The design of an IoT-GIS platform for performing automated analytical tasks

Hung Cao[a,*], Monica Wachowicz[a]

[a]*People in Motion Lab, University of New Brunswick*

Abstract

Society has a very ambitious vision of building smart interconnected cities through the Internet of Things (IoT). Billions of data streams will be generated by devices using different networking infrastructures of smart cities, enabling the automation of how the data that are being collected can be analysed for. However, significant scientific and technological challenges need to be overcome before IoT-GIS platforms can be widely used. This paper is a first step towards designing an IoT-GIS platform for performing automated analytical tasks that are able to retrieve, integrate and contextualize data streams with the purpose of adding value to the provision of transit services. Three automated tasks are used to describe our platform: (1) data ingestion for retrieving data streams; (2) data cleaning for handling missing and redundant data streams; and (3) data contextualization for representing the mobility context of transit driving behaviour. The Codiac Transit System of the Greater Moncton area, NB, Canada was used for building a mobility context and evaluating the cloud architecture that was used to implement our IoT-GIS platform. From the experimental results, the need for cloud computing for achieving scalability and high performance of our IoT-GIS platform is validated. Suggestions for the operational management of routes to improve service quality are proposed based on the analytical outcomes.

Keywords: Internet of Things; automated analytical tasks; mobility context; smart transit

1. Introduction

With the advent of the Internet of Things, the spread of geographically distributed devices equipped with sensing capabilities will generate real-time data streams that will be transported through communication networks such as WiFi, Bluetooth, Zigbee, LoRaWan, and 5G. The IoT devices are usually equipped with many kinds of sensors, ranging from accelerometers and gyroscopes to proximity, light, microphones, and cameras. They generate

☆*This is a preprint.* Download at DOI: https://doi.org/10.1016/j.compenvurbsys.2018.11.004

☆☆**Cite as**: Cao, H., & Wachowicz, M. (2019). The design of an IoT-GIS platform for performing automated analytical tasks. *Computers, Environment and Urban Systems*, 74, 23-40.

*Corresponding author

Email address: hcao3@unb.ca (Hung Cao) data streams that are usually an unbounded sequence of tuples that are most likely to be out-of-order and having a high data rate. A vast number of devices are being embedded into the very fabric of smart cities in such a way that they will revolutionize operational functioning and planning, through management, control and optimization of traditional services such as intelligent fleet management (Sun et al. 2016), smart parking (Mainetti et al. 2015) and digital health (Banos et al. 2016). This is already causing a shift from traditional GIS platforms towards IoT-GIS platforms in which IoT devices are linked by means of communication technologies that are crucial to enable smart cities functioning in real-time from routinely sensed data (Batty et al. 2012, Song et al. 2017). The fundamental assumptions underpinning GIS platforms are being challenged due to the proliferation of sensors, intelligent high bandwidth networks and cloud computing. In particular, traditional GIS platforms are inefficient mainly because they usually require heavily coordination of several tasks using limited computing resources. Moreover, the coordination of these tasks has been time-consuming and error-prone, since the tasks were not fully integrated, requiring human intervention for executing them to achieve new insights.

Automated analytical tasks must handle the continuous production of tuples flowing from the devices through a variety of tasks running on IoT-GIS platforms. These tasks will be performed at regular times (e.g. every hour) or be triggered every time the tuples arrive at a platform. Previous attempts have been focused on developing automated analytical tasks for network monitoring (Gupta et al. 2016), fraud detection (Rajeshwari and Babu 2016), data warehouse augmentation (Meehan et al. 2017), risk management (Puthal et al. 2016) and distributed processing of sensor-web data (Duckham 2012). No research efforts on developing IoT-GIS platforms have been found in the literature so far.

From a conceptual perspective, an IoT-GIS platform will play an important role in exploring data streams in time and space. Time is an important dimension of this platform, and different approaches have been proposed in the literature to handle unbounded data streams, including landmark windows (Leung et al. 2013), sliding windows (Lee et al. 2014), and tilted windows (Giannella et al. 2003). In contrast, the space dimension has been neglected so far, even though data streams are being generated over large geographical areas with fine spatial granularity. The scientific challenge is to integrate the notion of a mobility context into a IoT-GIS platform as being more than location, date and time (Bettini et al. 2010, Ranasinghe and Walpola 2016).

From an implementation perspective, an IoT-GIS platform will require (1) a pre-build connector that supports data connectivity to communicate with several devices, (2) a low-latency database for storing data streams, and (3) high performance processing for supporting the automated tasks. The technological challenge is to design an IoT-GIS platform that can perform analytical tasks without human intervention (e.g. an event from an IoT device triggers an analytical task), and at the same time, cope with the transportation of unbounded data streams where the data rate may overwhelm the processing power of this platform.

One way to address both scientific and technological challenges is to consider designing an IoT-GIS platform based on cloud computing for coupling data streams with automated tasks with the purpose of assessing existing transit services. Towards this end, this paper proposes the design of an IoT-GIS platform that supports three automated analytical tasks taking into account the mobility context given by a transit agency of a small urban area. They are: *data ingestion*, *data cleaning* and *data contextualization*. Each task consists of several automated steps that are designed bearing in mind a mobility context . Although the idea exists that context plays an important role in IoT, it continues to lack careful examination. Many mobility contexts may exist according to the relevance of taking into account the contextual history derived from the actual mobility of transit vehicles and their interaction with urban forms (i.e. streets and intersections).

Our research assumption is that mobility contexts help to explain the phenomena, reinforces different perspectives, provides truly understanding of the background of the problems and may have many dimensions such as spatial, physical, social, and temporal. And as a result, they are an important requirement, together with scalability and automation to take into account when designing an IoT-GIS platform. The proposed IoT-GIS platform is demonstrated with AVL stream data (Automatic Vehicle Location) collected by the Codiac Transit Agency of the Greater Moncton Area, which serves a small urbanized area in New Brunswick, Canada. Small transit agencies usually lack resources and have small fleet sizes and simpler route structuring, making the IoT-GIS platform relevant to improve their ability to collect data, to coordinate the analytical tasks and access the results, as well as to monitor operational strategies.

The remainder of this paper is organized as follows. In Section 2, related works in GIS platforms previously developed for smart transit applications are reviewed and the existing IoT platforms are described. In Section 3, the IoT-GIS platform is presented, including the details of the automated tasks, specifications, and requirements. Section 4 is dedicated to describing the cloud architecture used to implement the IoT-GIS platform. Section 5 describes in detail the experiment of implementing our IoT-GIS platform for the Codiac Transit Agency. Section 6 discusses both the performance of the proposed platform and the experiment analysis results. Section 7 concludes the paper and discusses further research.

2. Related work

Small transit agencies tend to have limited resources for facing the challenges of continually increasing the high quality of the delivered transit services and reducing private car dependency while ensuring low operational costs, low environmental impact, and safety. To this end, transit operators and managers need to understand the functioning of their services to develop strategies for their availability, reliability, and performance. Although a significant effort toward automating the collection of data has been achieved by transit agencies, including Automatic Fare Collection (AFC), Automatic Vehicle Location (AVL), and Generated Transit Feed Specification (GTFS), the actual stream data generated at the vehicular level continues to be difficult to be retrieved due to its large data volume and the absence of automated tasks. Traditionally, the platforms have been designed for sending the stream data to a server, where the data can be later stored in a GIS where further pre-processing is manually performed and ad-hoc queries are executed by the users of this platform. Some examples include the SQL database platform integrated with a web interface proposed by Pi et al. (2018) that allows users to perform interactive queries to examine the impact of bus bunching in a transit network performance using metrics about the bus routes, bus stops and trips obtained from four years of Automated Passenger Counter (APC) and AVL data. Luo et al. (2018) proposed a PostgreSQL-Matlab platform for carrying a sequence of pre-processing steps needed to integrate AFC, AVL, and GTFS datasets and later generating space-time seat occupancy graphs which have provided transit operators with information about crowding patterns that can be used to improve timetable optimization and fleet scheduling. The pre-processing steps are manually performed, and the time processing of each step can vary significantly depending of the availability of stream data.

Small transit agencies also lack the resources for performing analytical tasks that are vital for developing a long-range strategic plan or avoiding planning in a reactive manner. Previous research work has demonstrated the important role of analytical tasks in providing new insights for large transit agencies. Zhong et al. (2014) have applied a two-step analytical framework based on a probabilistic Bayesian model combined with IDW function in ArcGIS to build functions from equivalent daily social activities using data from surveys carried out every four years and the smart card system generated by the Singapore Land Transport Authority. Isukapati et al. (2017) demonstrates how descriptive analytics tasks can provide new insights on the dwell times at bus stops of two sample bus routes provided by Port Authority of Allengheny County, Pennsylvania. The results can be used for improving urban traffic signal control when the uncertainty in dwell times at bus stops might result in delays for the traffic flow. However most of these tasks tasks have not been developed to be executed in any platform yet, and as Lv et al. (2017) point out that not only data collection tasks but also analytical tasks will become more automated in the near future.

There is a growing interest and demand to develop IoT platforms that can support automated analytical tasks, ranging from data collection and pre-processing tasks to analytics and visualization tasks. Our research work is one step in this direction. A systematic overview of IoT can be found in several surveys that have been recently published. Some examples include the survey of Al-Fuqaha et al. (2015) that provides an overview of IoT enabling technologies, protocols, and applications where authors summarize key elements to realize the IoT, and point out the need for new IoT platforms that can offer automated management, data aggregation, and protocol adaptation among different IoT devices. In contrast, Li et al. (2015) have mainly focused on examining the Service-oriented Architectures(SoA) in IoT, showing that the main research challenges in designing the architecture of IoT platforms are the nature of heterogeneous, real-time movement of the IoT devices. Several IoT platforms have also been proposed based on Service-Based IoT Middleware, Cloud-Based IoT Middleware, and Actor-Based IoT Middleware which are supporting computing services in a cloud environment (Ngu et al. 2017). Gazis (2017) has recently pointed out the lack of standardization of IoT platforms in terms of services, data, and communications.

Over 400 IoT platforms have already been proposed to address sensor technologies and communication networking challenges for supporting supply-chain, manufacturing, and smart homes applications. Although these research efforts are in progress to design IoT-based systems (Carrez et al. 2017, Datta et al. 2014, Krco et al. 2014, Lloret et al. 2016, Nelson et al. 2017, Sarkar et al. 2014, 2015), most of the research work has been focused on platforms using fixed IoT devices that are tagged to a specific location, while not many efforts have attempted to solve the problems in the context of moving IoT devices (Chun and Park 2015, Gerla et al. 2014, Shibata and Sato 2017, Wu et al. 2015). Our research work envisages that a transit vehicle will become a moving IoT device in the future, and IoT-GIS platforms will play an important role in providing new insights in understanding transit network performances as well as automated tasks for fostering innovative transit applications.

One example includes the Smart Object platform that demonstrates the feasibility of supporting real-time monitoring of commodities in a supply chain by attaching RFID tags to objects such as consumer goods, product parts, pallets, containers, and vehicles. The RFID readings provide automatic object location and environmental sensors are also used to add additional information relevant to the context of a particular monitored item (López et al. 2012). A similar approach was used to design the Virtual Object (VO) platform for traffic monitoring in digital cities using inductive loop detectors for detecting vehicle passing or arriving at a certain point (Somov et al. 2013). Both approaches provide a virtual representation of real-world objects with a corresponding virtual object in the platform. The concept of VO allows us to deal with the problems of sensor heterogeneity and system scalability as well as enrich IoT data streams with metadata (i.e. context information).

Kantarci and Mouftah (2014) presents a pioneering research work on the conceptual design of the MATCS (Mobility-Aware Trustworthy Crowdsourcing) platform by incorporating user auction procedures based on current location of the users and their estimated dislocation during the crowdsourcing process. Although this platform has not been implemented yet, the simulated results validate the importance a mobility context has in collecting and verifying IoT data streams.

In contrast, very few cloud platforms can be found in the literature for supporting analytical tasks. Sun et al. (2016) proposes the MOMA (Moving Object Map Analytics) platform for manually performing a list of high performance tasks including GPS noise filtering, map matching, geo-fencing, contextual map fusion and trajectory pattern learning. Using a service-oriented architecture, the GPS trajectories were manually enriched by adding attributes such as weather, road type, and traffic condition that were used to build mobility contexts such as a single trip, personal profiling, and population profiling. Their preliminary results have pointed out that performance and scalability are the key technical challenges for improving their platform, especially for building mobility contexts that can handle a large number of data streams and automated tasks that can support high performance.

The UBICON platform proposed by Atzmueller et al. (2016) is the first attempt to take into account a social context within an analytical process. Although the tasks were not automated, the platform is developed for performing data capture, localization and activity recognition component in which different technologies and open-source tools are used such as the Sensor Data Collection Framework (SDCF), the WEKA toolkit, the VIKAMINE platform, and the GNU R environment for statistical computing. The social contexts were used for illustrating the capabilities of different tasks ranging from face-to-face social interactions to participatory open-sensing. Several applications were sketched by using this platform to perform analytical tasks. For example, indoor localization is identified through context inference using Bluetooth low-energy (BLE) technology or contexts are predicted based on interpretable class association rules.

