Developing an Analytics Everywhere Framework for the Internet of Things in Smart City Applications

Hung Cao

Doctor of Philosophy

"Ever tried. Ever failed.
NO MATTER.
Try again. Fail again.
***FAIL BETTER.**"*

Samuel Beckett
(Irish Novelist - Nobel Prize holder, 1906-1989)

Developing an Analytics Everywhere Framework for the Internet of Things in Smart City Applications

by

Hung Cao

B.Eng. in Computer Engineering, University of Information Technology, Vietnam National University Ho Chi Minh City, Vietnam, 2011
M.Sc. in Computer Science, University College Dublin, Ireland, 2015

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

In the Graduate Academic Unit of Geodesy & Geomatics Engineering

Supervisor(s):	Monica Wachowicz, Ph.D., Geodesy & Geomatics Engineering
Examining Board:	David Coleman, Ph.D., Geodesy & Geomatics Engineering Wei Song, Ph.D., Computer Science Yun Zhang, Ph.D., Geodesy & Geomatics Engineering
External Examiner:	Xin Wang, Ph.D., Department of Geomatics Engineering, Schulich School of Engineering, University of Calgary

This dissertation is accepted by the

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

September, 2019

Abstract

Despite many efforts on developing protocols, architectures, and physical infrastructures for the Internet of Things (IoT), previous research has failed to fully provide automated analytical capabilities for exploring IoT data streams in a timely way. Mobility and co-location, coupled with unprecedented volumes of data streams generated by geo-distributed IoT devices, create many data challenges for extracting meaningful insights. This research work aims at exploring an edge-fog-cloud continuum to develop automated analytical tasks for not only providing higher-level intelligence from continuous IoT data streams but also generating long-term predictions from accumulated IoT data streams. Towards this end, a conceptual framework, called *"Analytics Everywhere"*, is proposed to integrate analytical capabilities according to their data life-cycles using different computational resources. Three main pillars of this framework are introduced: resource capability, analytical capability, and data life-cycle. First, resource capability consists of a network of distributed compute nodes that can handle automated analytical tasks either independently or in parallel, concurrently or in a distributed manner. Second, analytical capability orchestrates the execution of algorithms to perform streaming descriptive, diagnostic, and predictive analytics. Finally, data life-cycles are designed to manage both continuous and accumulated IoT data streams. The research outcomes from a smart parking and a smart transit scenario have confirmed that a single computational re-

source is not sufficient to support all analytical capabilities that are needed for IoT applications. Moreover, the implemented architecture relied on an edge-fog-cloud continuum and offered some empirical advantages: (1) on-demand and scalable storage; (2) seamlessly coordination of automated analytical tasks; (3) awareness of the geo-distribution and mobility of IoT devices; (4) latency-sensitive data life-cycles; and (5) resource contention mitigation.

To the two loves of my life:

My wife, Phuong Nguyen & My son, Louis Cao.

Acknowledgements

This Ph.D. dissertation has been co-funded by NSERC, Cisco System, and UNB. This work would also not have been possible without the great support from the Codiac Transpo, HotSpot Parking company, the City of Moncton, and the City of Saint John, NB, Canada for allowing me to access to their data streams. I would also like to thank Compute Canada for providing me with the cloud computing resources. My special thanks are extended to The Black Arcs and Rimot.io Inc. for many other research projects that I have been involved in with these companies. During my time at UNB, there are a number of people who have influenced, encouraged, and supported me that I would like to take this chance to render my thankfulness.

First and foremost, I would like to acknowledge my indebtedness and express my warmest gratitude to my supervisor, Dr. Monica Wachowicz, for her invaluable advice, lessons, and guidance. The past 3 years have been very challenging which resulted in times where I wanted to give up. However, Dr. Monica Wachowicz gave me great motivation, encouragement, patience, and enthusiastic support during my toughest time. I believe that I would not be the person I am today without her exceptional assistance.

I would like to send my sincere thanks to Dr. Chiara Renso from the HPC Lab, CNR, Italy for her assistance and mentorship. In practice, Dr. Chiara Renso

was my co-supervisor during my Ph.D. research. Her valuable and constructive suggestions during the planning and development of my research proposal helped me establish a very clear direction from the beginning. Thoughtful comments and discussions, given by Dr. Emanuele Carlini also from the HPC Lab, CNR, Italy, have been a great help in publishing our second journal paper.

My grateful thanks are also extended to the rest of my oral examination committee: Dr. David Coleman, Dr. Wei Song, Dr. Yun Zhang, and Dr. Xin Wang for their insightful comments, expert advice, and tough questions. I would also wish to express my very great appreciation to MSc. Susan Kingdon who generously spent many hours to proofread my dissertation. Many thanks to the GGE staff, especially Ms. Alica Farnham and Ms. Lorry Hunt, for their administrative assistance.

I am particularly grateful to my colleagues, lab-mates for many interesting discussions and my friends for helping me for many years. I also offer my deepest regards and blessings to my parents, my brother, and my family who gave me love and unconditional support during my life. My sister in law provided me with tremendous assistance during our time living in Canada which I greatly appreciated.

Last but not least, nobody has been more important to me in the pursuit of this Ph.D. journey than my little family. I am wholeheartedly grateful to my loving and supportive wife, Phuong Nguyen, who always stays behind me during the stressful and toughest times in my life. The person with the greatest indirect contribution to this work is my son, Louis Cao. He gave me the greatest moment ever for being his father and sharing many cherishable times together that I could never forget.

Hung Cao

University of New Brunswick

September 2019

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AMQP	Advanced Message Queuing Protocol
APIs	Application Programming Interfaces
APUs	Accelerated Processing Units
ARF	Adaptive Random Forest
ASICs	Application Specific Integrated Chips
BLE	Bluetooth Low Energy
CEP	Complex Event Processing
CoAP	Constrained Application Protocol
CPUs	Central Processing Units
DAG	Directed Acyclic Graphs
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DCM	Data Control Module
DDS	Data Distribution Service
DTSL	Datagram Transport Layer Security
EFM	Edge & Fog Processing Module
FPGAs	Field Programmable Gate Arrays
GIS	Geographic Information Systems
GPS	Global Positioning System
GPUs	Graphics Processing Units
GMM	Gateway Management Module

HAC	Hierarchical Agglomerative Clustering
HMM	Hidden Markov Model
HTTP	HyperText Transfer Protocol
IaaS	Infrastructure as a Service
I/O	Input/Output
IoT	The Internet of Things
JDBC	Java Database Connectivity
IPSec	Internet Protocol Security
JSON	JavaScript Object Notation
LPWAN	Low-Power Wide Area Network
LTE	Long-Term Evolution
mMTC	massive Machine-Type Communications
MQTT	Message Queuing Telemetry Transport
NAT	Network Address Translation
NB-IoT	Narrowband Internet of Things
NIST	National Institute of Standards and Technology
ODBC	Open Database Connectivity
OS	Operating System
PaaS	Platform as a Service
PCA	Priciple Component Analysis
QoS	Quality of Service
REST	REpresentational State Transfer
RFID	Radio-Frequency IDentification
Saas	Software as a Service
SoC	System-on-Chip
STOMP	Streaming Text Oriented Messaging Protocol
TCP	Transmission Control Protocol

TLS/SSL	Transport Layer Security/Secure Sockets Layer
UDP	User Datagram Protocol
uMTC	ultra-reliable Machine-Type Communications
WLAN	Wireless Local Area Network
WF-nets	WorkFlow nets
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network
XaaS	Everything as a Service
xMBB	extreme Mobile BroadBand
XMPP	Extensible Messaging and Presence Protocol

Chapter 1

Introduction

The main technological drives for carrying on this research can be described as being the Internet of Things (IoT), Geographic Information Systems (GIS), and the synergy of edge, fog, and cloud computing. They play an important role in the conceptualization and implementation of streaming analytical capabilities and in the data life-cycle of automated analytical workflows for IoT applications. They are further discussed in the next sections.

