Using the MapReduce Approach for the Spatio-Temporal Data Analytics in Road Traffic Crowdsensing Application

Sandhya Armoogum(✉) and Shevam Munchetty-Chendriah

School of Innovative Technologies and Engineering,
University of Technology Mauritius,
La-Tour Koenig, Pointe-aux-Sables, Mauritius
asandya@umail.utm.ac.mu, shevam.mc@hotmail.com

Abstract. Crowdsensing applications are becoming more popular with time. In this work, we present a crowdsensing application for capturing road traffic information to help citizens to get real-time traffic condition. Such real-time information can be beneficial for citizens to plan their journeys. However, crowdsensing in this specific case, generates spatio-temporal data collected from numerous users; storing and processing such data in real-time can be quite challenging. The MapReduce programming approach has been proposed for processing data in this context. The MapReduce jobs used to process and analyze the data captured from the crowdsensing application are presented as well as the design of the crowdsensing application. Implementation of the MapReduce jobs proposed shows that data can be effectively processed and analyzed to present near real-time information about the road traffic flow while at the same time discarding used data which is no longer required.

Keywords: Crowdsensing · Big data · MapReduce · Spatio-temporal data
Data analysis

1 Introduction

In recent years, due to the affordable cost and availability of smart phones, the latter have become a ubiquitous part of everyday life, even in developing countries. Although, mobile phones are primarily used for communication, social networking, and web browsing; smart phones are also equipped with a plethora of sensors such as accelerometer, gyroscope, GPS, magnetometer, barometer, temperature sensor, humidity sensor, proximity sensor, digital compass, and ambient light sensor, as well as camera, and microphone. The sensing capabilities of smart phones allows the collection of a diverse range of data in the surrounding environment of the user. Given, that people almost always have their phones with them and are constantly interacting with them, they can thus capture real-time information (e.g. weather information, traffic information, street steepness, pavement type, noise levels, pollution) which can be aggregated and presented so as to be of use to other people [1, 2]. This people-centric sensing, commonly known as crowdsensing or mobile crowdsensing have opened up a world of opportunities for new applications that can have a huge societal impact [3].

In [4], five categories of crowdsensing can be distinguished as shown in Table 1. The first two categories, namely participatory crowdsensing and opportunistic crowdsensing, is based on the involvement of the user in the crowdsensing process. Participatory crowdsensing is when users willingly interacting with the application to send data while in opportunistic crowdsensing data is captured and sent automatically with minimal user involvement. The remaining three categories is based on the type of data which is sensed: Environmental (e.g. temperature, air pollution), Infrastructure (e.g. traffic congestion), and Social (e.g. popular TV shows). To engage participation, the application needs to provide incentives for users to share data [5, 6, 7, 16]. In many cases, users participate so as to receive sophisticated services based on the collective statistical information mined from the data.

Table 1. Crowdsensing typology [4].

Involvement of the user in the crowdsensing process	Type of measured phenomenon
Participatory crowdsensing	Environmental crowdsensing
Opportunistic crowdsensing	Infrastructure crowdsensing
	Social crowdsensing

Eventually, with large amount of user participation, crowdsensing applications collect a large volume and variety of data in real-time continuously. The efficient data collection and storage, data processing, analytics, and security is non-trivial in such cases. This is no different from big data collected from various sensors in an IoT system. Data can be aggregated and processed using analytics to provide services and results of new dimensions. The crowdsensing paradigm has pushed analytic possibilities even further given the broad range of applications including traffic planning, weather monitoring, price-dispersion monitoring, and public safety [12]. The knowledge acquired can not only enhance people's day-to-day life but also help utility providers, healthcare providers and the social sphere [13]. In this work, a mobile crowdsensing based application is used for collection, aggregation, and analytics of real-time spatio-temporal data regarding road traffic flow. The mobile crowdsensing application allows users to push their location information as they travel. Such location information provided by commuting users, allows the system to capture average speed of flow of traffic along different routes which can be used to dynamically inform the population at large about the road congestion levels.