In summary, it is important to point out that the first phase in the evolution of IoT has been focused on the proliferation of devices, protocols, and architectures where the main research challenges have been related to connectivity, physical infrastructure, sensors, and hardware configurations. A second phase is taking place where the core research challenges are shifting to software design, automated analytics, and platform configuration. Our research effort in designing an IoT-GIS platform is somewhere between the first and second phase, and therefore, it might be vulnerable to major disruptions yet to come due to the advances in networking and database technologies as has been previous revealed by Verma et al. (2017).

3. The automated analytical tasks

We propose an IoT-GIS platform focusing on using data streams which are defined as a sequence of tuples that usually contains attributes such as:

$$\{[T_1 = (S_1, x_1, y_1, t_1)], [T_2 = (S_2, x_2, y_2, t_2)], ..., [T_n = (S_n, x_n, y_n, t_n)]\}$$

where

> S_n: *is a set of attributes (i.e. measurements) obtained from an IoT device;*
>
> x_n, y_n, t_n: *is the geographical location of an IoT device at the timestamp t when a measurement has occurred.*

The main characteristics of tuples have been previously outlined by Gama and Rodrigues (2007). They can be described as one of the following:

- Each tuple in a stream arrives online. When the tuples are transported in batches, they are gathered in discrete packages at periodic intervals of time. An effective platform begins by prioritizing routing data packages to an automated task.

- A platform has no control over the order in which a tuple arrives within a data package or across data packages. When a task is automated, the platform used to carry out the task requires continuous queries. Two types of continuous queries are possible. First, pre-defined queries can be scheduled and they are one-time queries that can be provided by a task before any relevant tuple has arrived at the platform. Second, ad-hoc queries can be issued online and they are not known in advance by a task. They bring complexity to automating the tasks, and therefore, they were not used in this paper.

- Tuples are potentially unbounded in size. Ideally, an IoT-GIS platform should support flexible data rates to make sure any relevant tuple has arrived at the platform. Unfortunately, current network technologies do not support such a capability.

Three automated tasks have been designed including *(1) data ingestion*; *(2) data cleaning*; and *(3) data contextualization*. The automation of these tasks is of paramount importance to streamline large amount of tuples. The data ingestion task consists of retrieving the data streams from different IoT devices and connecting to a GIS in the cloud platform. The data cleaning task involves running continuous queries to execute common geo-processing tasks. Finally, the contextualization task is the most complex task because it contextualizes the tuples from the previous tasks by attaching new attributes to each original tuple according to a specific mobility context. The a-priori knowledge about the nature and scope of the movement of the IoT devices (i.e. the mobility context) is of paramount importance to design any automated task because it takes into account the geographical distribution of IoT devices, their mobility, and the low latency of a communication network. Figure 1 illustrates the overview of the automated analytical tasks.

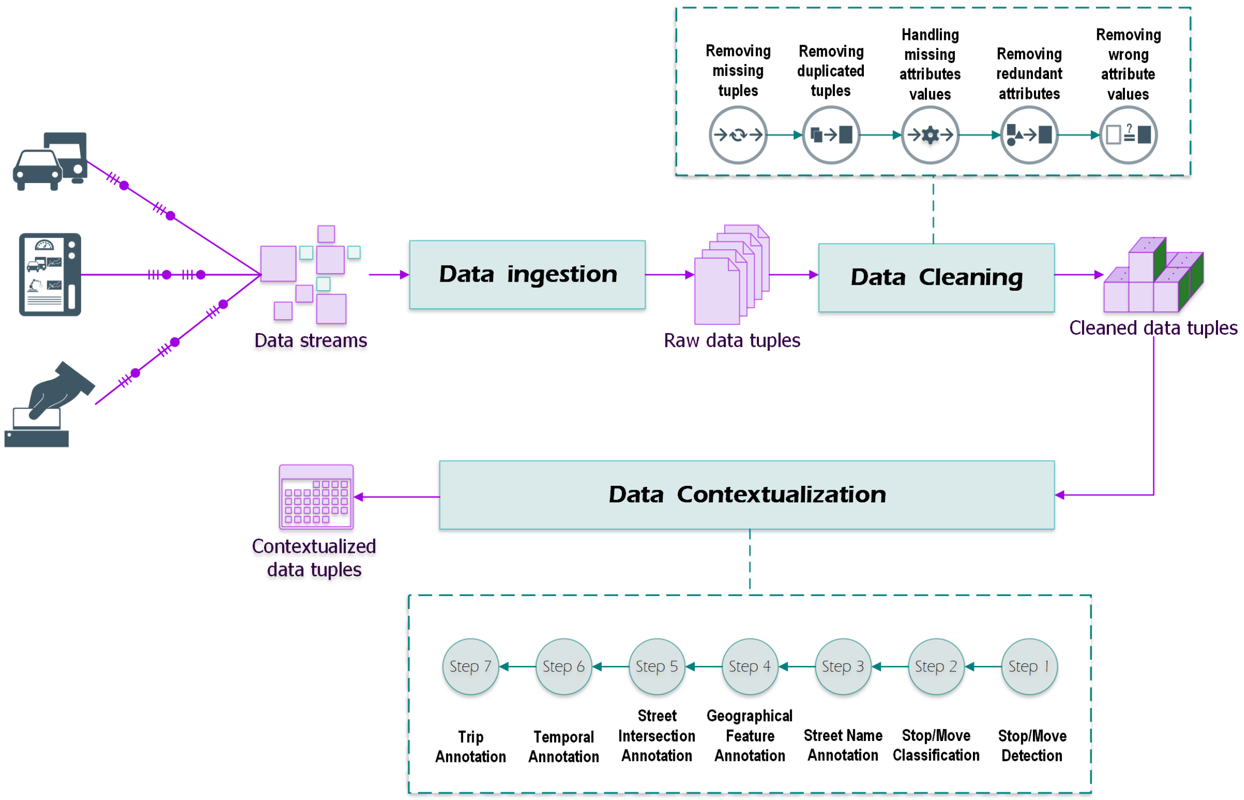


Figure 1: Automated tasks our IoT-GIS platform.

In our research work, determining a mobility context requires us to make several assumptions which are common in the literature of mobility analytics (Doulkeridis and Vlachou 2017, Velt et al. 2017). The first assumption is that the mobility context will be developed using the concepts of a *"trip"* at the individual scale and a *"network of trips"* at the aggregated scale. At the individual scale, a trip taken by a moving IoT device will dictate how the data streams are acquired, the sensors being used, and the mobility context of the data being harvested. There are many definitions of a trip, but in our mobility context we define a trip as a sequence of tuples which represents the origin, moves, stops, and destination of a moving IoT device. We do not claim that this definition captures the human mobility context of any IoT devices in the near future, but it can allow us to design the automated tasks with some reasonable certainty with the available IoT technology today. At the aggregated scale, a network of trips is needed to represent any trip of a moving IoT device. When the IoT data are aggregated into groups of trips based on a mobility context, it considerable reduces the processing time of our automated tasks. To achieve that, our second assumption is that a cloud computing platform is the most appropriate for implementing our automated tasks because it provides the flexibility of connecting it to a variety of IoT devices. Finally, the scalability characteristic of cloud computing allows us to design our automated tasks to be operated without processing power constraints.

3.1. Data Ingestion

The data ingestion task is known as the undertaking of pushing tuples from different devices into our IoT-GIS platform. The ingestion task allows an http POST, Wi-Fi and a 3G connection for rapid retrieval of tuples from the devices themselves as well as a broadcasting service in which a forever loop of event time windows can be applied. Selecting the time granularity of an event time window will depend on the selected mobility context. It should not be determined using the data rate of the IoT devices, since data rates are not useful to build a mobility context.

There are two advantages of using event time windows. First, they separate the semantics of program from the real streaming speed of the communication network (e.g. Wi-Fi or 3G). Hence, historical tuples can be processed, while streaming tuples are continuously produced within the same task. Moreover, the event time windows also restrict semantically inaccurate results in the scenario of delays due to network congestion or failure recovery. Second, they deliver more accurate outcomes, even if the tuples arrive out of their timestamp order.

All the tuples that arrive in the IoT-GIS platform are stored in a PostgreSQL database according to the a-priory specified event time windows. Although NoSQL databases such as MongoDB, Cassandra, and HBase are well suited for storing and indexing the tuples, they might lack the functionality of storing and manipulating geographical information that is needed to build a mobility context. Moreover, the lack of a database schema of NoSQL databases may cause a continuous query to fail due to unpredicted application behaviour. The PostgreSQL database provides a central database schema in our cloud platform, and a pre-defined query to retrieve the tuples needed for the automated tasks. Moreover, the PostgreSQL community have added many new features and better performance for big data use cases including the ability to store unstructured data and add a column on the fly in a dynamic table (Chihoub and Collet 2016).

In summary, a data package containing a set of unbounded tuples keeps being pushed to the IoT-GIS platform and stored in a PostgreSQL database which can be queried to retrieve the tuples using different event time windows, ranging from hour, day, week, month, and year. In this paper, we have executed a query to retrieve a year of tuples to illustrate the outcomes of our proposed IoT-GIS platform.

3.2. Data Cleaning

The data cleaning task is always necessary in order to eliminate inconsistencies and errors from the stored tuples. Guaranteeing data quality for continuous and high volume of tuples is a non-trivial task, and performing this task automatically is even more challenging because IoT devices usually produce a vast amount of noisy data. The task is automatically initiated using a continuous query that aims to retrieve all the raw tuples in the PostgreSQL database. Five automated data cleaning steps are designed including *(1) removing missing tuples*, *(2) removing duplicated tuples*, *(3) handling missing attribute values*, *(4) removing redundant attributes*, and *(5) removing wrong attribute values*. These steps are executed in conjunction with the pre-defined query running in the cloud platform and they can be described as one of the following:

- *Step 1 - Removing missing tuples:* Every data package is expected to arrive accordingly to the selected event time window (e.g. every 5 seconds, every hour). However, due to connectivity and/or sensor problems, missing tuples usually occur for a trip, and they are ignored.

- *Step 2 - Removing duplicated tuples:* It includes the case when the same tuple is transmitted twice. In this situation, any duplicated tuple is identified through its timestamp trace and then removed.

- *Step 3 - Handling missing attributes values:* S is a set of a finite number of attributes which is transmitted for each tuple. If the missing attribute value of a tuple is not used in the further steps, the *"N/A"* is assigned to this attribute. Otherwise, we delete the entire tuple.

- *Step 4 - Removing redundant attributes:* Although S has a fixed number of attributes, there are cases when a new attribute is added to a tuple during the transport to the cloud platform. For example, in the case of having a set of 4 attributes, it might occur that 5 attributes are retrieved instead. In this scenario, the additional attribute is automatically removed.

- *Step 5 - Removing wrong attribute values:* A wrong value for an attribute might occur due to uniqueness violation and misspelling. In this scenario, the data cleaning task tries to standardize and normalize the wrong value. But if the value cannot be standardized, the attribute is treated as a missing attribute value case.

Once the data cleaning task is finished, a target data set is automatically created. This is a cleaned data sample ready to be used by the data contextualization task.

3.3. Data Contextualization

This is the most complex task designed to be automated in our analytical process. Contextualization enriches the tuples step by step from the prior data cleaning task by adding new attributes to each tuple according to a specific mobility context. But before this task even starts, the tuples need to be sorted in the most effective manner for executing the data contextualization steps. Towards this end, the contextualization steps are executed using the Hadoop MapReduce framework. The *Map()* phase of MapReduce framework is utilized to bundle the tuples coming from the previous task into various groups that are later processed in a parallel style by the *Reduce()* phase aiming to execute the data contextualization steps using a Python script. The key feature of MapReduce is its ability to perform the processing steps of a contextualization task across an entire cluster of nodes, with each node processing a partition of the stream tuples.

For this paper, we selected the concept of a trip to illustrate our mobility context. To this end, the data contextualization task consists of seven automated steps which can be described as follows:

- *Step 1 - Stop/Move Detection:* The aim is to determine whether an IoT device is moving, has stopped off, or has suspended its movement during a trip. In this contextualization, the timestamp t and the geographical coordinates (x, y) of each tuple are utilized. First, a fixed mobility radius for each IoT device is determined according to the mobility context of interest. Usually parameters used to determine the threshold value are speed or a fixed time distance. Second, the Euclidean distance of a trajectory of an IoT device is identified based on two consecutive tuples (i.e. points). If this distance is larger than the mobility radius, a new attribute which contains the value *"move"* is attached to the second tuple. In contrast, if this distance is less than the threshold, the *"stop"* attribute value is attached to the second tuple.

- *Step 2 - Stop/Move Classification:* The aim is to classify the moves and stops of each trip obtained from the previous step in order to improve our understanding about their mobility context. Any stop may occur because of an accident, traffic congestion, picking up passengers at a bus station, or a traffic light of one street intersection. A new attribute is attached to the original tuple.

- *Step 3 - Street Name Annotation:* The aim is to annotate the moves and stops according to the street nomenclature of a city network. A new attributed is attached to the original tuple.

- *Step 4 - Geographical Feature Annotation:* The aim of this step is to annotate the stops and moves according to a place of interest. Some examples include a bus stop, a shopping mall, or a hospital. A new attribute is attached to the original tuple.

- *Step 5 - Street Intersection Annotation:* The aim is to annotate the moves and stops that occur at the intersections of a street network. A new attribute is attached to the original tuple.

