Mean values computed over five independent executions are reported for each metric. All times are reported in seconds.

Table 1. Results of the experimental analysis: computational efficiency of the proposed Map-Reduce implementation for processing GPS data

#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
10	3	22	22	1333.9	143.0	9.33	0.39
10	30	14	8	2108.6	178.0	11.84	0.49
10	30	22	22	2108.6	187.3	11.26	0.47
20	3	14	8	2449.0	351.1	6.98	0.29
20	3	22	22	2449.0	189.8	12.90	0.54
20	30	14	8	3324.5	275.6	12.06	0.50
20	30	22	22	3324.5	238.8	13.92	0.58
20	60	14	8	4762.0	300.8	15.83	0.66
20	60	22	22	4762.0	264.7	17.99	0.75
30	3	14	8	3588.5	546.9	6.56	0.27
30	3	22	22	3588.5	179.6	19.99	0.83
30	30	14	8	5052.9	359.6	14.05	0.59
30	30	22	22	5052.9	281.1	17.98	0.75
30	60	14	8	5927.9	383.4	15.46	0.64
30	60	22	22	5927.9	311.4	19.04	0.79
30	90	14	8	7536.9	416.6	18.09	0.75
30	90	22	22	7536.9	349.2	21.58	0.90
60	3	14	8	7249.6	944.0	7.68	0.32
60	3	22	22	7249.6	362.1	20.02	0.83
60	60	14	8	10037.1	672.6	14.92	0.62
60	60	22	22	10037.1	531.4	18.89	0.79
60	90	14	8	11941.6	709.6	16.83	0.70
60	90	22	22	11941.6	648.9	18.40	0.77
60	180	14	8	19060.8	913.7	20.86	0.87
60	180	22	22	19060.8	860.3	22.16	0.92

The results in Table 1 indicate that the distributed algorithm allows significantly improving the efficiency of the sequential version, especially when processing large volumes of data. The best speedup value was obtained when processing the 60GB input file: **22.16**, 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 60GB input data file. This efficiency result is crucial to provide a fast response to specific situations and to analyze different metrics and scenarios for both users and administrators.

Figure 5 graphically summarizes the computational efficiency results when using input data files with different size. and Figure 6 when processing records from different numbers of days.