1.1 Background

1.1.1 Internet of Things

The fast growth of IoT sensors, physical infrastructures, protocols, and IoT applications is currently considered to be the first phase of a technological revolution in smart cities (Al-Fuqaha et al., 2015). The *"Things"* are usually *"physical devices"* which can sense their surroundings and interact among themselves without human in-

tervention (López et al., 2012). The IoT enables traditional things to be transformed into embedded sensors within an environment or into wearable things carried by us (Mukhopadhyay, 2014; Phillips et al., 2017). These items rely on sensor networks that are supported by communication technologies such as Bluetooth, WiFi, ZigBee, RFID, and LTE (Lee et al., 2007; Crosby and Vafa, 2013; Yang et al., 2014), Internet protocols such as REST, HTTP, MQTT, AMQP (Naik, 2017; Thangavel et al., 2014), identification technologies such as IPv4, IPv6, uCode (Deering and Hinden, 2017; Koshizuka and Sakamura, 2010), and pervasive computing and applications (Kiljander et al., 2014).

The IoT data streams generated by IoT devices have fast-incoming data rates that are usually described as unbounded sequences of time-varying data tuples (Cao et al., 2017; Bonte et al., 2018; Almuammar and Fasli, 2018). Extracting useful insights from these data streams is a non-trivial task, especially when it comes to the IoT applications with continuous data streams (high velocity) and accumulated data streams (high volume). Some examples of IoT applications include fleet management (Xu et al., 2019), traffic monitoring (Barthélemy et al., 2019), healthcare (Abdellatif et al., 2018), manufacturing (Chen et al., 2018), agriculture (Elijah et al., 2018), and smart grids (Ozger et al., 2018). For these IoT applications, the efficient retrieval and processing of IoT data streams are important requirements that demand *(i) consideration of computing power, storage capability, communication capability, analytical capability, and energy for analyzing IoT data streams;* and *(ii) generating useful and higher-level information in a timely way before it becomes outdated for supporting IoT applications.*

Two phases can be distinguished in the evolution of IoT. The first phase has focused on the proliferation of sensors, protocols, and architectures, where the main research challenges were related to network connectivity, IoT platforms, and sensor

configurations. A second phase is gradually taking place where the core research challenges are shifting from physical infrastructures to analytical capabilities that are being developed according to the requirements of IoT applications (Marjani et al., 2017). This research is focused on addressing the challenges of developing analytical capabilities for IoT applications.

1.1.2 IoT and Geographic Information Systems

The stream data collected by IoT devices might contain spatial information (e.g. position, altitude, distance), temporal information (e.g. event time, processing time), or thematic information (e.g. temperature, humidity, speed, concentration, pressure) about their surrounding environments. Contextualizing the huge volume of IoT data streams with automated tagging that contains geographical information (e.g. proximity, topology, connectivity) and other contextual information about an environment, event or a phenomena, can significantly improve the results of analytical workflows, especially when the analytical tasks are automated. Additional correlations, patterns, and trends can be revealed using the spatial functionalities of GIS.

However, IoT-GIS platforms are being newly developed, and developers usually struggle with integrating the data acquired from multiple sources such as Things, Models, Virtual Objects, and Real Objects (Isikdag, 2015). Current GIS platforms do not allow efficient processing and storage of IoT data streams. Therefore, the research challenge is to develop IoT-GIS platforms that can handle processing of a variety of IoT data streams, which will be generated by smart cities in the near future (Pavlík et al., 2018). Specifically, an IoT-GIS platform is required (*i) to consume and process massive volumes of stream data;* and *(ii) to have the ability to interoperate with other platforms and IoT devices through open interfaces and*

standards. There is an increasing trend to shift from the traditional desktop-based GIS platform to cloud-based solutions in order to handle the volume of IoT data received. However, the newly developed cloud-GIS models will face many challenges in terms of performing analytical tasks without human intervention.

Miller and Goodchild (2015) emphasize that it is paramount to transform geospatial research to explore new forms of data and their respective new representations of real-world phenomena. Some of the challenges will be new, whilst others are longstanding in geospatial research and will be intensified by the large volume of IoT data. This research takes a step in this direction by exploring the challenges of building IoT-GIS platforms that must be developed for analyzing the exponential number of new forms of IoT data. These new forms of IoT data will be generated continuously and offer far greater spatial granularity.

1.1.3 Cloud Computing

According to the NIST definition (Mell et al., 2011), *cloud computing* is a paradigm that can provide pervasive, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, memory, applications, and services) which are maintained and controlled by service provider with minimal direct active management effort from the user. In the current literature, the cloud computing environment is the most preferred choice for processing and analyzing IoT data streams since it can provide virtually unlimited and on-demand resources, and scalability for storing IoT data streams. The cloud computing environment also supports complex operations. Due to its flexibility and efficient resource provisioning, previous research has been focused on finding a solution to support IoT virtualization features, alongside straightforward IaaS (computing and storage) virtualization in the cloud (Yu et al., 2018; Bruneo et al., 2018; Paulraj et al., 2018; Mekala and

Viswanathan, 2019; Alhussein et al., 2018).

Stream processing in the cloud is not new. Literature shows that most of the existing stream processing platforms follow one of the two typical cloud architectures: Lambda or Kappa Architecture (Wingerath et al., 2016). Lambda Architecture was first introduced by Marz and Warren (2015). Its core principle when handling the data streams is to combine both batch and stream processing systems to execute different paths of computation. These computation paths are always running in parallel in order to guarantee that we have a streaming fast path for timely approximate results, and a batch offline path for late accurate results. The advantage of this architecture is to manage the historical data to assure a low possibility of errors even if the system crashes. However, this approach introduces high latency imposed by every batch processing cycle. It also suffers from high complexity and is difficult to migrate or reorganize due to the involvement of comprehensive processing. For example, it is difficult for Lambda Architecture to successfully interact within several systems, and there is a high coding overhead.

In contrast, Kappa Architecture dispenses with the batch processing systems in favor of simplicity (Kreps, 2014). In this case, all computations are performed in a stream processing system alone and batch programs are treated as a special case of streaming programs. They only perform re-computation when the stream is bounded (fixed) by replaying historical data. This architecture finds its application in the processing of distinct events where the order and the event time does not matter since all data tuples implicitly belong to one all-encompassing time window. However, the absence of a batch processing system might result in errors during data processing. Also, the attempt to replay the entire historical data might create huge pressure on the storage requirements of a streaming system rather than periodically processing the new data and updating the existing data storage.

1.1.4 Edge Computing

It is important to point out that the mobility of IoT devices poses many challenges in pulling and pushing the data streams from these devices to remote clouds, independently of using a Lambda or a Kappa architecture. In fact, Lu et al. (2014) points out that one critical challenge in building the next generation of intelligent transportation systems is related to the harsh communication environments inside and/or outside of a moving vehicle. Solutions for vehicle-to-sensor, vehicle-to-vehicle, and vehicle-to-road infrastructure connectivity are stringently dependent on latency and reliability for controlling and monitoring purposes.

Moreover, moving IoT devices usually require seamless computation, storage, and connection services over a vast geographical area (i.e. entire transit system), which create challenges in the communication network used to transport IoT data streams between sensors in a vehicle and the core network. The main issues are related to data becoming unreliable and error-prone as well as the requirement of an extensive amount of storage (Nahrstedt et al., 2016). Furthermore, cloud computing is also facing increasing challenges in supporting the latency-sensitive, geo-distribution, and situational-awareness of IoT devices. Transporting the data streams from remote clouds to local applications becomes expensive because it requires a huge amount of bandwidth, time, and energy.

Bearing in mind the mobility of IoT devices, another streaming architecture has been proposed in the literature with the introduction of *edge computing* for decreasing network congestion and latency (Cortés et al., 2015; Shi et al., 2016). From the literature, the exact definitions of edge computing remain an ongoing discussion in academia (Shi et al., 2016; Varghese et al., 2016; Satyanarayanan, 2017b). In the context of this study, edge computing refers to the enabling technologies al-

lowing substantial computing and storage resources are located at the edge of the network in close proximity to mobile IoT devices or sensors (i.e. gateways one-hop away from an IoT device). Preliminary results are positive in alleviating the saturation of network bandwidth, as well as accelerating automated analytical tasks, all of which allows real-time decision making to become possible. More recently, edge computing is receiving more attention, and more research efforts can be found in the literature to exploit this computing paradigm for IoT applications (Nastic et al., 2017; Patel et al., 2017; Satyanarayanan, 2017a,b). However, very few architectural proposals can be found in the literature on edge computing. One example is IRESE (Janjua et al., 2019) which is a model that relies on the edge device to perform data stream analytics to detect various types of rare-events using two unsupervised clustering algorithms. Another example is GeeLytics (Cheng et al., 2015), which is a geo-distributed edge analytics platform that is still in the early design stage to support dynamic processing topologies. A final example is StreamBox-TZ (Park et al., 2019), a secure stream analytics engine used for isolating data streams and their computations in a trusted execution environment at the edge.