- *Step 6 - Temporal Annotation:* The goal of this step is to identify the actual arrival time and departure at a specific place of interest. A new attribute is attached to the original tuple.

- *Step 7 - Trip Annotation:* The aim is to tag each first tuple of a trip as origin and each last tuple of a trip as destination. The other tuples are then sequentially indexed.

At the end of the contextualization task, a new data set is generated and stored in the PostgreSQL database. This data set contains seven new attributes added to the original set of tuples obtained from the data cleaning task. These attributes represent the mobility context that characterises the interaction of IoT devices with their surrounding environment during their trips.

4. Cloud Architecture

Our IoT-GIS platform requires stream processing for supporting the continuous computation of data flowing through the automated tasks. This allows any tuple that is retrieved to be processed as soon as it arrives. The only constraint is that the output rate should be at least similar to the data input rate, mainly to have enough memory to store the data after each task is performed. Figure 2 provides an overview of the cloud architecture developed for our IoT-GIS platform. Cloud computing can facilitate massive-scale and complex data processing by taking advantage of virtualized resources, parallel processing, and data service integration with scalable data storage that can support the IoT data streams. Indeed, most of the analytical processes in IoT have been deployed in the cloud due to its flexibility and efficient resource provisioning (Botta et al. 2016, Cavalcante et al. 2016, Díaz et al. 2016, Fortino et al. 2014, Truong and Dustdar 2015, Wang and Ranjan 2015). Two virtual machines (VM) are leveraged to form the cloud architecture. The VM 1 is used to perform the data ingestion and data cleaning tasks as well as storing the GTFS and GIS data sets which are needed for the contextualization task. Moreover, both VM1 and VM2 are combined to implement a high performance Hadoop cluster for executing the contextualization analytical task.

4.1. PostgreSQL Specifications and Requirements

The database in the cloud has been designed to manage and store not only the raw tuples being generated by the IoT devices but also to integrate geospatial data provided as input to the automated tasks. The data sets used were the open spatial GIS transit network and the spatial data from the GTFS package. Therefore, one of the main requirements for our database is that it should be a fully ACID (Atomicity, Consistency, Isolation, Durability) compliant database with flexibility and high scalability in terms of geographical distribution. The PostgreSQL 9.5.3 database was deployed on a virtual machine *(CentOS 7.0 x64, Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz, 8 CPU cores, 29.3 GB RAM, 859 GB Disk).* PostgreSQL was selected not only because it supports storage of binary large objects but also because it provides a native programming interface for Python that is our language of choice for implementing the algorithms of the automated tasks. We have also used PostGIS which is an extension to the PostgreSQL, to support geo-processing needed for several steps of the contextualization task.

4.2. Hadoop Specifications and Requirements

Hadoop was primarily used for supporting the contextualization task. It is a Java-based open-source software framework that supports distributed storage and processing of massive

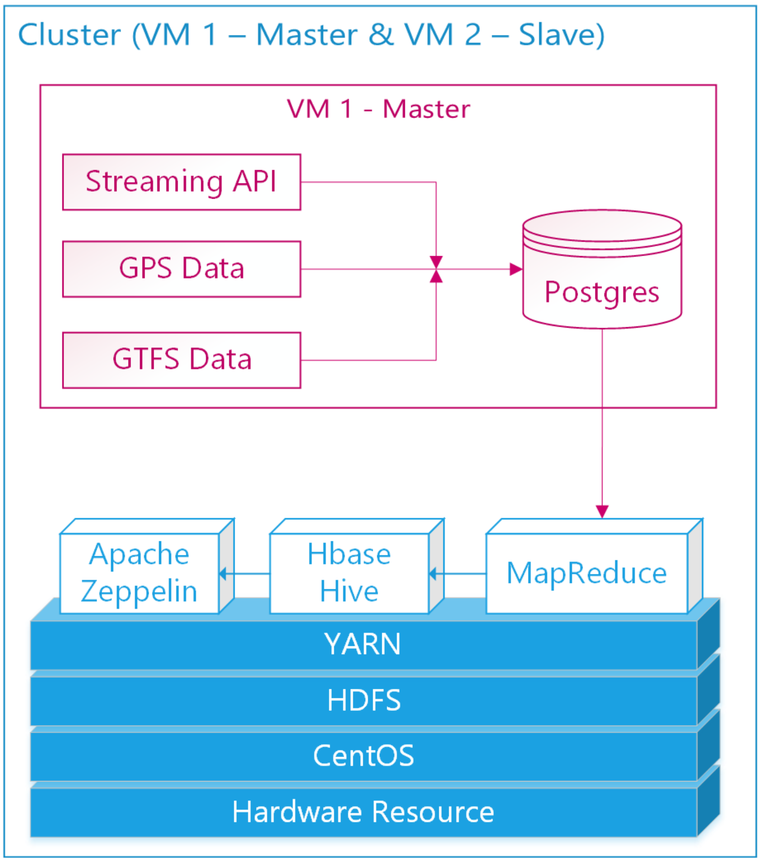


Figure 2: Overview of the cloud architecture developed for our IoT-GIS platform.

datasets across the clusters of commodity servers using the MapReduce framework (Dean and Ghemawat 2010). Hadoop was selected because it is designed to run applications on systems that have the scalability from a single virtual machine to many thousands of ones, with a high level of fault tolerance. The distributed file system (HDFS) facilitates rapid data transfer rates among machines and allows the system to keep working uninterrupted in case of a server failure. It divides HDFS data into large blocks that can be handled on many servers in the cluster. To handle the data, the Hadoop framework transfers packaged code for machines to process in a parallel manner, based on the data blocks each machine needs to process.

In our platform, a Hadoop cluster includes one master machine and one slave machine that were deployed on the Compute Canada West Cloud resource following the specifications listed in Table 1.

The main requirement for the cloud platform is to support the MapReduce framework for the sorting of the tuples of the IoT devices as illustrated in Figure 3.

The MapReduce framework basically performs two functions. First, the Map function divides the HDFS data set obtained from the cleaning data task into key-value pairs then shuffles them into many small subsets with the same key. The key-value pairs consist of

Hadoop cluster	
Master	Hostname: first-hung.westcloud OS: CentOS 7.0 (x86_64) CPU: Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz Number of CPU core: 8 RAM: 29.3 GB Disk: 859 GB IPv4 Address: 192.168.14.60
Slave	Hostname: third-hung.westcloud OS: CentOS 7.0 (x86_64) CPU: Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz Number of CPU core: 8 RAM: 29.3 GB Disk: 859 GB IPv4 Address: 192.168.14.67

Table 1: Overview of the Hadoop Specifications.

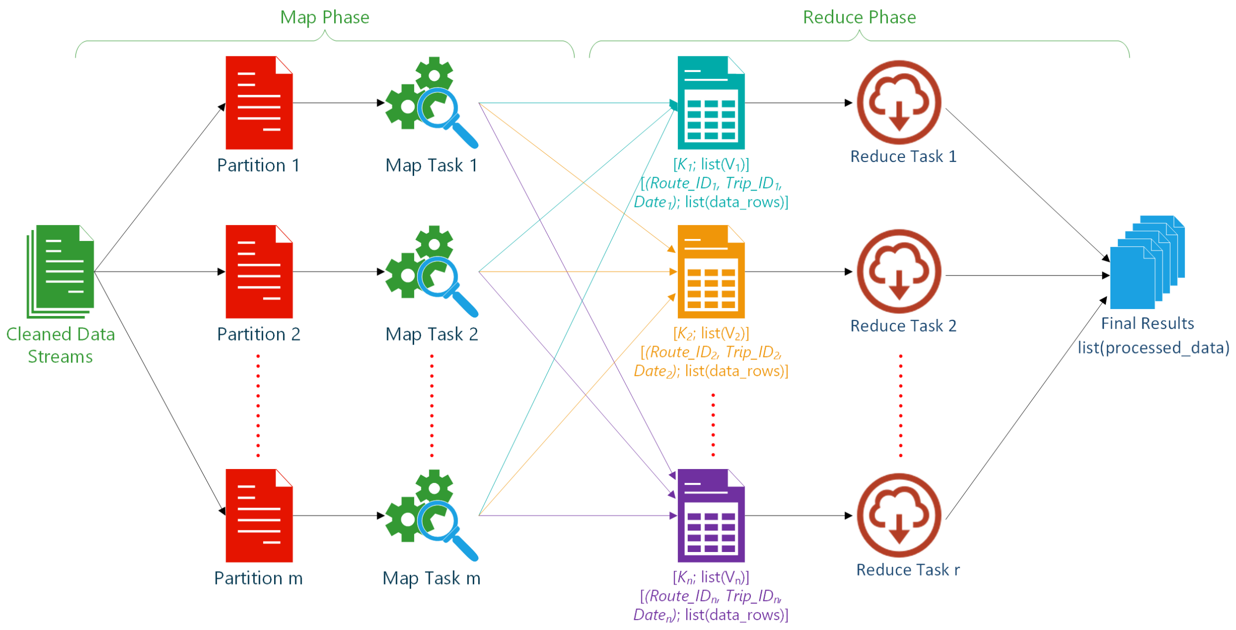

Map Phase
Reduce Phase
Cleaned Data Streams
Partition 1
Partition 2
Partition m
Map Task 1
Map Task 2
Map Task m
[K_1; list(V_1)] [($Route\_ID_1$, $Trip\_ID_1$, $Date_1$); list(data_rows)]
[K_2; list(V_2)] [($Route\_ID_2$, $Trip\_ID_2$, $Date_2$); list(data_rows)]
[K_n; list(V_n)] [($Route\_ID_n$, $Trip\_ID_n$, $Date_n$); list(data_rows)]
Reduce Task 1
Reduce Task 2
Reduce Task r
Final Results
list(processed_data)

Figure 3: Logical view of the contextualization steps using MapReduce.

any set of attributes that can uniquely identify a trip of an IoT device. Second, the map function maps this input data to a set of transitional key-value pairs. After executing the map function, the result is key-value pairs in which the value is the list of many sorted tuples T that have the same key as follows:

$$Map(T_1, ..., T_n) \rightarrow [K; list(sorted\_subset(T))]$$

The Reduce function takes these subsets and applies the contextualization steps in a parallel manner to produce a single result set. The Reduce function reduces a set of intermediate values which share a key K to a smaller set of values $list(F)$ as follows:

$$Reduce([K; list(sorted\_subset(T))]) \rightarrow list(F)$$

The input of the Reduce function is used to partition the tuples into groups having the same key-value pairs. At the end of the Reduce phase, all output of the Reduce function is grouped into a list of processed tuples of the form

$$F_1(S_1, \mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}, \mathbf{a_4}, \mathbf{a_5}, \mathbf{a_6}, \mathbf{a_7});$$
$$F_2(S_2, \mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}, \mathbf{a_4}, \mathbf{a_5}, \mathbf{a_6}, \mathbf{a_7});$$
$$...$$
$$F_n(S_n, \mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}, \mathbf{a_4}, \mathbf{a_5}, \mathbf{a_6}, \mathbf{a_7});$$

where $\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}, \mathbf{a_4}, \mathbf{a_5}, \mathbf{a_6}, \mathbf{a_7}$ are the new attributes.

5. Experiment: Smart Transit for Small Urban Areas

Small communities are usually confronted with unique challenges when providing transit services. Transit agencies typically serve populations that live in small urban areas with no congestion and plenty of parking facilities, and they are usually accustomed to short travel times. For our mobility context, we have selected the Codiac Transit that serves three communities: Moncton, Dieppe, and Riverview with a population of 130,000. Households pay $11 per month on average towards the operation of transit services. The latest available statistics show that 2,307,725 passengers used Codiac transit services in 2016 (Codiac 2018).

Codiac Transit operates 30 bus routes from Monday to Saturday; some of them provide evening and Sunday services. Despite the fact that all buses of the fleet are equipped with GPS, allowing transit managers to know the exact location of any bus every 5 seconds, every time Codiac Transit considers adjusting a route or launching a new route, they physically go out with a bus to pace. This operational undertaking is not ideal since Codiac Transit like most small transit agencies has limited human resources that can be devoted to perform such a task. Therefore, this mobility context was selected to illustrate that it is possible to accurately and automatically compute the pace of a route using our IoT-GIS platform. Figure 4 shows how data streams generated on the buses can be sent to our cloud infrastructure where the automated tasks are executed for building the mobility context. At the individual scale, every trip is automatically created by adding information about its actual origin, stops (e.g. stopover, and suspension of movement), moves (e.g. passing and running), destination, bus route, street names, and duration. At the aggregated scale, the trips are sorted out in chronological order and real-world patterns emerge showing the complex behaviour of the Codiac transit network.

Every bus in the Codiac Transit network is considered as an IoT device for the purpose of describing our mobility context. In total, data was collected for 800 trips containing 642 bus stations belonging to the 30 bus routes during the period of one year. The geographical distribution of the trip network is visualized in Figure 5.