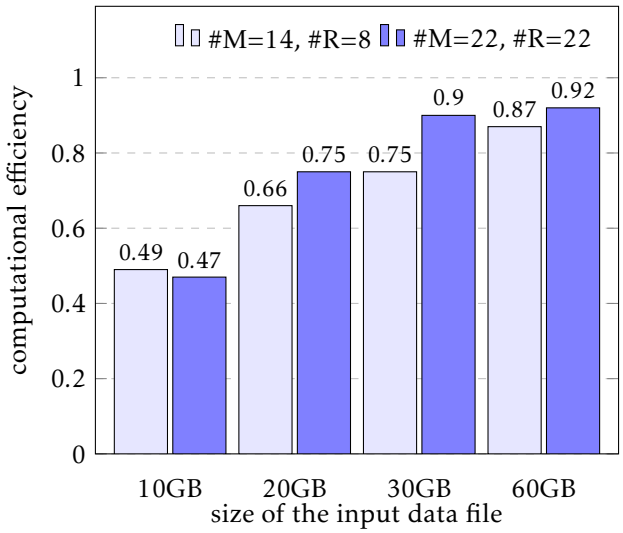

Figure 5. Computational efficiency for different input data files

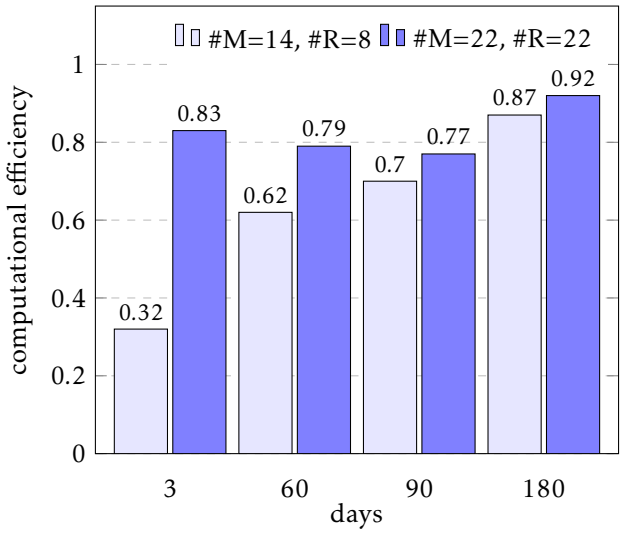

Figure 6. Computational efficiency results for different days

Using 22 mappers and 22 reducers allows obtaining the best efficiency, improving in up to **15**% the execution time (**9**% in average) over the one demanded when using 14 mappers and 8 reducers. Working on small problem instances causes data to be partitioned in small pieces, generating low loaded processes and not improving notably over the execution time of the sequential algorithm.

The efficiency analysis also determines that the Map and Reduce phases have similar execution times and reach the max CPU usage (above 97% at every moment). These results show that the load balance efforts in the proposed algorithm prevents a majority of idle or low-loaded mappers and reducers.

Demand and origin-destination matrix estimation. The proposed master-slave parallel model requires defining the size of the bag-of-tasks assigned to each slave to compute. A proper bag size must be used in order to have an appropriate load balance and avoid excessive communication between the master and the slaves. Experiments were performed varying the size of the bag-of-tasks as well as the number of cores used. The experimental results are reported on Table 2. The number of cores (#cores) and the size of the bag-of-tasks (#bag-of-tasks) 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 #bag-of-tasks.

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 bag-of-tasks 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 bag-of-tasks (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 bag-of-tasks (5000). Further experiments should be performed to assess if using a smaller size for the bag-of-tasks is still more efficient, and to determine the trade-off value before the communications between the slaves and the master become more expensive and have a negative impact on the execution time.

#cores	#bag-of-tasks	*direct trips*			*transfer trips*		
		avg. time±std. dev.	best	speedup	avg. time±std. dev.	best	speedup
1	1	25920.0	25920.0	-	30240.2	30240.2	-
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 2. Execution time results and performance analysis.

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 bag-of-tasks 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. Two sample studies

This section presents two sample studies performed using the proposed distributed system for ITS in Montevideo: average speed/troublesome locations detection and performance of the public transportation across socioeconomic stratas.

6.1. Average speed and troublesome locations

The calculation of the average speed of buses and the analysis of troublesome locations is a relevant study for the public transport in Montevideo.

Figure 7 presents the study of the average speed of the seven bus lines (100, 102, 103, 105, 106, D11, D8, D10 and CA1) traveling through 18 de Julio Avenue (the main avenue in Montevideo) in four relevant time ranges (including peak hours). The speed analysis is a valuable input for decision making in order to improve quality of service and travel experience for users.

Figure 8 presents a report extracted from the analysis of delays of buses to identify troublesome locations in the city. Results correspond to bus line 195 at night. Delay values are computed according to six months of historical GPS records, comparing the times to reach each bus stop against the scheduled times, as reported in the website of STM, Montevideo [5].

These results can be obtained in real time using the distributed algorithm, allowing a fast response to specific problems. In addition, the information can be reported to users via mobile ubiquitous applications.

6.2. Fairness of the public transportation service delivery

The preliminary work in this area allow characterizing each of the bus routes using the median household income for the census segments covered by all the bus lines. Using these indicators the bus line coverage area is defined as unit of analysis, deriving its socioeconomic characteristics from the census segments that are included on it.

Figure 9 shows an example of two different instances of the unit of analysis: the bus service areas for bus

Figure 7. Average speed of buses in 18 de Julio Avenue.