Spatial data (e.g. geographic data) is data pertaining to the location and geometry (space). Spatio-temporal data is spatial data which is time-varying in nature [8]. In [9], the authors define a spatio-temporal object as being one which has at least one spatial (e.g. location) and one temporal (e.g. timestamp) property. Typical examples of spatio-temporal objects include moving vehicles, spreading forest fire, earth quake, hurricane, and weather. With technological advances, large amount of spatio-temporal data is now available and it is becoming increasingly important to perform spatio-temporal data analysis and data mining in domains such as meteorology, biology, crop sciences, forestry, medicine, geophysics, ecology, and transportation management [10]. Moving objects (other than cars, trucks, buses, and train) are also more and more frequently connected such as drones, autonomous vehicles, and wireless sensors. However, representing, processing, analyzing, and mining of spatio-temporal datasets can be quite challenging due to the complex structure of spatiotemporal objects and the relationships among them in both spatial and temporal dimensions [11]. In this work, we investigate on the use of the MapReduce programming model to process and analyze spatio-temporal data captured by a crowdsensing application for traffic monitoring.

The paper is organized as follows. In Sect. 2, we present a mobile crowdsensing application for capturing spatio-temporal data from moving vehicles. In Sect. 3, we describe how the real-time continuous data is stored and processed using the MapReduce approach. Section 4 presents some results obtained and Sect. 5 concludes the paper and discusses the future work arising from this research.

2 Crowdsensing Application Design

There are two main architectures which are normally used for crowdsensing applications, namely, the client–server architecture, and the peer-to-peer architecture. Figure 1 below depicts the client–server architecture, where mobile users push information towards the server or requests information from the server. The server collects data from numerous users, perform processing, and responds to users' requests.

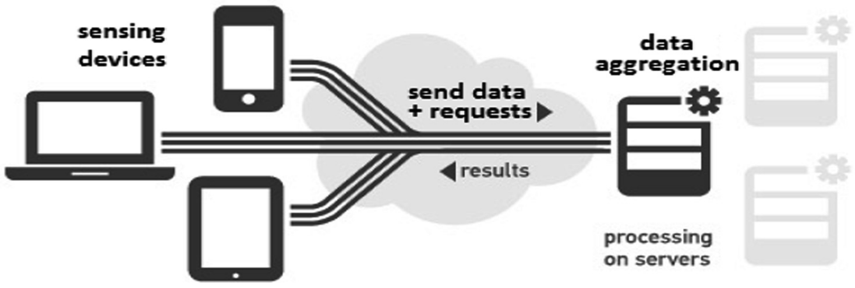


Fig. 1. Client-server crowdsensing architecture.

In the peer-to-peer architecture, sensing devices are inter-connected and share resources among themselves; there is no centralized server. Storage and computational functions are performed on the devices. An example of a sensing application using peer-to-peer architecture is vehicular application used to collect traffic information from all surrounding vehicles and determining the best route.

The client-server architecture has been adopted in this work for the mobile crowdsensing application. According to [14], mobile crowdsensing applications often involves the following six main processes:

1. User Registration: The user registration process is a one-time process enabling data providers (users) to register to the service and give details about themselves for identification. The process is usually performed during the first use of the

crowdsensing application. This process can be by-passed when acquiring the application using Play Store (Android) or App Store (iOS) since the latter has its own registration process which is inherited by the downloaded application.