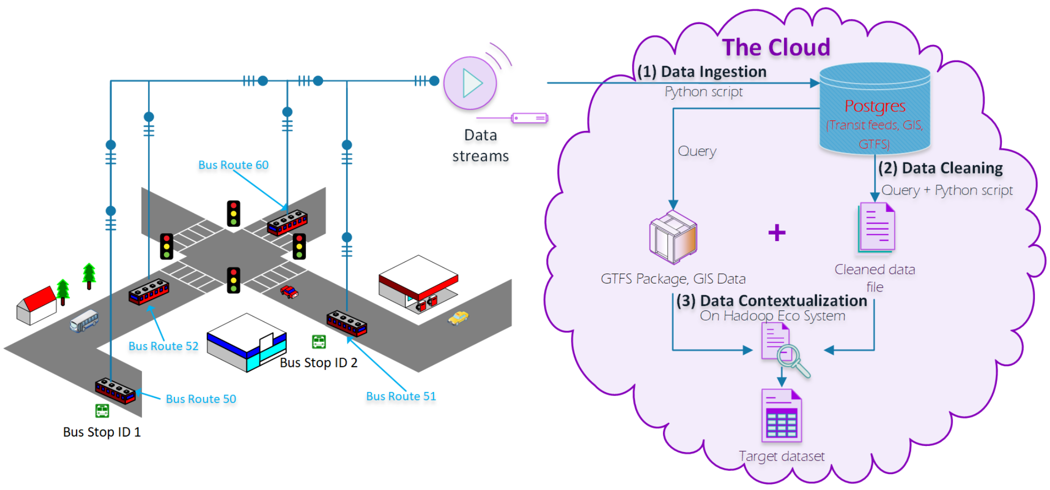


Figure 4: Overview of our IoT-GIS platform developed for Codiac Transit.

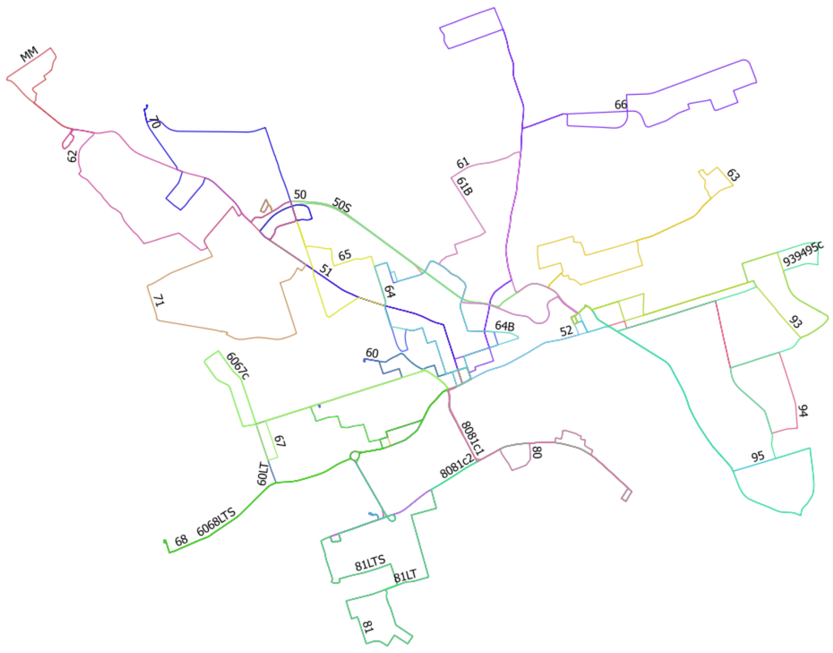


Figure 5: The Codiac Transit Network.

5.1. *Data Ingestion Task*

The tuples have been continuously pushed from the running buses to our PostgreSQL database on the west cloud of Compute Canada since 01/06/2016. We retrieved a year of streaming data from 01/06/2016 to 25/05/2017 to explain the implementation of our mobility context. Each tuple has 17 attributes described as one of the following:

1. *vlr_id:* The ID of the data point in the vehicle location reports table
2. *route_id_vlr:* The route ID in the vehicle location reports table

3. *route_name:* The route name
4. *route_id_rta:* The route ID in the route transit authority table
5. *route_nickname:* The abbreviation of the route
6. *trip_id_br:* The trip ID in the bid route table
7. *transit_authority_service_time_id:* Transit authority service time ID
8. *trip_id_tta:* Transit authority trip ID
9. *trip_start:* Start time of the trip
10. *trip_finish:* Finish time of the trip
11. *vehicle_id_vab:* Vehicle ID
12. *vehicle_id_vlr:* Vehicle ID in the vehicle location reports table
13. *vehicle_id_vlr_ta:* The descriptive name of the bus
14. *bdescription:* Bus description
15. *lat:* Latitude
16. *lng:* Longitude
17. *timestamp:* Timestamp of the data point

The data ingestion task is triggered every 5 seconds, and after a period of one year, there were 65,097,658 tuples stored in the PostgreSQL database that were used for the data cleaning task. Algorithm 1 provides the pseudo code for the execution of the data ingestion task.

Algorithm 1: Pseudo-code developed to perform the Data Ingestion Task.

```
Data: Set of $G = (T_1, T_2, T_3, ...)$ such that $T_i = (S_i, x_i, y_i, t_i)$ is a data tuple streamed
      every 5 seconds.
Result: $G = (T_1, T_2, T_3, ...)$ with $T_i \subseteq G$ such that $G$ was stored in PostgreSQL
        database in the Cloud.
Initializing database;
while True do
    // run the loop forever
    establish connection;
    read($G$);
    forall $T_i \subseteq G$ do
        if $T_i$ valid then
            insert($T_i$) to the database;
            print "Successful" ;
        else
            print "Failed" ;
            pass;
        end
    end
    delay(5); // ingest data streams every 5 seconds
end
```

5.2. Data Cleaning Task

The data cleaning task is triggered by a continuous query using a time window (i.e. the query runs automatically every 5 seconds) as shown in Table 2.

Description	SQL Statement

Get all raw tuples given a time period	SELECT *
	 FROM moncton_data
	 WHERE gps_timestamp BETWEEN '2016-06-01' AND '2017-05-25'

Table 2: SQL Statement implemented for the cleaning task.

Errors and inconsistencies information needed to be corrected and filtered out. In the case of missing tuples, 480,000 tuples were deleted accounting for 0.75% of total of 65,097,658 tuples, and because 6,000 bus trips had more than 100 missing tuples, they have been removed as well. Furthermore, around 6,000 tuples were standardized due to the cases of redundant attributes, missing attribute values, and wrong attribute values. Finally, 38,167,787 tuples were detected to be duplicated tuples, and consequently, they have been deleted as well. At the end, the cleaning data file consisted of 26,443,871 tuples which were used for the data contextualization task. Algorithm 2 provides the pseudo code developed to perform the data cleaning task.

Algorithm 2: Python pseudo-code developed to perform the data cleaning task.

```
Data: Set of $G = (T_1, T_2, T_3, ...)$ such that $T_i = (S_i, x_i, y_i, t_i)$ is the raw tuple queried
      from database
Result: $U = (T_i, ...)$ with $T_i$ is cleaned and $U \subseteq G$ is cleaned as well
Function Main($G$):
    forall $T_i \subseteq G$ do
        extract Trip $K_i$ from G ($K_i$= Set of different $T_i$);
        foreach $K_i \subseteq G$ do
            data_1 = clean_missing_tuple($K_i$);
            data_2 = fix_missing_attribute(data_1);
            data_3 = fix_wrong_attribute(data_2);
            data_4 = eliminate_redundant_attribute(data_3);
            D = eliminate_duplicated_tuple(data_4);
        end
        U = U.append(D);
    end
    return $U$;
```

5.3. Data Contextualization Task

The input data for this task consist of a set of cleaned tuples $U = (T_1, T_2, ...)$ for a one-year period. However these tuples require to be ordered by trip *(trip_id_br)*, bus route *(route_id_vlr)*, and date *(timestamp)*. This ordering is important to produce a set of contextualized tuples $Q = (T_i, ...)$ such that each tuple in the set Q is ordered for performing posteriori computations such as the Euclidean distances for detecting stops and moves. To this end, the MapReduce framework was used (Figure 3). The map function was responsible for selecting and ordering tuples into many small subsets in a parallel manner. All tuples with the same *Route ID*, same *Trip ID*, and same *Date* were grouped and sorted in a chronological order. Once the map function finished, the reduce function was used to implement the steps of the contextualization task. Algorithm 3 provides the pseudo-code developed to perform the data cleaning task.

Algorithm 3: Python code developed to perform the data contextualization steps.

```
Data: Set of $U = (T_1, T_2, T_3, ...)$ such that $T_i = (S_i, x_i, y_i, t_i)$ is the cleaned tuples
Result: $Q = (T_i, ...)$ such that $T_i = (S_i, x_i, y_i, t_i, context_1, ...context_n)$ is the
        contextualized tuples
Function Mapper($U$):
    foreach $T_i \subseteq U$ do
        key = $T_i(RouteID, TripID, Date)$;
        value = $T_i$;
        $Z_i$ = shuffle(key, value); /* sort all tuples with the same Route, same
         Trip and same Date to many small subsets $Z_i$ */
    end
    return $< T_{i(RouteID,TripID,Date)},\ Z_i >$;

Initialize Q = Empty;
Function Reducer($< T_{i(RouteID,TripID,Date)},\ Z_i >$):
    foreach key $T_{i(RouteID,TripID,Date)}$ do
        Initialize R = Empty;
        forall tuple $T_i \subseteq Z_i$ do
            $var_1$= stop_move_detection($T_i$);
            $var_2$= classification($var_1$);
            $var_3$= street_name_annotation($var_2$);
            $var_4$= bus_stop_identification($var_3$);
            $var_5$= intersection_identification($var_4$);
            $var_6$= arrival_departure_identification($var_5$);
            $var_7$=od_identification($var_6$);
            /* $var_7$ = $(S_i, x_i, y_i, t_i, context_1, ...context_7)$ */
            R = R.append($var_7$)
        end
        Q = Q.append(R)
    end
    return $Q$;
```

Figure 6 illustrates the contextualized steps for the bus trip 51-12 of the route 51 on the date 15-06-2016. This particular trip was randomly selected for explaining the outcomes of the contextualization task.

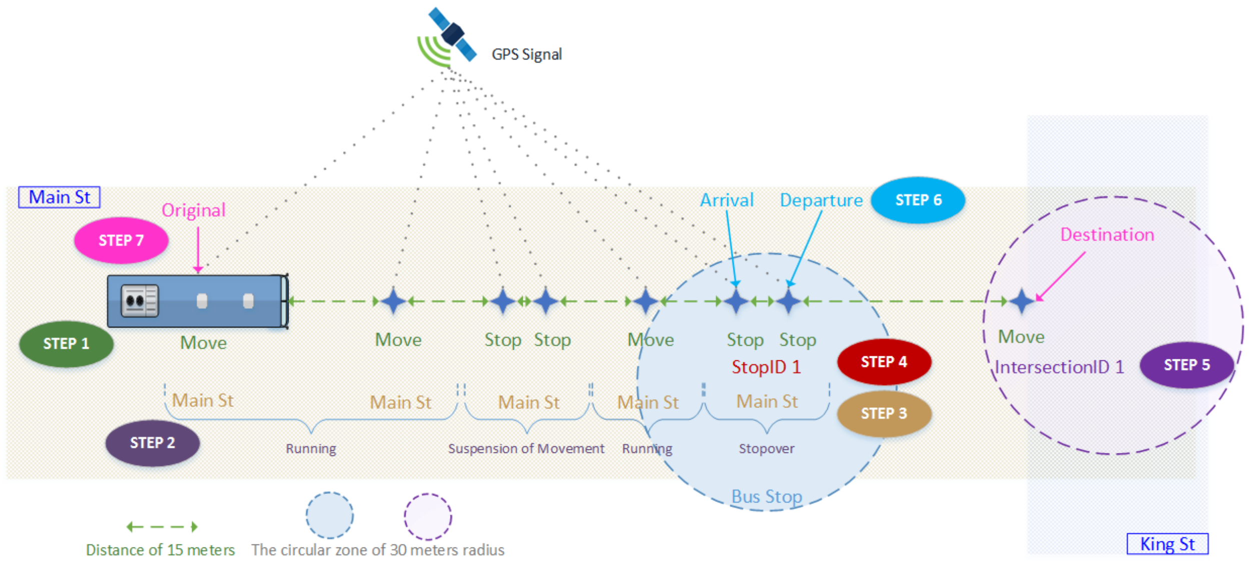


Figure 6: Overview of the automated steps designed for the data contextualization task.

Step 1 - Stop/Move Detection: For determining whether a bus was moving or had stopped off or had suspended its movement, an empirical radius value of 15 metres was selected to identify moves and stops. The Euclidean distance was also determined based on two consecutive tuples, and if this distance was larger than 15 metres, a new attribute which contains the value *"move"* was attached to the second tuple. In contrast, if this distance was less than 15 metres, the *"stop"* attribute value is attached to the second tuple. Figure 7(a) shows the contextualized results of Step 1.

Step 2 - Stop/Move Classification: This step was carried out by adding one new attribute which contained one of the following values:

- *Running:* when a bus is running on a street segment.
- *Passing:* when a bus passes a bus station because no passengers were waiting to be dropped off or get on.
- *Suspension of movement:* It may occur due to an intersection, stop sign, accident, or traffic jam.
- *Stopover:* when a bus stops at a bus station for dropping off or picking up passengers.