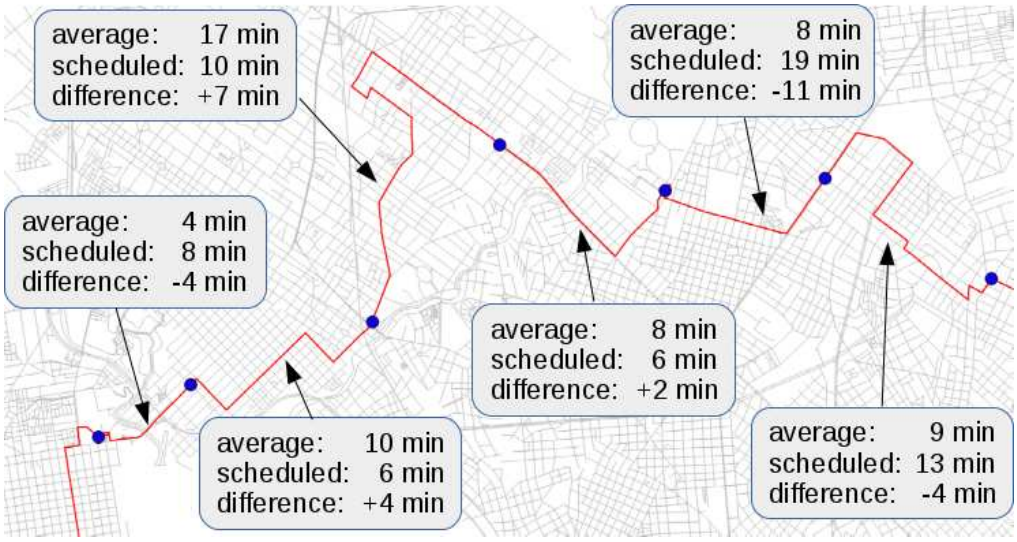

Figure 8. Average delay for bus line 195 in the night, using six months of historical data

lines 121 and 195. The map shows that the service area for bus line 195 covers a significantly larger number of census segments, and its median household income is visibly lower than the average income of the service area for bus line 121. This example illustrates the type of contrasts that exists between different bus service areas across the city, which is worth further studying. The proposed methodology to assess the service fairness involves deriving quality metrics–such as the standard deviations of the total routes duration and the deviation from the original schedules–and measuring the correlation coefficient with the socioeconomic indicators that we use to characterize the service zones, such as the household median income.

Once this phase of the study is completed, we aim at delivering a novel data product that will provide researchers and policy makers with a new perspective on the matter of the fairness of public services delivery. In particular, we will be contributing to answer the question of whether or not certain neighborhoods or areas in the city are dis-proportionally affected by poor public transportation services.

7. Conclusions and future work

This article described our experiences on designing and building a platform for big data analysis for smart cities. This platform combines distributed computational intelligence and geo-spatial analysis to process historical GPS data to compute quality-of-service metrics for the public transportation system in Montevideo, Uruguay. Furthermore, we presented two case studies that rely on the platform capabilities to answer relevant research questions related to two different urban problem domains: operational efficiency and equability.

An intelligent system for data processing was conceived, applying the Map-Reduce paradigm implemented over the Hadoop framework. Specific features were included to deal with the processed data: the

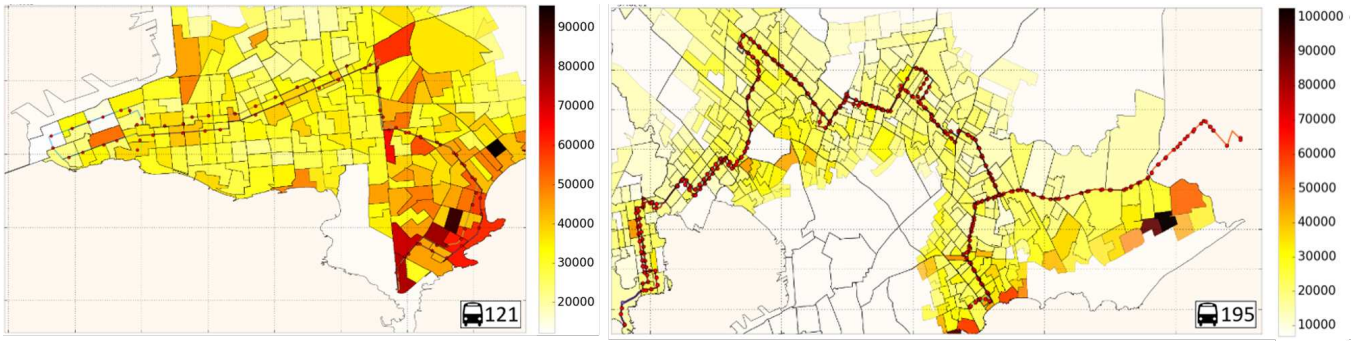

Figure 9. Maps of service areas for buses 121 and 195: The red areas correspond to census segments that display the highest household income (in Uruguayan pesos per inhabitant)