1.1.5 Fog Computing

From an analytical perspective, the combination of edge and cloud computing still has some limitations since edge nodes are lightweight with low processing and storage capabilities, which results in resource contention and increase processing latency (Yu et al., 2017). Dastjerdi and Buyya (2016) proposed *fog computing* as an intermediate resource that can seamlessly integrate edge and cloud resources. Fog computing can eliminate resource contention at the edge by supporting several analytical tasks at the fog nodes and by coordinating the use of geographically distributed IoT devices more efficiently than in the cloud.

When fog computing was first being introduced by Cisco (Bonomi et al., 2014), it was envisaged to apply in many applications, such as Smart Grids, Wind Farm, Healthcare, Industrial Automation, Smart Traffic Lights, and Connected Vehicles. It is originally defined as a highly virtualized resource to leverage untapped processing power in network middleboxes (e.g. routers, rack servers) that provides compute, storage, and networking services between the edge and cloud computing environment when those middleboxes are either over-provisioned or not running at full load. In the past few years, many applications have been realized in the fog computing environment. For example, Santoro et al. (2017) developed a platform based on open source technologies working in a fog computing environment called Foggy, which allows infrastructure owners and tenants to offer the functionality of negotiation, scheduling and workload orchestration, and diversified constraints on location and access rights. Seitz et al. (2017) introduced FRODO, which is a fog architecture created to establish a location-aware environment for conflict negotiation, discussion, and decision support to satisfy the individual preferences in smart buildings. Recently, Fog-IBDIS (Wang et al., 2019) was created, which is an industrial platform that is capable of integrating and sharing large amounts of data across all operations of manufacturing systems. This platform has been implemented by integrating the generated data of a commercial aircraft-manufacturing process in the fog clients within the manufacturing systems.

1.1.6 IoT Frameworks

Vast amounts of incoming IoT data streams generated by many geo-distributed devices create a huge challenge for ingesting and analyzing them at a high data rate. From the literature, there are over 400 architectures and frameworks that can be found to handle the incoming IoT data streams using different strategies such as

stream, *micro-batch*, and *batch processing* (Wingerath et al., 2016; Cao and Wachowicz, 2017).

Batch-oriented processing frameworks have been efficiently used for processing massive amounts of historical IoT data with high throughput but also with high latency. Aiming to increase efficiency, *micro-batch processing frameworks* buffer and process IoT data streams in many mini-batches. However, it will obviously increase the time that the data streams spend in the data pipeline. In contrast, *stream-oriented processing frameworks* typically provide short-time computations but have relatively high data processing costs on a continuous stream of IoT data. Stream-oriented processing architectures usually avoid putting data at rest. Instead, they minimize the time a single data tuple should spend in a processing pipeline.

From an analytics perspective, a paradigm shift has emerged recently in the evolution of IoT architectures aiming at software, analytics, and platform configuration. Streaming analytics algorithms are being developed to extract knowledge and insights from IoT data streams as soon as they arrive at a specific computational resource. However, it is challenging to extract value online, since the nature (or distributions) of IoT data streams change over time due to the geographical location of IoT devices (De Francisci Morales et al., 2016). In addition, streaming analytical algorithms are normally required to work within limited resources (time and memory). Some open-source frameworks for IoT data stream analytics are being developed including MOA, SAMOA and skit-multiflow (Montiel et al., 2018; Morales and Bifet, 2015; Bifet et al., 2011) using only streaming processors. However, these frameworks are still leveraging the cloud computing paradigm for handling the analytical tasks, since IoT data streams are fetched to and accumulated in the cloud over a long period of time and are later analyzed in batches using traditional machine learning and data mining algorithms.

The proposed architecture implemented in the next chapters is a step forward in finding a unique solution that combines the advantages of different computational resources into an integrated edge-fog-cloud fabric that is capable of capturing, managing, processing, analyzing and visualizing IoT data streams. This fabric of computational resources is designed to work towards an asynchronous approach for supporting an *"Analytics Everywhere"* framework making the development, deployment and maintenance more pragmatic and scalable. By breaking down the processing and analytical capabilities into a network of streaming tasks and distributing them into different compute nodes in an edge-fog-cloud continuum based on a predefined data life-cycle, our proposed architecture can support streaming descriptive, diagnostic and predictive analytics. For example, some predictive analytical tasks can be executed in the cloud while diagnostic analytical tasks can be performed online (on-the-fly) at an edge or fog node.

1.1.7 Summary

In this research, the edge-fog-cloud continuum is proposed based on the research premise that a single computational resource, such as the cloud, will be not sufficient to support all analytical tasks that are required to be automated for handling IoT data streams in a timely way. These current technologies allow us to rethink new approaches that can cope with the following challenges:

- Structured/Semi-structured/Unstructured IoT data streams will be constantly generated by geo-distributed devices in smart cities, requiring many streaming automated analytical tasks that can manage, process, and retrieve high velocity, variety, and volumes of data.
- Mobility and co-location of IoT devices in smart cities will require short range

communication networks as well as long range communication networks for transporting IoT data streams, avoiding bottlenecks, and reducing data latency.

All of these challenges are leading to the consideration of a new approach for extracting knowledge and insights from IoT data streams. This is further discussed in the next section.

1.2 Research Questions

The proposed *"Analytics Everywhere"* framework distributes analytical tasks as the IoT data streams are being transported through an edge-fog-cloud continuum that combines analytical tasks for supporting processing off the cloud, saving in data transfer, and analyzing data streams closer to the IoT devices. This new approach will directly require automated tasks and raise crucial research questions such as the following:

Q1. How can automated analytical tasks be developed for supporting analytical capabilities such as streaming descriptive/diagnostics/predictive algorithms/methods?

Q2. How can continuous and accumulated streams be combined while taking into account different IoT data life-cycles?

Q3. How can IoT and GIS be integrated into the edge-fog-cloud continuum without compromising resource capabilities?

Q4. What are the benefits and limitations of the proposed *"Analytics Everywhere"* framework in smart cities?

1.3 Research Objectives

The proposed *"Analytics Everywhere"* framework aims to achieve both low latency and low complexity by orchestrating several resource capabilities into a unique and seamless processing architecture. The main advantage is that users have the control to define user-defined time windows and geographical areas of interest. Additionally, arbitrary inaccuracy of the analytical results can be significantly alleviated because the *"Analytics Everywhere"* framework explicitly monitors the data life-cycles. This ensures that each automated analytical task has the right data at the right time. The main limitation, however, is that there is no guarantee that data tuples belonging to a stream are being handled, at most once, by a streaming task of an IoT application.

The measurable objectives of this research work can be described as follows:

(O1): Identify the algorithms/methods that can be used for supporting automated streaming analytical tasks (analytical capability).

(O2): Identify the off-the-shelf tools that can be used to implement the proposed framework (computational resources).

(O3): Identify the potential IoT applications in smart cities while taking into account the availability of IoT data streams.

(O4): Develop data life-cycles for executing automated analytical tasks and coping with continuous IoT data streams.

(O5): Develop data life-cycles for executing automated analytical tasks and coping with accumulated IoT data streams.

(O6): Build the architecture for this new framework based on a continuum of edge-fog-cloud nodes.

(O7): Implement the *"Analytics Everywhere"* framework in real-world experiments to evaluate the proposed approach. Two experiments were selected according to data availability: smart transit and smart parking.

(O8): Implement the data life-cycles for the real-world experiments.

(O9): Validate the proposed *"Analytics Everywhere"* framework.