First, a query was executed to retrieve the geographical location of all the bus stations of a bus route from the PostgreSQL database (Table 3). This information was available from the GTFS data previously stored in the PostgreSQL database.

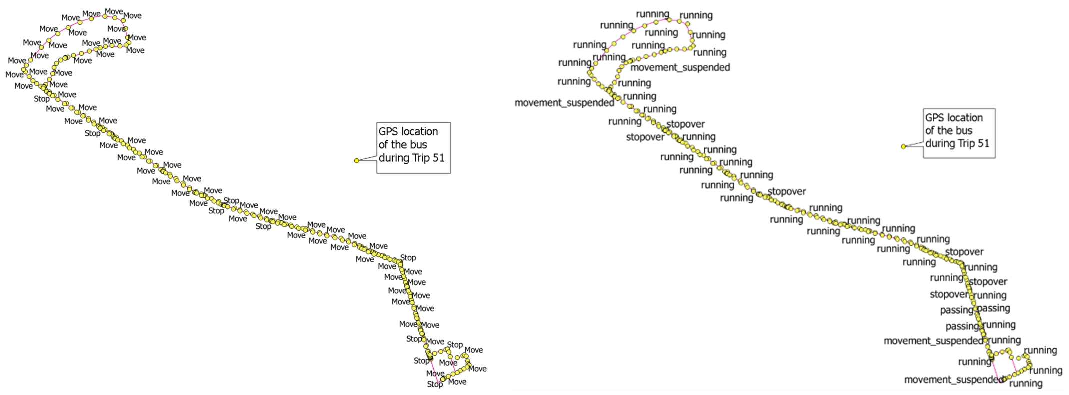
Afterwards, the algorithm created a circular zone with a radius of 30m for each bus station. The stops which are located inside the buffer are classified as *"stopovers"*; otherwise, they were classified as *"suspension of movement"*. Moreover, the moves which were located inside the buffer were classified as *"passing"*; otherwise they were classified as *"running"*

Description	SQL Statement

Get a list with all bus stations	SELECT trip_id, stop_id,
	stop_sequence, depart_return, change_direction, stop_lat, stop_lng
	FROM moncton_gtfs_dim
	WHERE trip_id = ‘trip’
	ORDER BY stop_sequence ASC

Table 3: SQL Continuous query for the Stop/Move Classification Step.

on a street. In this step, the moves and stops belonging to this bus trip were classified as running, passing, suspension of movement, and stopover. The classification results of this step are illustrated in Figure 7(b).


GPS location of the bus during Trip 51
Move
Stop
GPS location of the bus during Trip 51
running
movement_suspended
stopover
passing

(a) Stop/Move Detection. (b) Stop/Move Classification.

Figure 7: Results for one trip of the bus route 51.

Step 3 - Street Name Annotation: For this step, a query was designed to automatically retrieve the names of the street where a move or stop was located at (Table 4). Therefore, the GIS layer already stored in the PostgreSQL database was used for the contextualization. This was not a non-trivial step because the geographical coordinates of the stops and moves were obtained from GPS signals which can range from 10m to 100m accuracy in urban areas (Salarian et al. 2015). Using a grid-based buffer zone in PostGIS played an important role in indexing which street segment any cell belongs to, after localizing the moves and stops within a cell, and consequently, identifying the street name.

The GIS layer containing a 30m buffer zone along each bus route line of the Codiac transit network was created. Table 5 provides query statements to create the geographical grid cells for each bus route using bus route 51 as an example. First, a square grid of 10m cell is created using a geo-spatial query as reference layer and stored in the PostgreSQL database. Second, a query is used to retrieve only the square cells within the 30m radius from the street segments belonging to a bus route. The results of the query are stored in a new geo-spatial table in our database. Finally, we query the GTFS table to get the list street names of the bus route and assign the street name for each square cell (See Figure

Description	SQL Statement

Get a list with all street segments	SELECT nearest_street
	FROM route_"+line+"_grid AS g
	WHERE
	ST_SetSRID(ST_MakePoint" +str(point)+ ", 4326) &&
	.geom_lat_lon

Table 4: Continuous query for the Street Name Annotation Step.

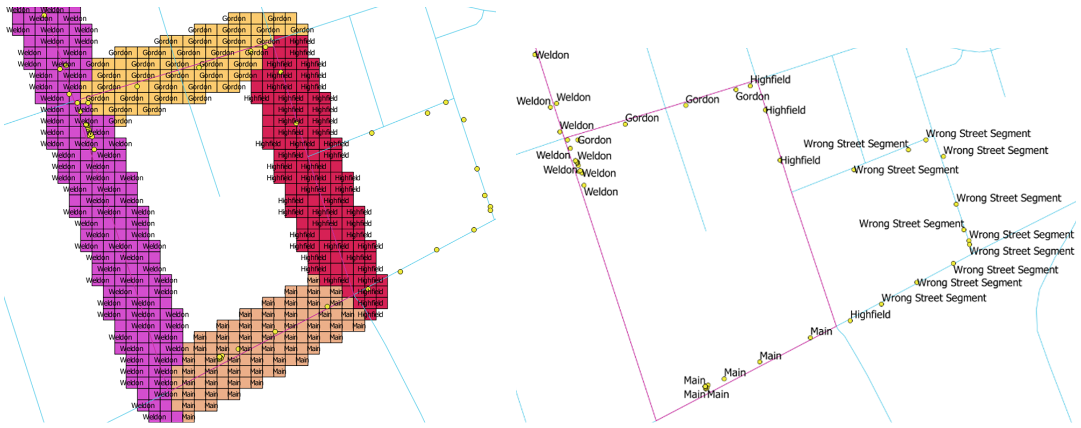
8(a)).

SQL Statement

– Create a table specifically for each route that holds the grid cells
DROP TABLE IF EXISTS route_51_grid;
 	
– Create the table for the route from those grid cells within a 30m radius of the road
CREATE TABLE route_51_grid AS
*(SELECT * From moncton_grid_10m AS m*
WHERE ST_DWithin((SELECT geom_lat_lon FROM bus_routes
WHERE route_id = '51'), m.geom_centroid, 30, false));
 	
CREATE INDEX geom_centroid_51_index ON route_51_grid USING GIST (geom_centroid);
CREATE INDEX geom_51_index ON route_51_grid USING GIST (geom);
CREATE INDEX geom_lat_lon_51_index ON route_51_grid USING GIST (geom_lat_lon);
 	
– Import the annotated bus lines
ALTER TABLE route_51_grid ADD nearest_street character varying(50);
UPDATE route_51_grid SET nearest_street = (
SELECT s.stname
FROM line51_streetnames as s
ORDER BYgeom_centroid <->ST_Transform(s.geom, 4326) LIMIT 1);

Table 5: Continuous queries used for creating the grid cells.

Figure 8 shows an example of the grid-based buffer zone used for tagging the street names for one trip of bus route 51. In this case, it is possible to see that the moving bus has not followed the assigned bus route (See yellow points on the street block on the right in Figure 8). This might have occurred due to an accident, road construction, or any other event that required the driver to drive on different street segments. In the case that a moving bus does not follow the designated street segment, the algorithm generates the *"wrong street segment"* value. Such a problem was not foreseen by our automated task. More research work is needed to determine how to deal with unexpected annotation errors in an automated way.



(a) (b)

Figure 8: Example of the 30m buffer zone for executing the street name annotation step.

Step 4 - Geographical Feature Annotation: The next step is to tag a bus station id to each tuple containing the attribute values equal to stopover and passing. This is an important step to provide a link with the bus station id information available from the GTFS data. This was achieved by creating a circular zone of a 30m radius around each bus station of a transit network, and matching it with the stop (i.e. stopover and passing) location of a moving bus (Figure 9(a)). It is important to point out that the algorithm also needs to verify the direction of a moving bus (e.g. eastbound and westbound) in order to identify the bus station where a stopover/passing was actually located. We selected a tuple located at the middle of a bus route to use it as a reference point for identifying the direction of a moving bus. Each stop can be then annotated using *"outbound"* and *"return"* values (Figure 9(b)). Using the GTFS data stored in the PostgreSQL database, the location of a bus station is compared with an actual stop of a moving bus (Figure 9).

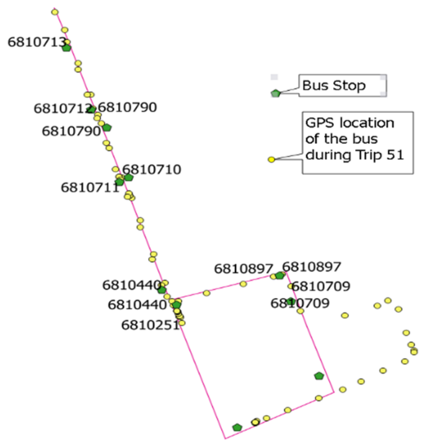
Step 5 - Street Intersection Annotation: The next step was to tag an intersection id to each tuple. This step starts with a continuous query used to select from the PostgreSQL database all the intersections (Table 6).

Description	SQL Statement
Get a list with all street intersections	SELECT * FROM moncton_intersection WHERE route_id = 'route';

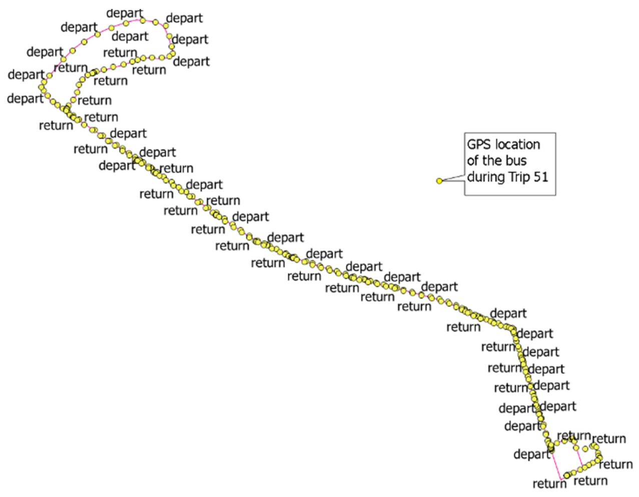
Table 6: Continuous query for the Street Intersection Annotation Step.

The algorithm creates a circular zone with a radius of 30m for each street intersection. The tuples containing stops and moves that were located inside the circular zone were tagged with the intersection id. Otherwise, the NULL value is used (Figure 10).

Step 6 - Temporal Annotation: The aim of the next step was to determine the actual arrival and departure time of a moving bus for dropping off or picking up passengers. In this



(a)



(b)

Figure 9: Results of the bus stop identification step for one trip of bus route 51.

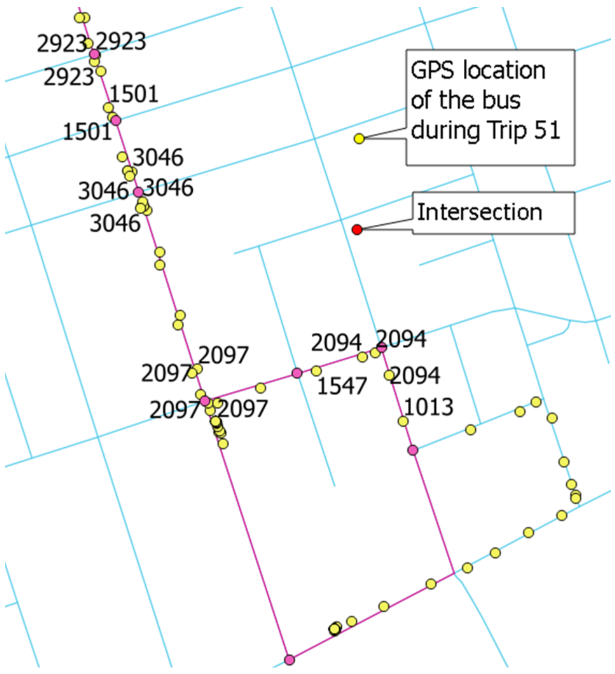
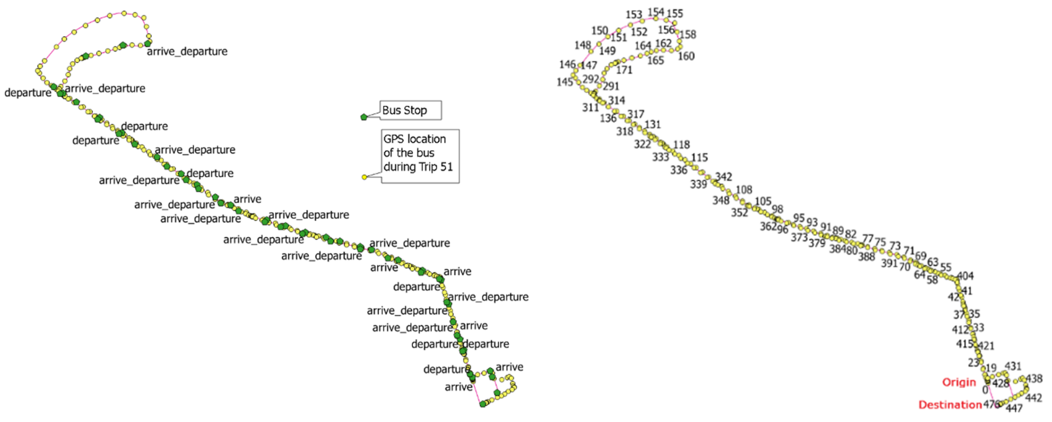


Figure 10: Results of the intersections identification step for one trip of bus route 51.

case, the algorithm verifies for the timestamp of the first stopover within the circular zone of 30m radius around each bus station, and considers it as the actual arrival time. Similarly, the timestamp of the last stopover within the circular zone is considered the departure time (Figure 11(a)). This step can be improved if automatic passenger counters (APCs) are used in a transit network because they provide information about passenger activity on bus trip time.