proposed implementation allows filtering and selecting useful information to compute a set of relevant statistics to assess the quality of the public transportation system. An application-oriented load balancing schema was also implemented. Additionally, a method for accurately estimating trips' destination based on smart card data is proposed, based on ideas presented in the related literature. This estimation allows computing demand and OD matrices, which are crucial for transport planning and are difficult to obtain using traditional methods.

The experimental analysis focused on evaluating the computational efficiency and the correctness of the implemented system, working over several scenarios built by using real GPS and ticket sales data collected in 2015 in Montevideo. The datasets comprise over 200 GB of data corresponding to over 1300 line services operating in the city. The main results indicated that the proposed solution scales properly when processing large volumes of input data, achieving a speedup of **22.16** when using 24 computing resources, when processing the largest input files. Regarding demand and OD matrices estimation, the experimental results suggest that the proposed platform is appropriate to increase efficiency, achieving speed up values of up to **16.41** when using 24 computing resources.

As examples, we computed two types of metrics that provide insights relevant to both citizens and decision makers. One is a collection of average speeds for different segments of bus lines in Montevideo using the available historical data. These averages allow to identify troublesome locations in the public bus network, based on the delay and deviation of the times to reach each bus stop. The second type of metrics are related to the bus routes service quality in relation to the socioeconomic characteristics of their service areas. Both studies aim at providing authorities and policy makers with a better understand of the transportation system infrastructure. Some of these insights can also be incorporated in mobile applications that might help improving the travel experience of the general population.

The research reported in this article is based on processing the bus GPS and ticket sales data gathered in 2015. However, the proposed distributed architecture would scale up efficiently when processing larger volumes of data, as shown in the experimental analysis. The city government collects the bus GPS and ticket sales data periodically, so it is possible to incorporate additional data in order to get even more accurate statistics. Furthermore, the Uruguayan government handles several other ITS and non-ITS data sources (including GPS data for taxis, mobile phone data, ticket sale data, special events in the city) which could be easily incorporated to the proposed model to get a holistic understanding of mobility in the city.

The main lines for future work are oriented to further extend the proposed system, including the calculation of several other important indicators and statistics to assess the quality of the public transportation. Relevant issues to include are the construction of OD matrices for public transport, the evaluation of bus frequencies (and dynamic adjustment), etc. The proposed approach can also be extended to provide efficient solutions to other smart city problems (e.g., pedestrian and vehicle fleets mobility, energy consumption, and others). Using other distributed computation frameworks (such as Apache Storm) is also a promising idea to better exploit the real-time features of the proposed system.

References

[1] Deakin, M. and Waer, H. (2012) *From Intelligent to Smart Cities* (Taylor & Francis).

[2] Sussman, J. (2005) *Perspectives on Intelligent Transportation Systems (ITS)* (Springer Science + Business Media).

[3] Massobrio, R., Pías, A., Vázquez, N. and Nesmachnow, S. (2016) Map-Reduce for Processing GPS Data from Public Transport in Montevideo, Uruguay. In *2nd Argentinian Symposium on Big Data*: 41–54.

[4] Fabbiani, E., Vidal, P., Massobrio, R. and Nesmachnow, S. (2016) Distributed big data for demand estimation in its. In *High Performance Computing Latin America, Communications in Computer and Information Science, Springer* **697**: 146–160.

[5] Intendencia de Montevideo (2010), Plan de movilidad urbana: hacia un sistema de movilidad accesible, democrático y eficiente.

[6] Zheng, X., Chen, W., Wang, P., Shen, D., Chen, S., Wang, X., Zhang, Q. *et al.* (2016) Big data for social transportation. *IEEE Transactions on Intelligent Transportation Systems* **17**(3): 620–630.