1.4 Scientific Contributions

This research work proposes a conceptual *"Analytics Everywhere"* framework based on an edge-fog-cloud continuum to handle the volume, velocity, and variety of incoming data streams from IoT devices. This proposed framework performs automated analytical tasks using complementary data life-cycles and resource capabilities such as edge, fog, and cloud computing. Three main components of this framework are resource capability, analytical capability, and data life-cycle. Firstly, resource capability comprises a network of distributed compute nodes that provide different computing resources (e.g. I/O, storage, computation, and processing power) for every analytical task. The compute nodes are usually deployed covering a large geographical area, and they can be static (i.e. a fog node deployed inside a building) or dynamic (i.e. an edge node deployed in a car). The core hardware of the compute nodes could be one or a combination of several processing units such as Graphics Processing Units (GPUs), Central Processing Units (CPUs), Accelerated Processing Units (APUs), Application Specific Integrated Chips (ASICs), Field Programmable Gate Arrays (FPGAs), or System-on-Chip (SoC) accelerators. These compute nodes can handle analytical tasks either independently or in parallel and have either concurrent or distributed styles. A network of distributed compute nodes is a noticeable contribution, since the past studies related to analyzing IoT data streams tend to

stick to one type of computational resource (i.e. cloud) or a two-tier topology consisting of edge and cloud nodes.

Secondly, analytical capability presents the best practice methods/algorithms. These methods/algorithms need to be automated to execute analytical tasks such as pre-processing, contextualization, and spatio-temporal analysis tasks, which are needed for supporting descriptive, diagnostic, and predictive analytics. I have classified IoT data streams into two types: accumulated IoT data streams and continuous IoT data streams. Accumulated IoT data streams are the IoT data streams that are transmitted to a compute node and are accumulated there over a period of time until an automated analytical task is triggered or finalized. For example, an automated contextualization task will require the accumulation of IoT data streams according to a time-window in order to provide a mobility context. This mobility context helps to explain phenomena, to reinforce different perspectives, and to provide an accurate understanding of the background surrounding the problem in an IoT application. In contrast, continuous IoT data streams are the IoT data streams that have constantly incoming data tuples, which are bouncing from one compute node to another. These data tuples are required to be processed or analyzed immediately at each hop. This is particularly the case for automated pre-processing tasks such as cleaning, filtering, and querying. To the best of my knowledge, my proposed framework is the first attempt to integrate different analytical capabilities using both accumulative and continuous IoT data streams, which can be transported and analyzed via an edge-fog-cloud continuum using a variety of tools such as Cisco Kinetic, GIS, Python libraries, geospatial databases (e.g. PostgreSQL, MongoDB, RethinkDB), Hadoop MapReduce, and GTFS packages.

Lastly, a data life-cycle describes the process that both continuous and accumulated IoT data streams go through during the automated execution of ana-

lytical tasks. The actual data-life cycle process will depend on the sequence of the analytical tasks designed to support an IoT application. We expect that different IoT applications will require specific data life-cycle processes. However, determining how to map different analytical capabilities with the most appropriate computational resources based on a data life-cycle of an IoT application is far from being a trivial endeavour, since several aspects must be taken into account. Not all analytical tasks can run on all compute nodes due to the complexity of learning paradigms that currently exist such as deep learning, on-line learning, local learning, and anticipatory learning, to mention a few. Moreover, it is important to point out that an *"Analytics Everywhere"* framework will have limitations. Real-world IoT applications will play an important role in providing empirical evidence to validate and improve such a framework. Developing automated streaming analytics for IoT data streams is still in its infancy, and applications usually require a diverse number of data life-cycles having different temporal granularities. There has been very little research reported on the impact of analytical tasks in the IoT architectures. The scientific contribution of this research is therefore to ascertain this impact, using real-world scenarios such as smart parking and smart transit.

On a final note, it is worth mentioning that the proposed conceptual framework is not the one-size-fits-all approach. At this moment in the evolution of IoT, it is not yet possible to envisage a framework that can fit all IoT applications. The empirical advantage of the proposed conceptual framework is that it allows users to develop, test, debug, operate, and manage their IoT applications on top of a single analytical framework. I do not claim that the proposed architecture will create more efficiency than other architectures. Instead, there is a set of trade-offs of desideratum such as high availability, low latency, high throughput, low cost, data distribution, data reliability, precise semantics, easy maintenance, processing flexibility, and platform openness for the developers to choose, depending on a particular

IoT application.

1.5 Structure of Dissertation

This article-based dissertation presents a collection of three scientific papers that have been assessed via the peer review processes.

1. **Cao, H.**, Wachowicz, M., Renso, C., & Carlini, E. (2019). Analytics Everywhere: generating insights from the Internet of Things. *IEEE Access*, *7*, 71749-71769. (**Peer Reviewed** - Impact Factor: 4.098)

2. **Cao, H.**, & Wachowicz, M. (2019). An Edge-Fog-Cloud Architecture of Streaming Analytics for Internet of Things Applications. Special Issue Edge/-Fog/Cloud Computing in the Internet of Things. *Sensors*, *19*(16), 3594. (**Peer Reviewed** - Impact Factor: 3.031)

3. **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. (**Peer Reviewed** - Impact Factor: 3.393)

Table 1.1 illustrates the main contents and objectives that have been achieved in each article in this dissertation. The remainder of this dissertation is organized and summarized as follows:

- *Chapter 2 (Article 1)* includes the answers for research questions **Q1.**, **Q2.**, and **Q4.** by addressing objectives **(O1)**, **(O2)**, **(O3)**, **(O5)**, **(O6)**, **(O7)**, **(O8)**, and **(O9)**. This chapter proposes an *"Analytics Everywhere"* framework that can integrate a variety of computational resources including edge, fog, and

Table 1.1: The summary of the structure of dissertation.

	Short Description	Article 1	Article 2	Article 3
Objectives	**(O1):** Analytical Capability	✓	✓	✓
	(O2): Computational Capability	✓	✓	✓
	(O3): IoT Applications	✓	✓	✓
	(O4): Data Life-cycles & coping with continuous IoT data streams.		✓	
	(O5): Data Life-cycles & coping with accumulated IoT data streams.	✓	✓	✓
	(O6): Continuum of edge-fog-cloud	✓	✓	
	(O7): Real-world experiments: smart transit & smart parking	✓	✓	✓
	(O8): Data life-cycles for the real-world experiments	✓	✓	
	(O9): Validate the proposed Analytics Everywhere framework	✓	✓	✓

cloud nodes (addressing objectives **(O2)** and **(O6)**) and analytical capabilities such as descriptive, diagnostic, and predictive analytics (addressing objective **(O1)**), according to a data life-cycle (addressing objectives **(O5)** and **(O8)**). I demonstrate the effectiveness of my proposed framework using an application in smart transit (addressing objectives **(O3)**, **(O7)**, and **(O9)**).

- *Chapter 3 (Article 2)*, as an evolution of *Chapter 2*, is meant to answer the same research questions (**Q1.**, **Q2.**, and **Q4.**) by tackling all research objectives mentioned in Table 1.1. In this chapter, an architecture based on the edge-fog-cloud continuum is proposed (addressing objectives **(O2)** and **(O6)**) to analyze IoT data streams for delivering data-driven insights (addressing objective **(O1)**) in a smart parking scenario in the City of Saint John, NB, Canada (addressing objectives **(O3)**, **(O7)**, and **(O9)**). Two data life-cycles are developed and implemented to carry out the analytical tasks in the contin-

uous data streams and accumulated data streams (addressing objectives **(O4)**, **(O5)**, and **(O8)**) that are constantly generated by IoT devices.

- *Chapter 4 (Article 3)* mainly aims to resolve the research question **Q3.**. As highlighted in Section 1.1.2, GIS plays an important role in the insights discovery process from IoT data streams. However, integrating IoT and GIS into the *"Analytics Everywhere"* framework is a non-trivial research work. Some objectives, including objectives **(O1)**, **(O2)**, **(O3)**, **(O5)**, **(O7)**, and **(O9)**, have been achieved during the process of conducting this research work. This chapter presents my design for an IoT-GIS platform (addressing objective **(O2)**) that performs automated analytical tasks (addressing objective **(O5)**). These tasks are able to retrieve, integrate, and contextualize data streams from the Internet of Moving Things without human intervention (addressing objective **(O1)**). My proposed platform has been validated via the realtime data collected from the transit service provided by the Codiac Transit System of the Greater Moncton area, NB, Canada (addressing objectives **(O3)**, **(O7)**, and **(O9)**).