Step 7 - Trip Annotation: Finally, the last step was to tag each first tuple of a bus trip as origin, and each last tuple of a bus trip as destination (Figure 11(b)).



(a) (b)

Figure 11: Results from steps 6 and 7 of the contextualization task.

6. Discussion of the Results

The subsections below analyze several aspects of the IoT-GIS platform performance as well as the analysis of the mobility context to support smart transit application in the small urban areas. The first one evaluates computing performance of the analytical tasks run on the IoT-GIS platform based on the processing time metric. The second set of analyses focuses on many aspects to improve service quality of the smart transit application, including service coverage, pace behavioural, congestion patterns, and route connections.

6.1. Overall Computing Performance of the IoT-GIS Platform

The section evaluates the computing performance of the IoT-GIS platform. The data ingestion task was performed every 5 seconds achieving a performance latency near to 0.0 ms. Low latency processing is key when running the data ingestion task, and this could be achieved by optimizing algorithms to minimize the impact of disk I/O and the use of faster networking. Any delay in the execution of this task will have an impact on the execution of the other automated tasks in our IoT-GIS platform. Figure 12 shows the total processing time to execute the data cleaning task using the data streams gathered for one day, one week, two-week, and one month periods. Three bus routes having different trip frequency scheduling, high (Bus Route 51), medium (Bus Route 61), and low (Bus Route 80), were selected for this comparison. As we can see, the processing time varies according to the type of route and number of data streams.

Aiming to evaluate the automatic batch processing of the data contextualization task using MapReduce, two datasets were extracted from the cleaned tuples to run in Hadoop. The first dataset A contains the 12.75 million cleaned tuples from 01/06/2016 to 15/12/2016. The second data set B contains 13.69 million cleaned tuples from 16/12/2016 to 25/05/2017.

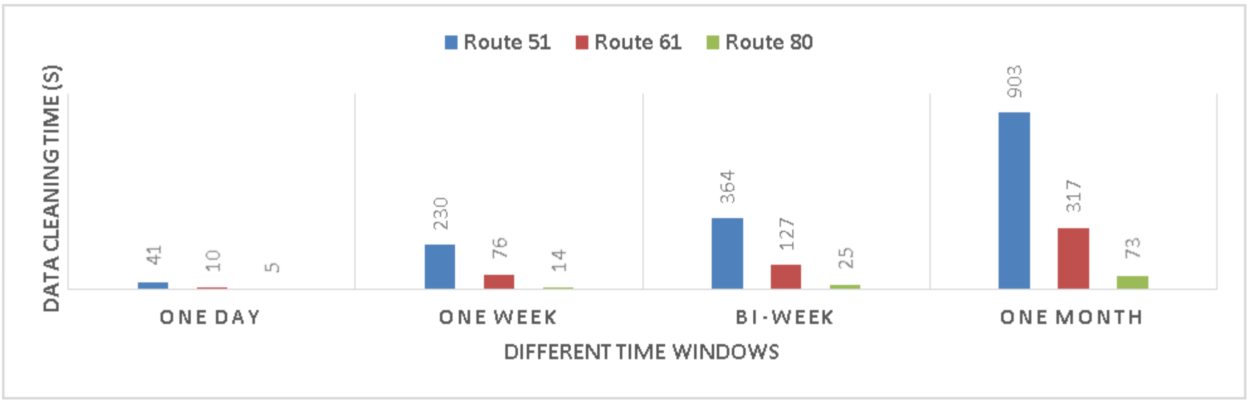


Figure 12: The measured cleaning times of 3 sample bus routes (51, 61, 80) operating in different areas over different time windows.

Figure 13 shows the processing time for all phases including map phase, shuffle phase, and reduce phase. Notably, the Reduce processing time is much longer than Map processing time because the Reduce phase runs all the data contextualization steps while the Map phase mainly sorts tuples into separate cluster of the same bus route.

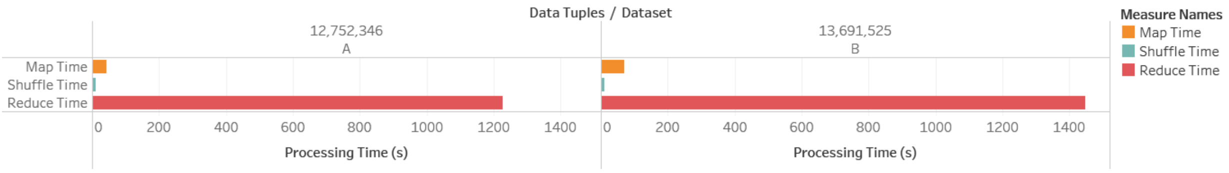


Figure 13: The measured processing times obtained from the MapReduce framework.

6.2. Experiment Evaluation

Table 7 provides an overview of the total number of tuples that have been contextualized according to our mobility context. In total, 82,044 trips have been processed by the analytical tasks and as a result, they have been stored in the PostgreSQL database. It is worth noticing that the total number of trips can vary significantly according to the bus routes, from 129 up to 10,263 trips, showing the high scalability of our proposed approach. Moreover, the total number of tuples that have been contextualized can also vary from, for example, 32,541 tuples annotated as *"Move"* for bus route 60LT to 2,049,041 tuples annotated as *"Move"* for the bus route 51.

With the statistics at hand, new insights have emerged about the patterns of the different paces of the bus routes of the Codiac transit network that point out a variety of transit improvements, infrastructure enhancements, and ridership strategies. First, the highest bus stop activity was found in the 52 bus route which is a a route with 20 bus stations and running in approximately a circular path in downtown Moncton. With 892,585 stopovers against to only 71,043 cases of buses passing a bus station, it reveals a captive ridership for this bus route (92% of usage pattern). However, the high number of cases of movement

	MOBILITY CONTEXT						
BUS ROUTE	Moves	Stops	Running	Passing	Stopover	Movement suspended	Trips
50	661,373	552,356	643,592	17,781	268,294	284,062	4,270
50S	45,809	35,820	44,755	1,054	13,113	22,707	301
51	2,049,041	2,135,951	1,779,599	269,442	888,435	1,247,516	10,263
52	1,359,354	2,079,425	1,288,311	71,043	892,585	1,186,840	9,900
60	744,855	579,926	608,990	135,865	362,600	217,326	4,591
60LT	32,541	9,648	27,903	4,638	2,345	7,303	129
61	907,851	562,502	815,458	92,393	262,306	300,196	5,133
61B	493,097	300,222	491,805	1,292	91,043	209,179	2,884
62	966,933	462,028	862,076	104,857	218,922	243,106	5,039
63	1,073,340	429,122	944,673	128,667	219,738	209,384	5,217
64	868,025	621,854	707,958	160,067	252,693	369,161	5,246
64B	177,181	98,161	157,832	19,349	33,207	64,954	1,011
65	810,943	624,252	707,095	103,848	211,322	412,930	5,085
66	320,882	117,009	294,982	25,900	22,079	94,930	936
67	346,419	138,161	305,164	41,255	30,794	107,367	1,774
68	359,649	151,026	306,361	53,288	36,722	114,304	1,855
70	350,666	223,059	331,537	19,129	109,925	113,134	2,033
71	363,895	242,956	326,833	37,062	54,591	188,365	2,159
80	235,039	106,009	206,429	28,610	18,658	87,351	1,174
8081c1	185,254	90,095	163,520	21,734	37,506	52,589	478
81	728,384	398,293	650,614	77,770	240,081	158,212	1,966
93	616,593	295,629	566,486	50,107	91,111	204,518	3,077
939495	11,346	1,804	10,552	794	344	1,460	40
94	833,145	390,926	760,908	72,237	161,111	229,815	4,578
95	541,689	289,693	494,189	47,500	114,486	175,207	2,905

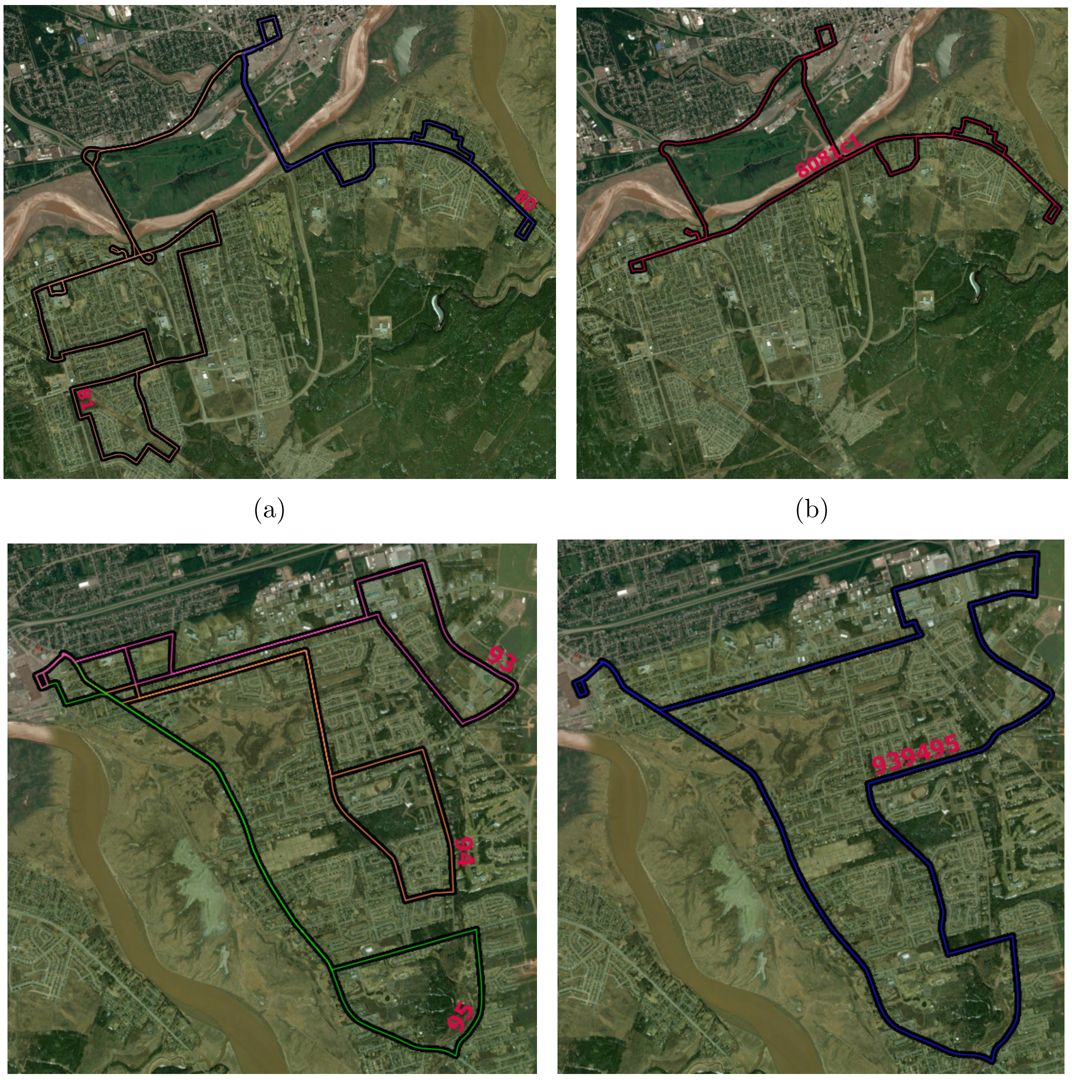
Table 7: Contextual statistics for the Codiac transit network.

suspension (1,186,840 or 61%) indicates the need for signal synchronization and bus priority on the Main Street where this bus route operates. In contrast, the feeder route 51 having 53 bus stations shows a similar pace behaviour in terms of total number of stopovers (888,435), but in this case, having a much higher number of passing events (269,442 or 23%). This might be an indication of bus stations that are not being utilized by the catchment ridership area, mainly because this service is serving the disadvantaged and the elderly.

Second, the results also reveal the bus routes where there is a larger number of passing events in relation to stopovers. This pace behaviour emerges from the bus routes serving remote areas of the metropolitan region and the Codiac agency must entice non-captive riders with improved levels of service or other improvements. They are bus route 66 serving the north region of Moncton, bus route 67 serving the Industrial Park of Moncton, bus route 68 serving the rural area of Moncton towards Salisbury, and finally bus route 80 serving

Riverview. Moreover, all these routes present a moderate number of movement suspensions, in particular bus routes 67 and 68 which have similar suspension patterns of 31% and 32% respectively. In this case, both services have to cross Highway 15, requiring an innovative strategy to optimize these services given this network infrastructure constraint.