2. Service Request: This process is performed by the end-user whereby the mobile phone requests a particular service from the service provider (or other mobile phones if using a peer-to-peer architecture). For instance, a request is sent to obtain road traffic details for the different routes available from a source location to a destination.
3. Data Collection: The mobile phones collect data using embedded sensors such as light sensor, temperature sensor, GPS, digital compass, camera, accelerometer, microphone, and Bluetooth as proximity sensor. The captured data is sent to a centralized server or other mobile phones depending on the application architecture for aggregation and processing. This process may be performed on periodically, continuously, on request, or when a certain condition is met.
4. Data Aggregation: The system, receiving the collected data, aggregates the latter with all the other data collected. The centralized approach is used for aggregation whereby all nodes send information to a single centralized server, or mobile phone requesting data. Data is usually aggregated in chronological order [15].
5. Data Processing: The aggregated data is processed using advanced analytic techniques to produce meaningful results and respond to queries. Raw data may require pre-processing tasks such as conversion, filtering, and cleaning.
6. Service Reply: In the case of client-server architecture, the server sends processed information to clients (mobile devices) depending on requests. This process may not be applicable to P2P architecture since processing may be performed by the mobile device itself.

The six main processes are implemented in the proposed crowdsensing application. When the application starts, the user sends a service request to the server. The service request is the starting point whereby the user provide details about the new trip in order to have congestion details about the available routes and hence be able to choose the best one. The application performs opportunistic crowdsensing whereby the mobile phone uses its' inbuilt GPS to capture the location of the user and sends this data to the server with minimal involvement of the user. Data collected from many other users in the same region is aggregated and processed to calculate the level of congestion of routes by calculating the average time taken to travel from one point to another. Such traffic information is the service reply sent in response to the service request. The mobile application was implemented using Android Studio, Google Maps, and Google Directions API on a smartphone with GPS. Figure 2 depicts the use case diagram of the application.

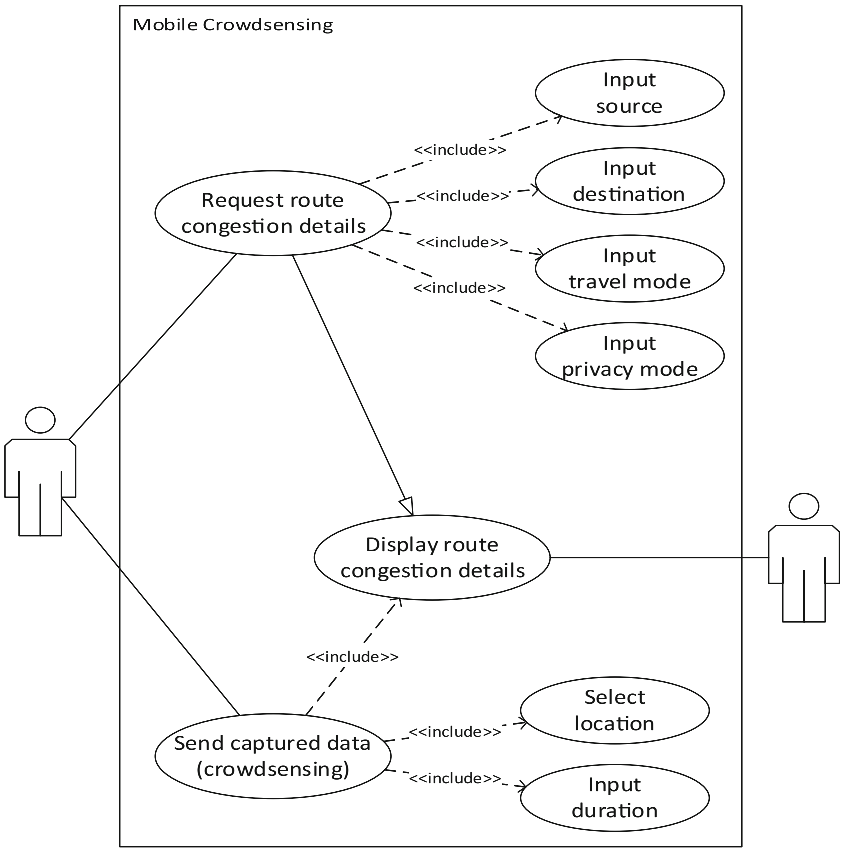


Fig. 2. Client-server crowdsensing architecture.