- Finally, *Chapter 5* concludes the dissertation by summarizing the key findings in each chapter and by highlighting the main contributions of this research as well as discussing some limitations and suggesting future work.

References

Abdellatif, A. A., Khafagy, M. G., Mohamed, A., and Chiasserini, C.-F. (2018). Eeg-based transceiver design with data decomposition for healthcare iot applications. *IEEE Internet of Things Journal*, 5(5):3569–3579.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015).

Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376.

Alhussein, M., Muhammad, G., Hossain, M. S., and Amin, S. U. (2018). Cognitive iot-cloud integration for smart healthcare: Case study for epileptic seizure detection and monitoring. *Mobile Networks and Applications*, 23(6):1624–1635.

Almuammar, M. and Fasli, M. (2018). Learning patterns from imbalanced evolving data streams. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2048–2057. IEEE.

Barthélemy, J., Verstaevel, N., Forehead, H., and Perez, P. (2019). Edge-computing video analytics for real-time traffic monitoring in a smart city. *Sensors*, 19(9):2048.

Bifet, A., Holmes, G., Pfahringer, B., Read, J., Kranen, P., Kremer, H., Jansen, T., and Seidl, T. (2011). Moa: a real-time analytics open source framework. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 617–620. Springer.

Bonomi, F., Milito, R., Natarajan, P., and Zhu, J. (2014). Fog computing: A platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*, pages 169–186. Springer.

Bonte, P., Tommasini, R., Della Valle, E., De Turck, F., and Ongenae, F. (2018). Streaming massif: Cascading reasoning for efficient processing of iot data streams. *Sensors*, 18(11):3832.

Bruneo, D., Distefano, S., Longo, F., Merlino, G., and Puliafito, A. (2018). I/ocloud: Adding an iot dimension to cloud infrastructures. *Computer*, 51(1):57–65.

Cao, H. and Wachowicz, M. (2017). The design of a streaming analytical workflow for processing massive transit feeds. In *The 2nd International Symposium on Spatiotemporal Computing*.

Cao, H., Wachowicz, M., and Cha, S. (2017). Developing an edge computing platform for real-time descriptive analytics. In *Big Data (Big Data), 2017 IEEE International Conference on*, pages 4546–4554. IEEE.

Chen, B., Wan, J., Celesti, A., Li, D., Abbas, H., and Zhang, Q. (2018). Edge computing in iot-based manufacturing. *IEEE Communications Magazine*, 56(9):103–109.

Cheng, B., Papageorgiou, A., Cirillo, F., and Kovacs, E. (2015). Geelytics: Geo-distributed edge analytics for large scale iot systems based on dynamic topology. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 565–570. IEEE.

Cortés, R., Bonnaire, X., Marin, O., and Sens, P. (2015). Stream processing of healthcare sensor data: studying user traces to identify challenges from a big data perspective. *Procedia Computer Science*, 52:1004–1009.

Crosby, G. V. and Vafa, F. (2013). Wireless sensor networks and lte-a network convergence. In *38th Annual IEEE conference on local computer networks*, pages 731–734. IEEE.

Dastjerdi, A. V. and Buyya, R. (2016). Fog computing: Helping the internet of things realize its potential. *Computer*, 49(8):112–116.

De Francisci Morales, G., Bifet, A., Khan, L., Gama, J., and Fan, W. (2016). Iot big data stream mining. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2119–2120. ACM.

Deering, S. and Hinden, R. (2017). Internet Protocol, Version 6 (IPv6) Specification. Technical report.

Elijah, O., Rahman, T. A., Orikumhi, I., Leow, C. Y., and Hindia, M. N. (2018). An overview of internet of things (iot) and data analytics in agriculture: Benefits and challenges. *IEEE Internet of Things Journal*, 5(5):3758–3773.

Isikdag, U. (2015). Bim and iot: A synopsis from gis perspective. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40:33.

Janjua, Z. H., Vecchio, M., Antonini, M., and Antonelli, F. (2019). Irese: An intelligent rare-event detection system using unsupervised learning on the iot edge. *Engineering Applications of Artificial Intelligence*, 84:41–50.

Kiljander, J., D'elia, A., Morandi, F., Hyttinen, P., Takalo-Mattila, J., Ylisaukko-Oja, A., Soininen, J.-P., and Cinotti, T. S. (2014). Semantic interoperability architecture for pervasive computing and internet of things. *IEEE access*, 2:856–873.

Koshizuka, N. and Sakamura, K. (2010). Ubiquitous id: standards for ubiquitous computing and the internet of things. *IEEE Pervasive Computing*, (4):98–101.

Kreps, J. (2014). Questioning the lambda architecture. *Online article, July*.

Lee, J.-S., Su, Y.-W., Shen, C.-C., et al. (2007). A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi. *Industrial electronics society*, 5:46–51.

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

Lu, N., Cheng, N., Zhang, N., Shen, X., and Mark, J. W. (2014). Connected vehicles: Solutions and challenges. *IEEE internet of things journal*, 1(4):289–299.

Marjani, M., Nasaruddin, F., Gani, A., Karim, A., Hashem, I. A. T., Siddiqa, A., and Yaqoob, I. (2017). Big iot data analytics: architecture, opportunities, and open research challenges. *IEEE Access*, 5:5247–5261.

Marz, N. and Warren, J. (2015). *Big Data: Principles and best practices of scalable real-time data systems*. New York; Manning Publications Co.

Mekala, M. S. and Viswanathan, P. (2019). Clay-mist: Iot-cloud enabled cmm index for smart agriculture monitoring system. *Measurement*, 134:236–244.

Mell, P., Grance, T., et al. (2011). The nist definition of cloud computing.

Miller, H. J. and Goodchild, M. F. (2015). Data-driven geography. *GeoJournal*, 80(4):449–461.

Montiel, J., Read, J., Bifet, A., and Abdessalem, T. (2018). Scikit-multiflow: a multi-output streaming framework. *The Journal of Machine Learning Research*, 19(1):2915–2914.

Morales, G. D. F. and Bifet, A. (2015). Samoa: scalable advanced massive online analysis. *Journal of Machine Learning Research*, 16(1):149–153.

Mukhopadhyay, S. C. (2014). Wearable sensors for human activity monitoring: A review. *IEEE sensors journal*, 15(3):1321–1330.

Nahrstedt, K., Li, H., Nguyen, P., Chang, S., and Vu, L. (2016). Internet of mobile things: Mobility-driven challenges, designs and implementations. In *Internet-of-Things Design and Implementation (IoTDI), 2016 IEEE First International Conference on*, pages 25–36. IEEE.

Naik, N. (2017). Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In *2017 IEEE international systems engineering symposium (ISSE)*, pages 1–7. IEEE.

Nastic, S., Rausch, T., Scekic, O., Dustdar, S., Gusev, M., Koteska, B., Kostoska, M., Jakimovski, B., Ristov, S., and Prodan, R. (2017). A serverless real-time data analytics platform for edge computing. *IEEE Internet Computing*, 21(4):64–71.

Ozger, M., Cetinkaya, O., and Akan, O. B. (2018). Energy harvesting cognitive radio networking for iot-enabled smart grid. *Mobile Networks and Applications*, 23(4):956–966.

Park, H., Zhai, S., Lu, L., and Lin, F. X. (2019). Streambox-tz: secure stream analytics at the edge with trustzone. In *2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19)*, pages 537–554.

Patel, P., Intizar Ali, M., and Sheth, A. (2017). On Using the Intelligent Edge for IoT Analytics. *IEEE Intelligent Systems*, 32(5):64–69.

Paulraj, G. J. L., Francis, S. A. J., Peter, J. D., and Jebadurai, I. J. (2018). Resource-aware virtual machine migration in iot cloud. *Future Generation Computer Systems*, 85:173–183.

Pavlík, J., Benda, P., Šilerová, E., Jungwirth, M., et al. (2018). Scalability of gis applications with regards to iot. In *Agrarian Perspectives XXVII. Food Safety-Food Security, Proceedings of the 27th International Scientific Conference, 19-20 September 2018, Prague, Czech Republic*, pages 210–214. Czech University of Life Sciences Prague, Faculty of Economics and Management.