Third, Table 7 also shows the pace behaviour of two new bus routes envisaged for merging routes 80 and 81, as well as merging bus routes 93, 94, and 95. Figure 14 illustrates how the routes have been changed. These new routes have been operated for only one trip a day for a total of 40 days. It is interesting to point out that new route 80/81 has shown a ridership improvement due to an increase of the number of stopovers of old route 80. Conversely, this is not the case for new route 93-94-95, since there has not been an increase of the number of stopover.



(a) (b) (c) (d)

Figure 14: Illustration of the old and new bus routes: (a) Bus route 80 and 81. (b) Merged bus route 8081. (c) Bus route 93,94, and 95. (d) Merged bus route 939495.

Finally, the Pace Behavioural Driving Index (PBDI) is computed for each bus route as:

$$\widehat{PBDI} = NORM\left(\frac{\sum S_i + \sum D_i}{\sum R_i + \sum P_i}\right)$$

Where:

$\sum S_i$: is the total number of stopovers.
$\sum D_i$: is the total number of movement suspended.
$\sum R_i$: is the total number of running.
$\sum P_i$: is the total number of passing.

In order to classify the traffic flow as follows:

- from 0 to <= 0.29484388 : no traffic
- > 0.29484388 to <= 0.3574634 : unblocked flow
- > 0.3574634 to <= 0.46832302 : optimal flow
- > 0.46832302 to 0.999 : congested flow

Table 8 shows the results for each bus route. This index can be used by transit managers to identify the bus routes that maximize the passenger carrying capacity of existing corridors, streamline transit services, and improve access to the transit system.

Bus Route	Normalized Pace Behaviour Driving Index	Traffic Flow
939495	0.10394008	no traffic
60LT	0.19381871	no traffic
66	0.23837638	no traffic
67	0.26071923	no traffic
63	0.26135618	no traffic
68	0.27451253	no traffic
80	0.29484388	no traffic
94	0.30673496	unblocked flow
62	0.31236418	unblocked flow
93	0.31342797	unblocked flow
8081c1	0.31792333	unblocked flow
95	0.34960472	unblocked flow
81	0.3574634	unblocked flow
64B	0.36216888	optimal flow
61B	0.39801502	optimal flow
61	0.40504083	optimal flow
70	0.4158296	optimal flow
71	0.43645638	optimal flow
64	0.46832302	optimal flow
65	0.50322119	congested flow
60	0.50896762	congested flow
50S	0.51116849	congested flow
50	0.54596138	congested flow
51	0.68144364	congested flow
52	0.999	congested flow

Table 8: The overview of Pace Behavioural Driving Index of each route in the transit network.

For the evaluation of these results, we have examined the monthly number of total stops and moves that have been computed for bus route 51. Table 9 shows the similar patterns encountered for *"stops"* and *"moves"*, having the highest peaks in the months of December and March.

Moreover, congestion patterns have also been inferred by looking at the occurrence of *"stops"* and *"moves"* at different street segments. Figure 15 shows that the highest number

	Jun-16	Jul-16	Aug-16	Oct-16	Nov-16	Dec-16	Jan-17	Feb-17	Mar-17	Apr-17	May-17
Stop	171,503	13,760	3,687	48,259	166,013	378,158	225,701	180,005	349,587	303,612	280,513
Move	160,543	16,336	4,697	58,568	153,390	385,446	216,495	155,427	354,073	285,827	245,907
Passing	20,433	2,727	793	9,012	18,990	49,826	27,785	20,748	49,129	41,756	35,282
Movement suspend	89,117	7,501	1,976	35,320	107,572	224,933	136,093	111,212	209,068	181,363	175,555
Running	140,110	13,609	3,904	49,556	134,400	347,952	188,710	143,157	321,316	258,846	224,184
Stopover	82,386	6,259	1,711	12,939	58,441	168,378	89,608	77,648	157,141	137,805	120,381

Table 9: Monthly total number of stops and moves for bus route 51.

of stops of bus route 51 have occurred at Plaza and Main Street probably due to traffic and weather conditions, meanwhile the Weldon St. and Mountain St. have a larger number of *"moves"*.

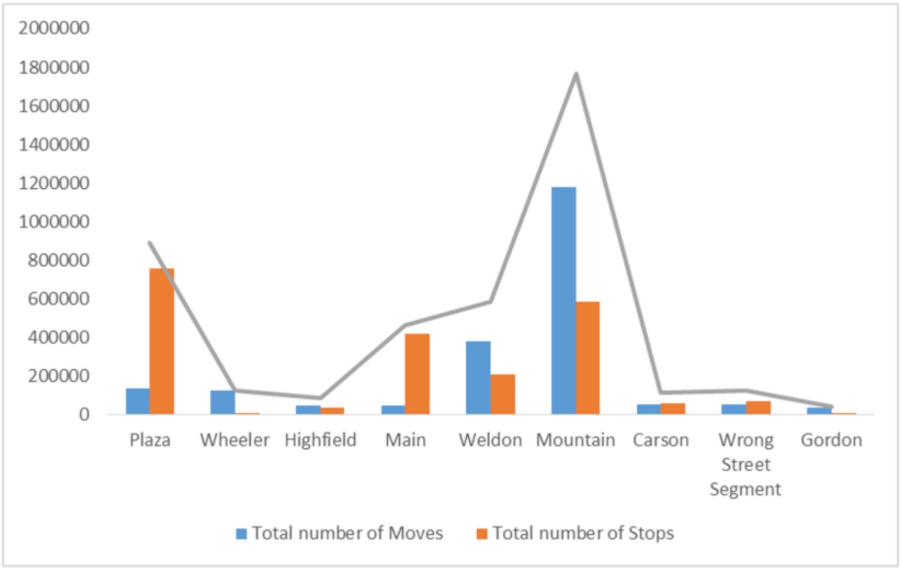


Figure 15: Overview of the total number of stops and moves of all trips of bus route 51.

Furthermore, the most congested intersections were found by looking at the the total number of the suspension of movement for bus route 51. Figure 16 shows the most congested intersections as being Intersection ID 778: (Birchmount & Mountain), Intersection ID 2215: (High & Mountain), Intersection ID 1592: (Maplelon & Mountain), Intersection ID 2836: (Mountain & Vaughan Harvey).

Transit vehicles require the synchronization of urban traffic signals since their suspension of movement at the intersections might cause delays. Figure 17 illustrates the location of the intersections that have the most impact on time adherence for bus route 51. This information shows a need for synchronization among these intersections

Despite the fact that 51 is the most used bus route in the network, Figure 18 shows while five bus stops near downtown were very busy, over 10 bus stops were unlikely to stop to pick up passengers, and 9 bus stops have not been used for a period of one year. According to this analytical result, the allocated resource for the bus stops need to be optimized to

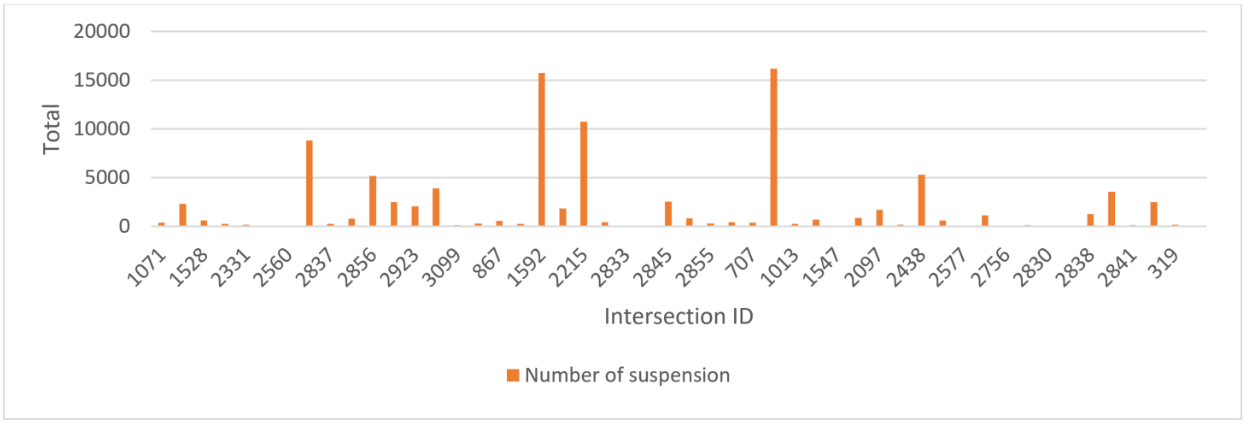


Figure 16: Total number of stops (suspension of movement) per intersection for all trips of the bus route 51.

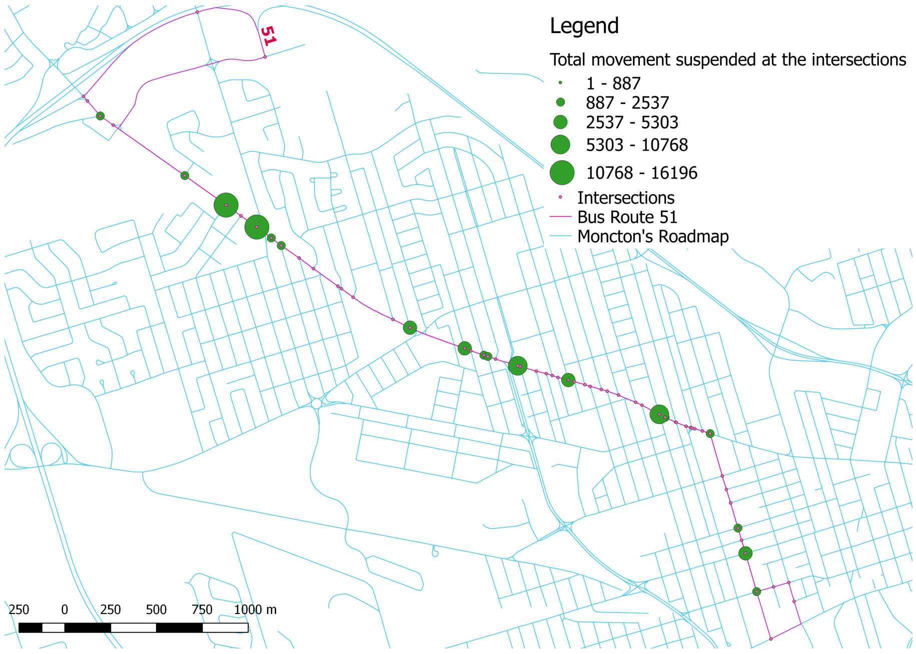


Figure 17: The movement suspended pattern along the bus route 51.

eliminate the redundant bus stops along this bus route.

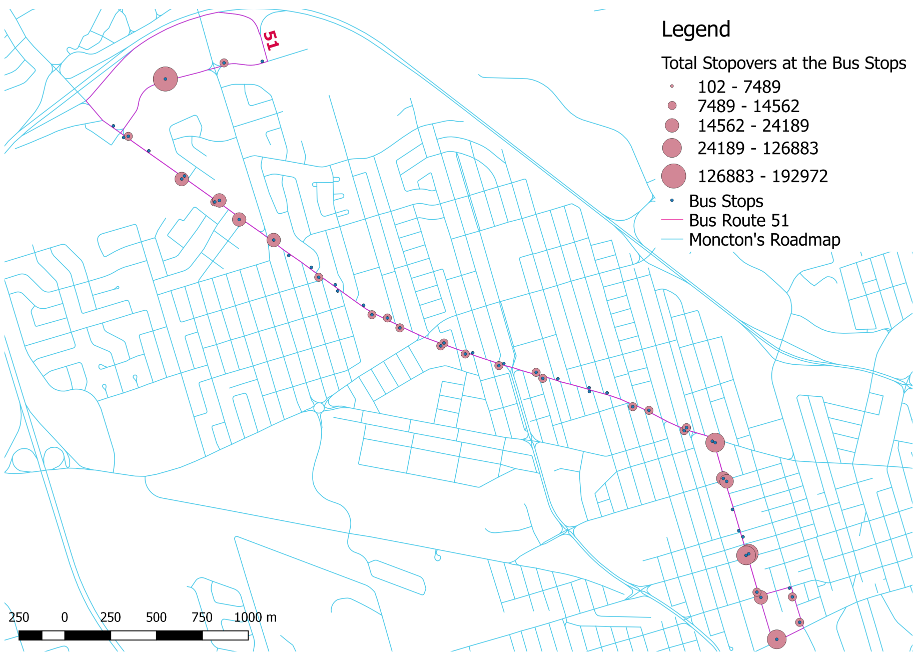


Figure 18: Total number of stopovers at each bus stop for all trips of the bus route 51.

7. Conclusions

Developing an IoT-GIS platform for supporting automated tasks requires an understanding of the structure of data streams (i.e. sequence of tuples) and communication network together with the cloud architecture needed for running the tasks. This is a challenging process, mainly because any automated analytical task will consist of many automated steps that rely on the selected mobility context. In this paper we have used the Codiac transit network to describe a mobility context that illustrates how pace driving behaviour can be computed and routing alternatives can be evaluated to improve the average speed of service. Our IoT-GIS platform provides operational information to small transit agencies despite the disadvantage of not having APC and AFC data. The platform has also the potential to be used by small agencies that tend to have limited staff available to develop dedicated programs for analysing the data to conduct their strategic planning process. Other mobility contexts where we could apply our IoT-GIS platform include autonomous vehicles networks using V2X communication for improving safety.