3 Data Collection, Processing, and Analytics

Depending on the number of users sending the data to the server, the server typically receives a large amount of data continuously. The crowdsensing mobile application may be programmed to send location information every 30 s or every few minutes. However, the crowdsensing application was designed to send data only when the user passes through a predefined point along the travel route specified by the user. This design choice is made such that the crowdsensing application uses less power and do not cause the battery of the smart phone to drain quickly. Moreover, for a higher adoption of the application, the mobile app is designed to send minimal amount of data to the server (identifier, location, time) as it passes predefined points along the route so as to reduce the data network cost for the transmission of data.

As such, there are predefined coordinates on the map along various routes and the user only send data to the server whenever the user reaches these points along the route. The data analytics part in this particular context is quite simple when considering the data provided by one user: by computing the time taken to travel from one point to the next along a route, the server is able to determine the average traffic flow and identify congestion zones (i.e. stretches where the traffic flow is slow). The higher the number of users feeding data, the more accurate the traffic condition can be estimated. However, it also becomes more challenging to keep track of data from different numerous users moving along different routes in real time. Data processing in this context involves real time or near real time analytics i.e. the data has to be analyzed as soon as it arrives so as to have an updated view of the traffic condition. In other applications, data may be stored and batch processed at a later time. But in applications such as the monitoring of road traffic, climate, seismic activity, data has to be processed as soon as it is received by the server [18, 19]. Another important challenge is storing the large volume of data. In the context of road traffic monitoring, we argue that once data has been aggregated, the data contributed by each user may not be useful at a later stage. Thus, the user data are discarded once they have been aggregated. Applying, the traditional programming model to process the data captured from the crowdsensing application for traffic monitoring was found to be difficult and not efficient.

Several big data analytics frameworks have been proposed in literature to address the storage, processing, and analytics of large volume of data. In [17], the authors distinguish between batch based processing technologies and technologies based on stream processing of big data. Batch processing technologies identified include Apache Hadoop, Jaspersoft, Dryad, Pentaho, Tableau, and Karmasphere. Some existing stream based big data processing technologies are Storm, Splunk, S4, SAP Hana, SQLstream s-Server, and Apache Kafka. Depending on the context at hand, different researchers have been using different tools for big data processing. In this work, we attempt to investigate the use of the MapReduce programming model to process the large volume of spatio-temporal data collected in real-time from the crowdsensing mobile application. MapReduce is a programming model for batch processing large datasets and is widely used in large-scale and data-intensive applications. This programming model is simple yet expressive and can be adapted to many real-world tasks. Through the two MapReduce functions, map () and reduce (), it is possible to perform many data analytical tasks e.g. data mining, and machine learning. Furthermore, the MapReduce programming model is capable of handling different types of data and is independent of the underlying storage system [20]. The map function typically processes key/value pairs to generate some intermediate data which can then be further processed by the reduce function which usually merges all intermediate values associated with a key. MapReduce has been specifically formulated such that the programs implementing the map and reduce functions can be executed in parallel on a large number computer for efficiency especially when processing big data. In this work, an Oracle database and the MapReduce programming model is proposed for processing the resulting spatio-temporal data from the crowdsensing application. MapReduce, despite being a batch processing framework, is deemed to be appropriate in this context, since it required to collect at least two readings regarding two points in a route to be able to calculate average speed of traffic thus it makes no sense to process each user data in a stream fashion. The Oracle database has been chosen for several reasons namely, its performance, stability, robustness in handling big data, and its ability to manage multiple databases within the same transaction. The database schema is shown in Fig. 3.