Phillips, L. J., DeRoche, C. B., Rantz, M., Alexander, G. L., Skubic, M., Despins, L., Abbott, C., Harris, B. H., Galambos, C., and Koopman, R. J. (2017). Using embedded sensors in independent living to predict gait changes and falls. *Western journal of nursing research*, 39(1):78–94.

Santoro, D., Zozin, D., Pizzolli, D., De Pellegrini, F., and Cretti, S. (2017). Foggy: a platform for workload orchestration in a fog computing environment. In *2017 IEEE*

International Conference on Cloud Computing Technology and Science (CloudCom), pages 231–234. IEEE.

Satyanarayanan, M. (2017a). Edge computing for situational awareness. In *2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pages 1–6. IEEE.

Satyanarayanan, M. (2017b). The emergence of edge computing. *Computer*, 50(1):30–39.

Seitz, A., Johanssen, J. O., Bruegge, B., Loftness, V., Hartkopf, V., and Sturm, M. (2017). A fog architecture for decentralized decision making in smart buildings. In *Proceedings of the 2nd International Workshop on Science of Smart City Operations and Platforms Engineering*, pages 34–39. ACM.

Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646.

Thangavel, D., Ma, X., Valera, A., Tan, H.-X., and Tan, C. K.-Y. (2014). Performance evaluation of mqtt and coap via a common middleware. In *2014 IEEE ninth international conference on intelligent sensors, sensor networks and information processing (ISSNIP)*, pages 1–6. IEEE.

Varghese, B., Wang, N., Barbhuiya, S., Kilpatrick, P., and Nikolopoulos, D. S. (2016). Challenges and opportunities in edge computing. In *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 20–26. IEEE.

Wang, J., Zheng, P., Lv, Y., Bao, J., and Zhang, J. (2019). Fog-ibdis: Industrial big data integration and sharing with fog computing for manufacturing systems. *Engineering*, 5(4):662–670.

Wingerath, W., Gessert, F., Friedrich, S., and Ritter, N. (2016). Real-time stream processing for big data. *it-Information Technology*, 58(4):186–194.

Xu, G., Li, M., Luo, L., Chen, C.-H., and Huang, G. Q. (2019). Cloud-based fleet management for prefabrication transportation. *Enterprise Information Systems*, 13(1):87–106.

Yang, L., Chen, Y., Li, X.-Y., Xiao, C., Li, M., and Liu, Y. (2014). Tagoram: Real-time tracking of mobile rfid tags to high precision using cots devices. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 237–248. ACM.

Yu, T., Wang, X., Jin, J., and McIsaac, K. (2018). Cloud-orchestrated physical topology discovery of large-scale iot systems using uavs. *IEEE Transactions on Industrial Informatics*, 14(5):2261–2270.

Yu, W., Liang, F., He, X., Hatcher, W. G., Lu, C., Lin, J., and Yang, X. (2017). A survey on the edge computing for the internet of things. *IEEE access*, 6:6900–6919.

Chapter 2

Analytics Everywhere: generating insights from the Internet of Things

This chapter has been published in the IEEE Access Journal. The full citation of this published article is:

Cao, H., Wachowicz, M., Renso, C., & Carlini, E. (2019). Analytics Everywhere: generating insights from the Internet of Things. *IEEE Access*, *7*, 71749-71769.

Abstract

The Internet of Things is expected to generate an unprecedented number of unbounded data streams that will produce a paradigm shift when it comes to data analytics. We are moving away from performing analytics in a public or private

cloud to performing analytics locally at the fog and edge resources. In this paper, we propose a network of tasks utilizing edge, fog and cloud computing that are designed to support an *"Analytics Everywhere"* framework. The aim is to integrate a variety of computational resources and analytical capabilities according to a data life-cycle. We demonstrate the proposed framework using an application in smart transit.

2.1 Introduction

Across the Internet of Things (IoT), transferring data from IoT devices to remote data centers is currently not efficient from a performance perspective due to the limitation on bandwidth and the high latency. In fact, the technological gap between the computational resources in the spectrum between an IoT device and the cloud is closing rapidly, especially with the advent of edge and fog devices that can support federated multi-tasking computation (Smith et al., 2017; Bonawitz et al., 2019) and virtualization (Morabito et al., 2018). In addition, an important requirement of IoT applications is related to privacy and confidentiality (Sicari et al., 2015). Keeping sensitive data closer to their sources may potentially reduce the risk of infringing privacy rights and breaking confidentiality.

Two phases can be distinguished in the evolution of IoT. The first phase has focused on the proliferation of sensors, protocols, and architectures where the main research challenges were related to network connectivity, IoT platforms, and sensor configurations. A second phase is gradually taking place where the core research challenges are shifting from physical infrastructures to analytical capabilities that are being developed according to the requirements of IoT applications (Marjani et al., 2017).

In this paper we introduce the concept of *"Analytics Everywhere"* as a con-

ceptual framework that facilitates building computational resources that are needed to support data analytics for IoT applications. We advocate that supporting the new generation of IoT applications is more than just moving computation from the cloud to the edge/fog nodes in a straightforward way. Instead, it requires an *"Analytics Everywhere"* framework in which computational resources are designed and work as a whole toward the completion of a network of analytical tasks. This embeds the concept of data streams moving around distributed computational resources (i.e. cloud, fog, and edge nodes) that provide storage and processing power for the execution of a network of tasks in such a way that a graph, sparse, and low-rank structure between the tasks is known *a priori*.

The research challenge is three-fold. First, there is a need to rethink how previous analytical algorithms have been independently developed. They must now be integrated in a network structure, in a way that makes explicit the dependency between the same tasks belonging to different algorithms as well as different tasks belonging to the same algorithms. This network structure will require a mathematical formulation such as Directed Acyclic Graphs (DAG), Petri-Nets, and WF-nets. Research work has been done in the past years on the mapping of DAG nodes onto computational resources, as for example in (Kliazovich et al., 2016; Anastasi et al., 2017). Second, a mapping between analytical capabilities and computational resources for running the analytical tasks must be defined, taking into account the variety of data life-cycles of IoT applications. In this case, analytical capabilities can be described as being descriptive, diagnostic, and predictive. However, it is still unknown what type of behaviour data streams exhibit during the data-life cycles of IoT applications. Finally, an overall orchestration of the computational resources (i.e. edge, fog, and cloud nodes) must be accomplished in order to guarantee a smooth execution of a variety of analytical tasks.

The contribution of this paper can be summarized as follows:

- We propose an *"Analytics Everywhere"* framework that integrates computational resources needed for a seamless execution of a network of analytical tasks having automated analytical capabilities, generating useful and high level information in a timely way.
- We demonstrate that a single computational resource (i.e. cloud) is not sufficient to support all analytical capabilities that are needed for IoT applications, considering computing power, data stream management, storage and networking capabilities.
- We discuss the challenges and how an *"Analytics Everywhere"* framework can be designed to perform descriptive, diagnostic, and predictive analytical tasks.
- We validate our *"Analytics Everywhere"* framework using a transit experiment by highlighting the pitfalls and discussing our experience.

The remainder of this paper is organized as follows. In Section 2.2, we reviewed different IoT enabling technologies and the data analytics that have been previously implemented using cloud/fog/edge computing. In Section 2.3, the *"Analytics Everywhere"* framework is presented, including the components of resource capability, analytical capability, and data life-cycle. Section 2.4 is dedicated to building an *"Analytics Everywhere"* architecture. Section 2.5 describes in detail the experiment of implementing our framework for a smart transit scenario and discusses the results. Section 2.6 concludes the paper and discusses further research.

2.2 Related Work

It is indisputable that IoT devices will produce a large amount of high-speed streamed and heterogeneous data that poses many challenges to performing management, processing, and analytical tasks within an acceptable time (Chen et al., 2014).

2.2.1 IoT Enabling Technologies

Al-Fuqaha et al. (2015) provide an overview of IoT enabling technologies that can offer automation, data aggregation, and protocol adaptation using different IoT devices. Overall, four main technologies can be identified in IoT: cloud, fog, edge, and communication technologies.