[7] Oh, S., Byon, Y. and 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.

[8] Shi, Q. and 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.

[9] Ahn, J., Ko, E. and Kim, E.Y. (2016) Highway traffic flow prediction using support vector regression and bayesian classifier. In *International Conference on Big Data and Smart Computing*: 239–244.

[10] Chen, X., Pao, H. and Lee, Y. (2014) Efficient traffic speed forecasting based on massive heterogenous historical data. In *IEEE International Conference on Big Data*: 10–17.

[11] Xia, D., Wang, B., Li, H., Li, Y. and Zhang, Z. (2016) A distributed spatial-temporal weighted model on mapreduce for short-term traffic flow forecasting. *Neurocomputing* **179**: 246–263.

[12] Pelletier, M., Trépanier, M. and Morency, C. (2011) Smart card data use in public transit: A literature review. *Transportation Research Part C: Emerging Technologies* **19**(4): 557–568.

[13] Trépanier, M., Tranchant, N. and 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.

[14] Wang, W., Attanucci, J. and Wilson, N. (2011) Bus passenger origin-destination estimation and related analyses using automated data collection systems. *Journal of Public Transportation* **14**(4): 131–150.

[15] Munizaga, M. and 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.

[16] Sun, C. *Dynamic Origin/Destination Estimation Using True Section Densities*. Tech. Rep. UCB-ITS-PRR-2000-5, University of California, Berkeley.

[17] Toole, J., Colak, S., Sturt, B., Alexander, L., Evsukoff, A. and González, M. (2015) The path most traveled: Travel demand estimation using big data resources. *Transportation Research Part C: Emerging Technologies* **58**: 162–177.

[18] Mellegård, E. (2011) *Obtaining Origin/Destination-matrices from cellular network data*. Master's thesis, Chalmers University of Technology.

[19] Huang, E., Antoniou, C., Lopes, J., Wen, Y. and Ben-Akiva, M. (2010) Accelerated on-line calibration of dynamic traffic assignment using distributed stochastic gradient approximation. In *13th International IEEE Conference on Intelligent Transportation Systems*: 1166–1171.

[20] Parsian, M. (2015) *Data Algorithms, Recipes for Scaling Up with Hadoop and Spark* (O'Reilly Media).

[21] Johnson, B. and Shneiderman, B. (1991) Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the 2nd Conference on Visualization*: 284–291.

[22] Intendencia de Montevideo (2017), Sistema de información geográfica. capas de informacion vial, http://sig.montevideo.gub.uy/. Accessed: April 2017.

[23] GeoPandas developers (2017), GeoPandas. URL http://geopandas.org. Accessed: April 2017.

[24] Gillies, S. *et al.* (2011), Fiona is OGR's neat, nimble, no-nonsense API. URL https://github.com/Toblerity/Fiona. Accessed: April 2017.

[25] Gillies, S. *et al.* (2007), Shapely: manipulation and analysis of geometric objects. URL https://github.com/Toblerity/Shapely. Accessed: April 2017.

[26] Instituto Nacional de Estadística (2015), Encuesta continua de hogares, http://www.ine.gub.uy/web/guest/encuesta-continua-de-hogares1. Accessed: April 2017.

[27] Instituto Nacional de Estadística (2011), Mapas vectoriales, http://www.ine.gub.uy/web/guest/338/. Accessed: April 2017.

[28] Cirne, W., Brasileiro, F., Sauvé, J., Andrade, N., Paranhos, D. and Santos-Neto, E. (2003) Grid computing for bag of tasks applications. In *Proceedings of the 3rd IFIP Conference on E-Commerce, E-Business and EGovernment*.

[29] QGIS Development Team (2009) *QGIS Geographic Information System*, Open Source Geospatial Foundation. URL http://qgis.osgeo.org. Accessed: April 2017.

[30] Pemmasani, G., dispy: Distributed and parallel computing with/for python, http://dispy.sourceforge.net/. Accessed July 2016.

[31] 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**: 12–15.

[32] Foster, I. (1995) *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering* (Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.).