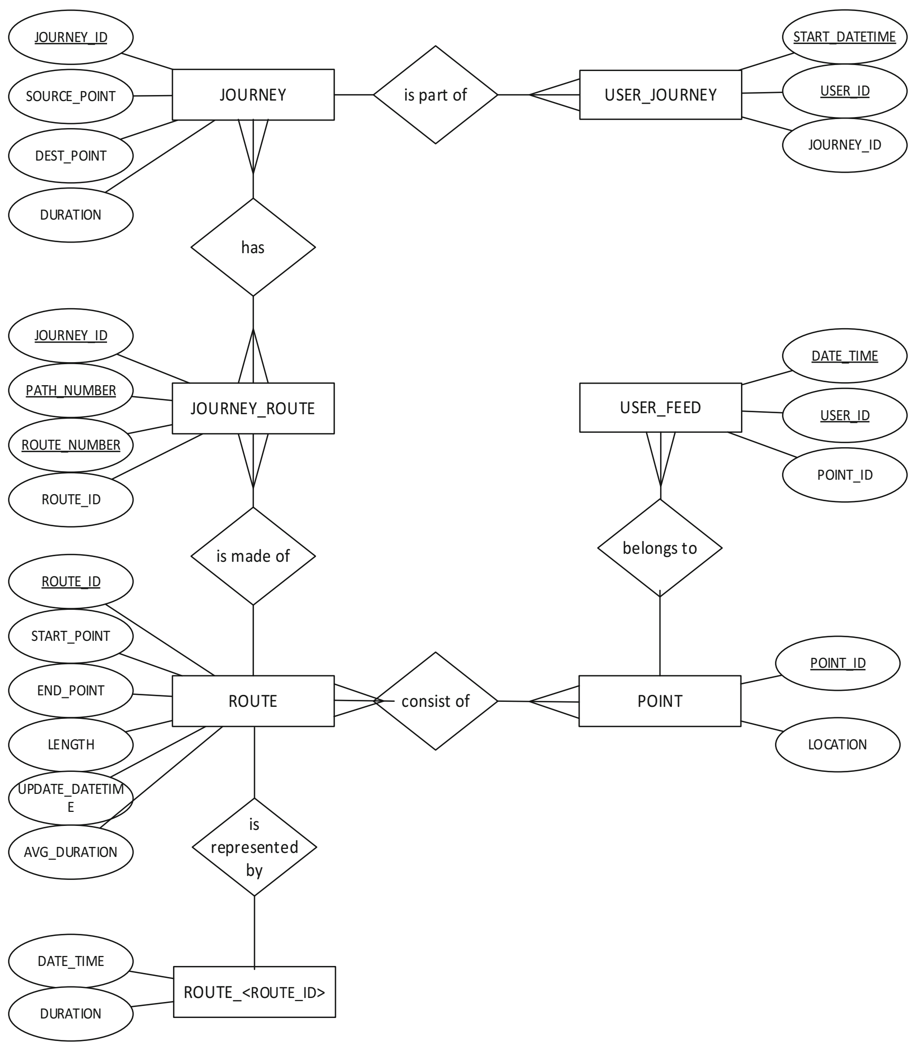


Fig. 3. Entity-Relationship Diagram (ERD) of the database.

Raw user_feeds when received are added as a new row in a table USER_FEED. The user_feeds involve both spatial (location) and temporal (date & time) data. To be able to determine the traffic flow, it is required to determine the time taken by users to travel from one point to the next in near real-time. Two MapReduce jobs are thus proposed to handle the task at hand. The first MapReduce job loops through every record in the table raw user_feeds (if any) and for each record, it converts the data into key/value pair, the key being the User ID and value being the Point ID concatenated with the date and time information. These key/value pairs are then sorted and merged so as to extract the data provided by different users (key User ID). The MapReduce job

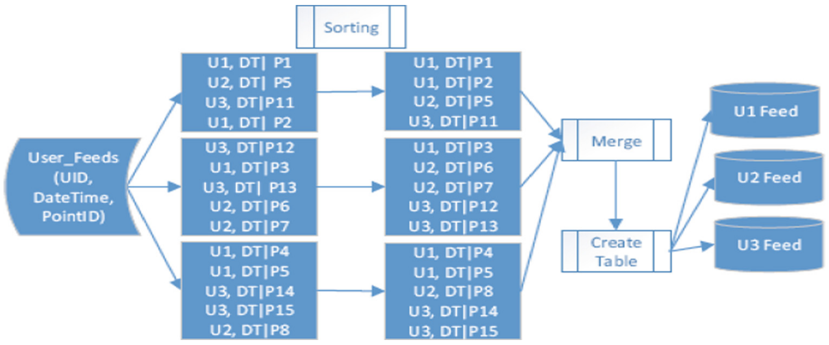


Fig. 4. MapReduce job for processing user_feed.

will automatically execute every few seconds on the database. Figure 4 depicts the MapReduce job for processing the raw user_feeds.