2.2.1.1 Cloud Computing

Cloud computing has dominated the infrastructure and processing architectures developed to support Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) models during the last decade, leading to a trend of Everything as a Service (XaaS) (Banerjee et al., 2011). By providing on-demand processing services with high availability and rapid elasticity through a selection of cloud architectures (e.g. Private, Public, Community, and Hybrid Cloud), previous research has pointed out that IoT devices can benefit from the virtually unlimited resources of the cloud, which compensates for their limitations in storage and computing capabilities (Botta et al., 2014; Díaz et al., 2016; Rao et al., 2012). As a result, most of the architectures used to monitor (Galache et al., 2014; Ren et al., 2015), optimize (Zhang et al., 2017), and analyze (Mukherjee et al., 2014) IoT data streams have been developed based on cloud computing.

However, cloud computing has shown limitations in supporting the short response time needed for processing the high data rates generated by IoT devices. Several open sources for processing IoT data streams such as Apache Storm (Karunaratne et al., 2017) or Apache Spark (Zaharia et al., 2016) have been proposed in the literature but they still present major drawbacks due to the geographic distribution, large-scale, and latency-sensitive characteristics of IoT applications (Patel et al., 2017; Sun and Ansari, 2016). It is worth noting that transporting the data streams to the cloud can still generate bottlenecks. While data storage density and computing power have increased 10^{18} and 10^{15} times respectively, the broadband capability has increased only 10^4 times over the last 20 years (Botta et al., 2014). Pushing the processing closer to IoT devices has emerged as an alternative solution, and edge and fog computing have been proposed as alternative IoT enabling technologies (Patel et al., 2017; Sun and Ansari, 2016; Satyanarayanan, 2017; Bonomi et al., 2014).

2.2.1.2 Edge Computing

According to Shi et al. (2016), edge computing refers to *"the enabling technologies allowing computation to be performed at the edge of the network, on downstream data on behalf of cloud services and upstream data on behalf of IoT services"*. The rationale behind edge computing is that 45% of IoT data will be processed and analyzed at the edge of the network in the future (Shi et al., 2016). Recently, Harth et al. (2018) have attempted to alleviate the network burden of transporting IoT data to the cloud by locally applying aggregation analytics at the edge, and sacrificing the analytical capability power due to the constraints of edge resources. A sliding window was applied to execute a simple linear classification to infer the context vectors (n-dimension row vector of contextual parameters such as temperature, sound, and humidity) within a specific tolerance threshold. Then, an aggregation analytics task

including distributive, algebraic, and holistic functions was triggered if the errors of the inferred context vectors were lower than the threshold. Otherwise, the smoothing algorithm reconstructed the context vectors before executing the aggregation analytics task.

2.2.1.3 Fog Computing

Fog computing was first introduced by Cisco as a bridge between the edge and cloud resources (Bonomi et al., 2012). Other technologies having a similar concept were also proposed in the literature such as cloudlet (Satyanarayanan et al., 2009) and mobile cloud computing (Khan et al., 2013) as well as mobile edge computing (Nokia and Intel, 2014). Lee et al. (2017) proposed an online computational caching framework to minimize the latency by storing and reusing intermediate computation results using fog nodes. Moreover, near realtime analytics was demonstrated in a seismic case study and realtime analytics was also achieved in an ambient noise imaging case study where a fog computing middle-ware architecture was developed for distributed cooperative analytics (Clemente et al., 2017). Other scenarios have been envisaged to apply fog computing, including Augmented Reality, Realtime Video Analytics, Mobile Big Data Analytics (Stojmenovic and Wen, 2014), Smart Grid, Smart Traffic Lights and Connected Vehicles (Bonomi et al., 2014), Decentralized Smart Building Control, Wireless Sensors and Actuators Networks (Yi et al., 2015). Unfortunately, none of these scenarios have been implemented so far.

2.2.1.4 Communication Technologies

Advances in communication technology play a vital role in bolstering the current growth of IoT. The proliferation of IoT devices is partially thanks to the advance-

ments in wireless communication technologies including Wireless Local Area Network (WLAN), Wireless Personal Area Network (WPAN), and Low-Power Wide Area Network (LPWAN) (Liao et al., 2018). While WLAN/WPAN provide a short range connectivity (about 1-100 metres) to support device-to-device (D2D) communication with a high data rate, LPWAN does not require much power, nor bandwidth to operate and provides long range connectivity (up to 50 kilometres) (Raza et al., 2017). Some typical communication technologies of WLAN/WPAN including Radio-frequency Identification (RFID) (Rose et al., 2015), Bluetooth Low Energy 4.0 (Huh and Seo, 2017), Zigbee (Wang et al., 2016), and Wi-Fi (IEEE 802.11) are applied in different IoT applications such as Smart Tourism (Cha et al., 2016), Smart Home (Aburukba et al., 2016), Connected Health (Rahmani et al., 2018). LPWAN technologies including unlicensed (e.g. SigFox, LoRa (Mekki et al., 2019)) and licensed (i.e. NB-IoT (Yang et al., 2017)) spectrum band are promising in terms of lowering power consumption, and cost, and increasing reliability and range (Sinha et al., 2017).

Cellular technologies that offer reliable broadband communication have had a certain role in shaping the IoT applications in the past, and they are expected to play an important role in the future. We have witnessed the growth of several generations of cellular networks from 2G and 2.5G which were designed to support voice services with an extension of small amount of data transmission, to 3G and 4G LTE that were capable of offering a wide coverage area, high security, and a dedicated spectrum allocation (Akpakwu et al., 2018). Although cellular technologies are not fit for all IoT applications, since they require very high operational cost and power consumption, they have shown to be suitable for specific scenarios such as connected cars or fleet management (van Dam et al., 2019). In particular, the next-generation, 5G, is expected to provide extreme mobile broadband (xMBB), massive machine-type communications (mMTC), and ultra-reliable machine-type communications (uMTC)

and is positioned to be the future communication technology for IoT applications that require ultra-low latency (Molina-Masegosa and Gozalvez, 2017; Li et al., 2018).

2.2.2 Data Analytics for IoT

Table 2.1 provides an overview of the type of analytical capability that has been implemented using cloud/fog/edge resources for different IoT applications. Most of the research efforts have been focused on descriptive analytics, and in particular, using edge computing resources to support near realtime/realtime analytics. The variety of IoT devices requires analyzing heterogeneous data *"on the fly"* and storing these data using various storage technologies. Very few studies found in the literature propose diagnostics and predictive analytics and were usually implemented in the cloud. To the best of our knowledge, our proposed *"Analytics Everywhere"* framework is the first research effort to combine different analytical capabilities in such a way that data streams can be transported and analyzed using the edge, fog, and cloud resources. These resources are inter-dependent and should be jointly developed to support IoT applications.

2.3 Analytics Everywhere Framework

This section describes our *"Analytics Everywhere"* framework to support the development of new data life-cycles and facilitate the building of effective resource and analytics capabilities for IoT applications. The three main components are as follows:

- Resource capability: This component consists of distributed compute nodes (i.e. cloud, fog, and edge nodes) that provide I/O, storage, computation and processing power for the execution of a network of analytical tasks;

Table 2.1: Overview of the analytical capabilities and their cloud/fog/edge resources for IoT applications.