Our IoT-GIS platform has contextualized the raw data to show that it is possible to explore the semantics of a mobility context as well. However, our approach requires high performance computing power to support all the automated tasks, especially the contextualization task. Analytics performed over contextualized streaming data could potentially revolutionize transit network services that will be able to adapt at near real time to current or expected mobility contexts, implementing real-time operation controls and recommender systems. The outcomes from the data cleaning task indicate that it is not worth to send all the data streams to the cloud since most of them will not be used in the contextualization task. Almost half of the tuples used in our implementation were deleted during the data cleaning task. This implies that a significant number of moves and stops will not be used and could lead to errors and bias in the further analysis. Therefore, other computing architectures such as mobile fog computing might be more appropriate for performing the data cleaning task at the edge of the network, rather than the cloud. Mobile fog computing is defined as *"a scenario where a huge number of heterogeneous (wireless and sometimes autonomous) ubiquitous and decentralized devices communicate and potentially cooperate among them and with the network to perform storage and processing tasks without the intervention of third-parties"* (Vaquero and Rodero-Merino 2014). Data cleaning tasks can be designed for running in a sandboxed environment at a fog node. This will help to incorporate a new step in the data digestion task to handle late tuple arrivals. Future research work includes implementing the data cleaning task at a mobile fog node which would be installed inside a vehicle of a transit network.

Finally, our IoT-GIS platform has an enormous potential to be used to calculate transit performance indicators that have been previously computed using expensive transit demand models. Some examples include daily trip pattern construction for service adjustment planning, schedule coordination planning as well as in links re-routing and on-time transit performance improvement. While researchers have recognized the potential of using GPS coordinates for transit performance monitoring, there has been limited research in dealing with the practical considerations associated with the analysis of massive amounts of transit feeds. It is also important to point out that the 30m circular zones might not be a universal mobility radius to be adopted by any transit network. More research work is needed to identify the optimal radius value for the circular zones used for bus stations and intersections. Our IoT-GIS platform provides a unique approach to enable online applications for transit performance analysis in the near future.

Acknowledgement(s)

This research was supported by the NSERC/Cisco industrial Research Chair [Grant IRCPJ 488403-14]. The authors would like to thank the Codiac Transpofor providing the transit data. We also appreciate the feedback provided by three anonymous reviewers on the previous version of this manuscript.

References

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M., 2015. Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials 17 (4), 2347–2376.

Atzmueller, M., Fries, B., Hayat, N., 2016. Sensing, processing and analytics: augmenting the ubicon platform for anticipatory ubiquitous computing. In: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct. ACM, pp. 1239–1246.

Banos, O., Amin, M. B., Khan, W. A., Afzal, M., Hussain, M., Kang, B. H., Lee, S., 2016. The mining minds digital health and wellness framework. Biomedical engineering online 15 (1), 76.

Batty, M., Axhausen, K. W., Giannotti, F., Pozdnoukhov, A., Bazzani, A., Wachowicz, M., Ouzounis, G., Portugali, Y., 2012. Smart cities of the future. The European Physical Journal Special Topics 214 (1), 481–518.

Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D., 2010. A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing 6 (2), 161–180.

Botta, A., De Donato, W., Persico, V., Pescapé, A., 2016. Integration of Cloud computing and Internet of Things: A survey. Future Generation Computer Systems 56, 684–700.

Carrez, F., Elsaleh, T., Gomez, D., Sanchez, L., Lanza, J., Grace, P., jun 2017. A Reference Architecture for federating IoT infrastructures supporting semantic interoperability. In: 2017 European Conference on Networks and Communications (EuCNC). IEEE, pp. 1–6.

Cavalcante, E., Pereira, J., Alves, M. P., Maia, P., Moura, R., Batista, T., Delicato, F. C., Pires, P. F., 2016. On the interplay of Internet of Things and Cloud Computing: A systematic mapping study. Computer Communications 89-90, 17–33.

Chihoub, H., Collet, C., 2016. A scalability comparison study of data management approaches for smart metering systems. In: Parallel Processing (ICPP), 2016 45th International Conference on. IEEE, pp. 474–483.

Chun, S.-M., Park, J.-T., jan 2015. Mobile CoAP for IoT mobility management. In: 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC). IEEE, pp. 283–289.

Codiac, 2018. Codiac transpo sevices.
URL `http://www.codiactranspo.ca/Information/About_Codiac_Transpo.htm`

Datta, S. K., Bonnet, C., Nikaein, N., mar 2014. An IoT gateway centric architecture to provide novel M2M services. In: 2014 IEEE World Forum on Internet of Things (WF-IoT). IEEE, pp. 514–519.

Dean, J., Ghemawat, S., 2010. Mapreduce: a flexible data processing tool. Communications of the ACM 53 (1), 72–77.

Díaz, M., Martín, C., Rubio, B., 2016. State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing.

Doulkeridis, C., Vlachou, A., 2017. The datacron ontology for semantic trajectories. The Semantic Web: ESWC 2017 Satellite Events: ESWC 2017 Satellite Events, Portorož, Slovenia, May 28–June 1, 2017, Revised Selected Papers 10577, 26.

Duckham, M., 2012. Decentralized spatial computing: foundations of geosensor networks. Springer Science & Business Media.

Fortino, G., Guerrieri, A., Russo, W., Savaglio, C., may 2014. Integration of agent-based and Cloud Computing for the smart objects-oriented IoT. In: Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD). IEEE, pp. 493–498.

Gama, J., Rodrigues, P. P., 2007. Data stream processing. In: Learning from Data Streams. Springer, pp. 25–39.

Gazis, V., 2017. A survey of standards for machine-to-machine and the internet of things. IEEE Communications Surveys & Tutorials 19 (1), 482–511.

Gerla, M., Lee, E.-K., Pau, G., Lee, U., mar 2014. Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In: 2014 IEEE World Forum on Internet of Things (WF-IoT). IEEE, pp. 241–246.

Giannella, C., Han, J., Pei, J., Yan, X., Yu, P. S., 2003. Mining frequent patterns in data streams at multiple time granularities. Next generation data mining 212, 191–212.

Gupta, A., Birkner, R., Canini, M., Feamster, N., Mac-Stoker, C., Willinger, W., 2016. Network monitoring as a streaming analytics problem. In: Proceedings of the 15th ACM Workshop on Hot Topics in Networks. ACM, pp. 106–112.

Isukapati, I. K., Rudová, H., Barlow, G. J., Smith, S. F., 2017. Analysis of trends in data on transit bus dwell times. Transportation Research Record: Journal of the Transportation Research Board (2619), 64–74.

Kantarci, B., Mouftah, H. T., 2014. Mobility-aware trustworthy crowdsourcing in cloud-centric internet of things. In: Computers and Communication (ISCC), 2014 IEEE Symposium on. IEEE, pp. 1–6.

Krco, S., Pokric, B., Carrez, F., mar 2014. Designing IoT architecture(s): A European perspective. In: 2014

IEEE World Forum on Internet of Things (WF-IoT). IEEE, pp. 79–84.

Lee, G., Yun, U., Ryu, K. H., 2014. Sliding window based weighted maximal frequent pattern mining over data streams. Expert Systems with Applications 41 (2), 694–708.

Leung, C. K.-S., Cuzzocrea, A., Jiang, F., 2013. Discovering frequent patterns from uncertain data streams with time-fading and landmark models. In: Transactions on Large-Scale Data-and Knowledge-Centered Systems VIII. Springer, pp. 174–196.

Li, S., Da Xu, L., Zhao, S., 2015. The internet of things: a survey. Information Systems Frontiers 17 (2), 243–259.

Lloret, J., Tomas, J., Canovas, A., Parra, L., dec 2016. An Integrated IoT Architecture for Smart Metering. IEEE Communications Magazine 54 (12), 50–57.

López, T. S., Ranasinghe, D. C., Harrison, M., McFarlane, D., 2012. Adding sense to the internet of things. Personal and Ubiquitous Computing 16 (3), 291–308.

Luo, D., Bonnetain, L., Cats, O., Van Lint, H., 2018. Constructing spatiotemporal load profiles of transit vehicles with multiple data sources. Transportation Research Record, 0361198118781166.

Lv, Z., Song, H., Basanta-Val, P., Steed, A., Jo, M., 2017. Next-generation big data analytics: State of the art, challenges, and future research topics. IEEE Transactions on Industrial Informatics 13 (4), 1891–1899.

Mainetti, L., Patrono, L., Stefanizzi, M. L., Vergallo, R., 2015. A smart parking system based on iot protocols and emerging enabling technologies. In: Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on. IEEE, pp. 764–769.

Meehan, J., Aslantas, C., Zdonik, S., Tatbul, N., Du, J., 2017. Data ingestion for the connected world. In: CIDR.

Nelson, A., Toth, G., Hoffman, D., Nguyen, C., Rhee, S., 2017. Towards a foundation for a collaborative replicable smart cities IoT architecture. In: Proceedings of the 2nd International Workshop on Science of Smart City Operations and Platforms Engineering - SCOPE '17. ACM Press, New York, New York, USA, pp. 63–68.

Ngu, A. H., Gutierrez, M., Metsis, V., Nepal, S., Sheng, Q. Z., 2017. Iot middleware: A survey on issues and enabling technologies. IEEE Internet of Things Journal 4 (1), 1–20.

Pi, X., Egge, M., Whitmore, J., Silbermann, A., Qian, Z. S., 2018. Understanding transit system performance using avl-apc data: An analytics platform with case studies for the pittsburgh region. Journal of Public Transportation 21 (2), 2.

Puthal, D., Nepal, S., Ranjan, R., Chen, J., 2016. A secure big data stream analytics framework for disaster management on the cloud. In: High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE 18th International Conference on. IEEE, pp. 1218–1225.

Rajeshwari, U., Babu, B. S., 2016. Real-time credit card fraud detection using streaming analytics. In: Applied and Theoretical Computing and Communication Technology (iCATccT), 2016 2nd International Conference on. IEEE, pp. 439–444.

Ranasinghe, Y. S., Walpola, M. J., 2016. Integrating context-awareness with reminder tools. In: Advances in ICT for Emerging Regions (ICTer), 2016 Sixteenth International Conference on. IEEE, pp. 216–221.

Salarian, M., Manavella, A., Ansari, R., 2015. Accurate localization in dense urban area using google street view images. In: SAI Intelligent Systems Conference (IntelliSys), 2015. IEEE, pp. 485–490.

Sarkar, C., Nambi, S. N. A. U., Prasad, R. V., Rahim, A., mar 2014. A scalable distributed architecture towards unifying IoT applications. In: 2014 IEEE World Forum on Internet of Things (WF-IoT). IEEE, pp. 508–513.

Sarkar, C., Nambi S. N., A. U., Prasad, R. V., Rahim, A., Neisse, R., Baldini, G., jun 2015. DIAT: A Scalable Distributed Architecture for IoT. IEEE Internet of Things Journal 2 (3), 230–239.

Shibata, Y., Sato, G., mar 2017. IoT Based Mobility Information Infrastructure in Challenged Network Environment toward Aging Society. In: 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA). IEEE, pp. 645–648.

Somov, A., Dupont, C., Giaffreda, R., 2013. Supporting smart-city mobility with cognitive internet of things. In: Future Network and Mobile Summit (FutureNetworkSummit), 2013. IEEE, pp. 1–10.

Song, H., Srinivasan, R., Sookoor, T., Jeschke, S., 2017. Smart cities: foundations, principles, and applications. John Wiley & Sons.

Sun, W., Zhu, J., Duan, N., Gao, P., Hu, G. Q., Dong, W. S., Wang, Z. H., Zhang, X., Ji, P., Ma, C. Y., et al., 2016. Moving object map analytics: A framework enabling contextual spatial-temporal analytics of internet of things applications. In: Service Operations and Logistics, and Informatics (SOLI), 2016 IEEE International Conference on. IEEE, pp. 101–106.

Truong, H.-L., Dustdar, S., 2015. Principles for engineering iot cloud systems. IEEE Cloud Computing 2 (2), 68–76.

Vaquero, L. M., Rodero-Merino, L., 2014. Finding your way in the fog: Towards a comprehensive definition of fog computing. ACM SIGCOMM Computer Communication Review 44 (5), 27–32.

Velt, R., Benford, S., Reeves, S., 2017. A survey of the trajectories conceptual framework: investigating theory use in hci. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. ACM, pp. 2091–2105.

Verma, S., Kawamoto, Y., Fadlullah, Z. M., Nishiyama, H., Kato, N., 2017. A Survey on Network Methodologies for Real-Time Analytics of Massive IoT Data and Open Research Issues.

Wang, L., Ranjan, R., jan 2015. Processing distributed internet of things data in clouds. IEEE Cloud Computing 2 (1), 76–80.

Wu, D., Arkhipov, D. I., Asmare, E., Qin, Z., McCann, J. A., apr 2015. UbiFlow: Mobility management in urban-scale software defined IoT. In: 2015 IEEE Conference on Computer Communications (INFOCOM). IEEE, pp. 208–216.

Zhong, C., Huang, X., Arisona, S. M., Schmitt, G., Batty, M., 2014. Inferring building functions from a probabilistic model using public transportation data. Computers, Environment and Urban Systems 48, 124–137.