The second MapReduce job takes the output of the first MapReduce job (i.e. inputs from different users) and sorts the data in chronological order for each user. The information from several users are processed and data for the same route are pulled together such that the average time taken to travel from one point to the next along a path is calculated. Finally, the current average time taken for a route being traversed (defined by a starting point and an ending point) is calculated and stored in the appropriate row in the ROUTE table. Several different journeys may consist of common routes (a journey itself, consist of a number of routes). The time taken for a particular journey is computed by summing the time taken along every route that the journey consists of. In response to Service Request, the journey information is retrieved and sent to the user. Data from the user_feed that have already been processed are discarded so that every time fresh data are used to calculate the traffic flow in near real-time. Using the distance information between the two points, the average speed along the route can be calculated in real-time. Similarly, using the average time taken for each route on a journey, the time take to reach the destination can be estimated. Figure 5 depicts the second MapReduce jobs' data flow diagram.

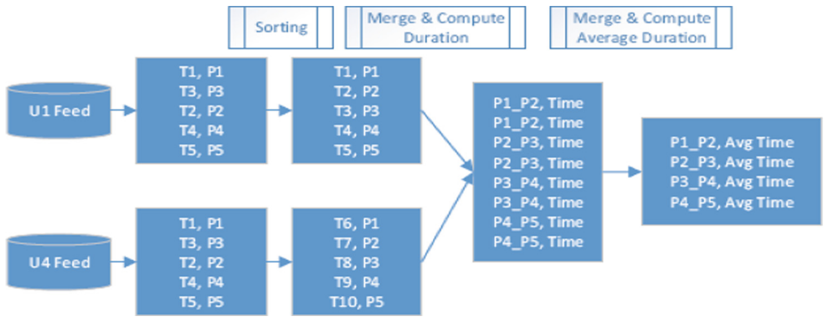


Fig. 5. MapReduce job for processing route duration.

4 Implementation and Results

A simple mobile application was developed to implement the crowdsensing for capturing road traffic information on the motorway M1 of Mauritius and gather some sample data that can be collected in real-time. However, since, it is beyond the scope of this work to deploy the mobile application on a large number of smart phones of users travelling at different points in time, a program (Data Generator) has been designed to simulate and generate input for a number of users. Figure 6 below shows the data processing top-down design on the server side. The processes to be performed by the server have been broken down into 5 sub-modules as shown in Fig. 6.

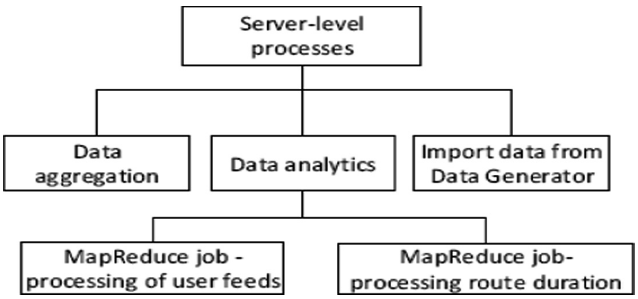
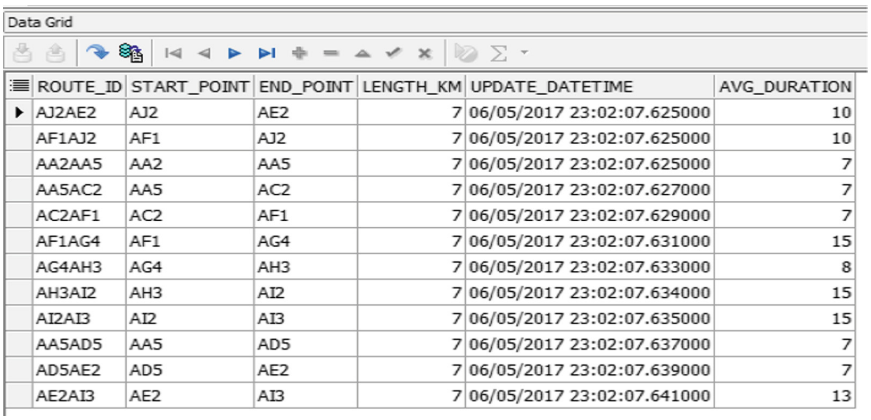

Server-level processes
Data aggregation
Data analytics
Import data from Data Generator
MapReduce job - processing of user feeds
MapReduce job- processing route duration

Fig. 6. Server top-down design.

The details for each sub-module are listed below:

- Data aggregation: This module is responsible in inserting new journeys (table user_journey) and crowdsensing data (table user_feeds) in their respective tables for processing.
- Import data from Data Generator: Retrieve data from the flat file created by the Crowdsensing Data Generator and insert the records in their respective tables for processing.
- MapReduce job – processing user feeds: All unprocessed records are arranged and placed in tables corresponding to each user.
- MapReduce job – processing route duration: The average duration for each route are calculated and stored/updated in the routes_duration field. For instance, all crowdsensing data for point AA2 to point AA5 are sorted and merged to compute the duration.

The data generator was designed to generate data from a random number of users (program takes as input a random number representing the number of users). The data generator also chooses a path randomly for each user and sets the start time to be different. The data feed generated for the users were output to a file for import to the database. However, the generator program was limited to generating data for a limited number of journeys for testing. Google Maps was used to display the traffic flow using color codes. A sample of the route table that was obtained is shown in Fig. 7.



ROUTE_ID	START_POINT	END_POINT	LENGTH_KM	UPDATE_DATETIME	AVG_DURATION
AJ2AE2	AJ2	AE2	7	06/05/2017 23:02:07.625000	10
AF1AJ2	AF1	AJ2	7	06/05/2017 23:02:07.625000	10
AA2AA5	AA2	AA5	7	06/05/2017 23:02:07.625000	7
AA5AC2	AA5	AC2	7	06/05/2017 23:02:07.627000	7
AC2AF1	AC2	AF1	7	06/05/2017 23:02:07.629000	7
AF1AG4	AF1	AG4	7	06/05/2017 23:02:07.631000	15
AG4AH3	AG4	AH3	7	06/05/2017 23:02:07.633000	8
AH3AI2	AH3	AI2	7	06/05/2017 23:02:07.634000	15
AI2AI3	AI2	AI3	7	06/05/2017 23:02:07.635000	15
AA5AD5	AA5	AD5	7	06/05/2017 23:02:07.637000	7
AD5AE2	AD5	AE2	7	06/05/2017 23:02:07.639000	7
AE2AI3	AE2	AI3	7	06/05/2017 23:02:07.641000	13

Fig. 7. Sample data from Route Table showing average time taken in minutes.

5 Conclusion and Future Work

In this work, the MapReduce programming model have been used to process and analyze data captured in real-time from moving objects via a crowdsensing application. Such real-time information can be beneficial for citizens to plan their journeys. The proposed data analytics effectively processes spatio-temporal data to determine the average traffic flow along a route in real-time. Used data are discarded so that the system does not end up with massive data tables after a few days. In this crowdsensing application, the data captured is only being processed to calculate the traffic flow, however such an application's data, when collected for a certain period of time, can be for predicting traffic flow and evaluating fuel efficient routes.

Future work includes implementing a web interface to capture other data about the journey e.g. rain, accident, working day, bank holiday and to extend the data analytics on the data captured to predict traffic flow in the long run as well as using currently captured data to find actual traffic flow. Moreover, for such crowdsensing applications to be successful, user participation is important. It is also being planned to investigate on possible incentives for such crowdsensing applications to ensure high adoption. Finally, when users provide traffic information, there are privacy concerns, unless the application is to be used for tracking vehicle movement by an organization. Further work to study and assess the privacy issues is also planned.

References

1. Rodrigues, J.G., Aguiar, A., Barros, J.: SenseMyCity: crowdsourcing an urban sensor. arXiv preprint arXiv:1412.2070 (2014)
2. Campbell, A.T., et al.: The rise of people-centric sensing. IEEE Internet Comput. **12**, 12–21 (2008)
3. Ganti, K., Ye, F., Lei, H.: Mobile crowdsensing: current state and future challenges. IEEE Commun. Mag. **49**, 32–39 (2011)

4. InfoSec Institute: Crowdsensing: state of the art and privacy aspects, July 2014. http://resources.infosecinstitute.com/crowdsensing-state-art-privacy-aspects/
5. Lee, J., Hoh, B.: Sell your experiences: a market mechanism based incentive for participatory sensing. In: Proceedings of IEEE PerCom 2012, Manheim, Germany (2010)
6. Tham, C., Luo, T.: Quality of contributed service and market equilibrium for participatory sensing. IEEE Trans. Mob. Comput. **14**(4), 829–842 (2015)
7. Yang, D., Xue, G., Fang, X.: Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing. In: Proceedings of ACM MobiCom 2012, Istanbul, Turkey (2012)
8. Yang, H., Parthasarathy, S.: Mining spatial and spatio-temporal patterns in scientific data. In: Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW 2006), p. 146 (2006)
9. Venkateswara Rao, K., Govardhan, A., Chalapati Rao, K.V.: Spatiotemporal data mining: issues, tasks and applications. Int. J. Comput. Sci. Eng. Surv. (IJCSES) **3**(1), 39 (2012)
10. Shekhar, S., et al.: Spatiotemporal data mining: a computational perspective. ISPRS Int. J. Geo-Inf. **4**, 2306–2338 (2015). https://doi.org/10.3390/ijgi4042306
11. Bogorny, V., Shekhar, S.: Spatial and spatio-temporal data mining. In: The Proceedings of the IEEE 10th International Conference on Data Mining (ICDM), Sydney, NSW, Australia (2010)
12. Pallavi, A.R., Annapurna, V.K.: Enforcing security for smartphone user by crowdsourcing model using internet of things. Int. J. Adv. Res. Comput. Sci. Technol. (IJARCST 2016) **4**(2), 1217 (2016)
13. Gilbert, P., Cox, L.P., Jung, J., Wetherall, D.: Toward trustworthy mobile sensing. In: Proceedings of the Eleventh Workshop on Mobile Computing Systems, HotMobile 2010, Annapolis, Maryland, pp. 31–36 (2010)
14. Talasila, M., Curtmola, R., Borcea, C.: Handbook of Sensor Networking: Advanced Technologies and Applications. CRC Press, Boca Raton (2015)
15. Bhatlavande, A.S., Phatak, A.A.: Data aggregation techniques in wireless sensor networks: literature survey. Int. J. Comput. Appl. **115**(10), 4 (2015)
16. Tham, C.-K., Sun, W.: A Spatio-temporal incentive scheme with consumer demand awareness for participatory sensing. J. Comput. Netw. **108**, 148–159 (2016)
17. Yaqooba, I., et al.: Big data: from beginning to future. Int. J. Inf. Manag. **36**, 1231–1247 (2016)
18. Marz, N., Warren, J.: Big Data: Principles and Best Practices of Scalable Real-Time Data Systems. Manning Publications Co., Shelter Island (2015)
19. Gill, A.Q., Phennel, N., Lane, D., Phung, V.L.: IoT-enabled emergency information supply chain architecture for elderly people: the Australian context. Inf. Syst. **58**, 75–86 (2016)
20. Jiang, D., Ooi, B.C., Shi, L., Wu, S.: The performance of MapReduce: an in-depth study. J. Proc. VLDB Endow. **3**(1–2), 472–483 (2010)