Resource Capability	IoT devices	Analytical Capability	Applications	Ref.
Cloud	RFID Tags, BLE	Class Association Rule Mining using Sub-group Discovery	Anticipatory Ubiquitous Computing	(Atzmueller et al., 2016)
Cloud	WiFi, BLE	Clustering and Aggregating, Naïve Bayes	Location/Future Movement Prediction	(Nahrstedt et al., 2016)
Cloud	Spatial-Temporal Data, GPS, Camera, Environmental Sensors	Clustering (DBSCAN), Querying	Moving Object Map Analytics (MOMA), Contextual Spatial-Temporal Analytics	(Sun et al., 2016)
Cloud	GPS, Rain Gauge Data, Road Incident Report, Social Media	Descriptive (Statistical) and Predictive (Addictive Model, Kernel, SVM)	Urban Trajectory Data Analytics System	(Vieira et al., 2015)
Edge + Cloud	BLE	Descriptive (Statistical)	O/D Transportation Planning	(Herrera-Quintero et al., 2016)
Edge + Cloud	RFID Tags	Descriptive (Statistical)	RFID Ecosystem for management, IoT applications	(Welbourne et al., 2009)
Edge + Cloud	Sensors, Traffic Lights	Diagnostic (Virtual Representation and Data enrichment)	Virtual Object (VO) model to enrich context information with Cognitive Internet of Things	(Somov et al., 2013)
Edge	Phone Camera	Event Detection	Pedestrian Safety Detection (Offline Training/Online Detection)	(Wang et al., 2012)
Edge	Sensors, RFID	Descriptive (Statistical)	Proposed the Smart Object framework to encapsulate RFID, sensor, Internet-based data	(López et al., 2012)
Edge	Wearable Sensors, GPS Receivers, Laptop, Smartphone	Descriptive (survey, threshold analysis, self-report)	wearable system which can learn context-dependent personal preferences	(Krause et al., 2006)
Edge	Wifi Signal, GPS	Descriptive (Quantitative Analysis, Statistics)	Wireless monitoring system that can track pedestrian and passenger behaviors	(Qi et al., 2017)
Fog	R1+ Seismograph Nodes	Descriptive (Onboard cooperative processing)	A fog computing middleware for distributed cooperative data analytics for the seismic and ambient noise imaging case studies	(Clemente et al., 2017)
Fog	Augmented Reality, Virtual Reality data	Descriptive (online computational caching algorithm)	A computational caching frame-work in a fog network to minimize the transmission latency and computational latency by storing and reusing intermediate computation results	(Lee et al., 2017)
Fog	Sensors tagged on animals	Descriptive (Statistical)	Analyzing animal behaviors and monitoring animal's health	(Taneja et al., 2018)
Fog	Wearable Sensors	Diagnostics (K-mean Clustering)	Clustering on clinical speech data obtained from patients with Parkinson's disease	(Borthakur et al., 2017)

- Analytical capability: This component describes the best practice methods/algorithms for the execution of a network of analytical tasks that can meet the requirements of IoT applications;

- Data life-cycle: This component describes the changes that data streams go

through during the automated execution of a network of analytical tasks.

2.3.1 Resource Capability

An *"Analytics Everywhere"* framework is required to integrate resource capabilities taking into account one of the following aspects:

- Vicinity: This dimension describes how geographically close a compute node is to the source of data in order to execute a network of analytical tasks in that particular node. This dimension plays an important role in supporting IoT applications since compute nodes can be static (i.e. deployed inside a building) or mobile (e.g. deployed in a car), and their proximity to IoT devices, which are usually widespread geographically and mobile, will require integrated resource capabilities.

- Reachability: This dimension represents how easy it is to reach a compute node via a network. Typically, if a compute node is connected to the Internet with a fixed IP address, this can be considered a highly reachable resource, as opposed to a node connected using a private network and behind a Network Address Translation (NAT). In the case of IoT applications, the heterogeneity of IoT devices combined with the predominance of wireless access and short range networks will require an always-on reachability.

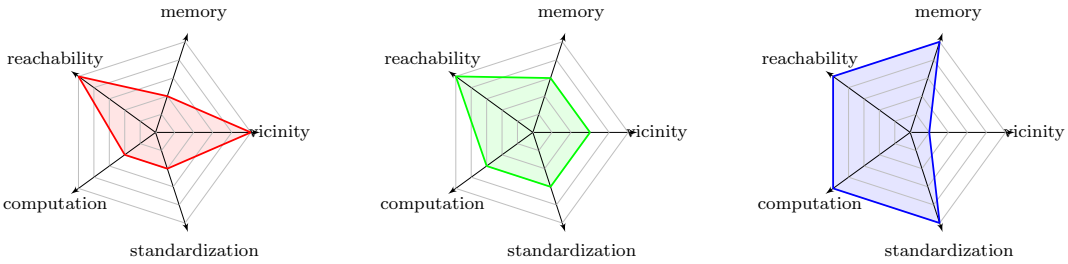
- In-memory and storage: This aspect describes how much data in a compute node should be kept in memory or be stored as a single ordinary disk file or in a database. The IoT data streams are expected to stay in-memory for a limited period of time as needed by an analytical task, and this decision will also depend on the data rate and data latency of the compute nodes. The

data rate varies from high rates of data collected at the edge to a low rate of aggregated and cleaned data arriving at the cloud. The latency is clearly very low at the edge due to the proximity to the IoT devices and increases as we move to the cloud.

- Computation: This dimension describes how much processing power is available at a compute node for performing a network of analytical tasks. A proper modeling taking into account the IoT application requirements can help in driving the decision about which computational resource to use in executing the analytical tasks.

- Standardization: This dimension represents the strongest challenge yet to be met in the implementation of *"Analytics Everywhere"* frameworks. The IoT standards range from network protocols and data-aggregation standards to security and privacy.

These dimensions play an important role in designing an *"Analytics Everywhere"* framework as shown in Figure 2.1. While computation and memory capabilities can increase as the analytical tasks are run from the edge to the cloud, reachability must be always available to an analytical task. Reachability is a critical dimension that requires analytical tasks to return well-timed and synchronized results, which demand a rapid increase in computational resources. Because fog nodes are intermediary gateways that seamlessly integrate edge and cloud resources, they can eliminate resource contention in the compute nodes and the communication links. In contrast, edge nodes can facilitate the necessary scaling of IoT applications because of their proximity to the IoT devices, making them an important computational resource for supporting near or realtime data analytics. However, the lack of adoption of standards in edge resources and IoT devices is currently hampering the implementation of *"Analytics Everywhere"* frameworks for IoT applications.



(a) Edge (b) Fog (c) Cloud

Figure 2.1: The main dimensions of resource capabilities.

2.3.2 Analytical Capability

In *"Analytics Everywhere"* frameworks, analytical capabilities can be described as being *descriptive, diagnostic,* and *predictive*. In general, descriptive analytics aims to summarize a given dataset, which can be either a representation of the entire population or a sample of it. While descriptive analytics can provide some key metrics and measures that might reveal *"What is happening in the real-world?"*, the diagnostic analytics aims to provide some insight to answer the question *"Why is it happening?"*. The findings of descriptive and diagnostic analytics can be utilized in predictive analytics to build prediction models for predicting tendencies, clusters and exceptions, and future trends. Based on the insights obtained from predictive analytics we can answer *"What will happen?"*.

Four major types of methods can be used to support descriptive analytics: *frequency measurement, central tendency measurement, dispersion or variation measurement,* and *position measurement*. Although descriptive analytics can be performed at the edge, fog, and cloud, we anticipate that it will be more often executed at the edge. This is due to its proximity to IoT devices, and also because (i) raw data are usually small in volume at the edge, and (ii) raw data can be subject to IoT application requirements that prevent data from being moved to a cloud due to privacy concerns.

Diagnostic analytics can be executed close to or far from an IoT device, depending on where it is more feasible to install relatively powerful computational resources. Diagnostic analytical tasks are usually supported by several algorithms such as DBSCAN (Ester et al., 1996) and Affinity Propagation Clustering (Frey and Dueck, 2007), which are executed to uncover hidden insights, patterns from contextualized data. Fog and cloud resources can be used to perform diagnostic analytics since they provide more powerful computation, storage, and accelerator resources than edge nodes. They can improve the accuracy and reduce the computational complexity of the diagnostic process by performing automated tasks in near realtime or periodically.

Predictive analytics requires on-demand processing services with high availability and rapid elasticity through the virtually unlimited resources of the cloud. New insights can be achieved by applying prediction algorithms such as Random Forest, Hidden Markov Model (HMM), and Neural Networks. Auto-scaling, scheduling, and monitoring services can also be used to handle the data streams received from the edge and fog nodes. The analytical tasks use a massive amount of historical IoT data that need to be processed according to the nature of IoT applications.

The overall network of tasks of our *"Analytics Everywhere"* framework is represented as a Petri-Net model in order to ensure the optimal conceptualization and execution of analytical tasks by avoiding path deviations, bottlenecks, and parallelism. For example, bottlenecks directly impact the speed at which the data streams flow, causing the tasks involved in the bottleneck to experience higher processing time than expected, and as a result, causing a delay in the execution of a network of analytical tasks. Petri-Nets can not only detect bottlenecks, but it can also help us unfolding their causes. In the case of path deviations, our Petri-Net model allows us to detect the data streams that have followed different paths to those expected

to occur within a network of analytical tasks. However, our Petri-Net model is not further discussed in this paper since it is out of the scope of this research work.

2.3.3 Data Life-cycle

In our *"Analytics Everywhere"* framework, the data life-cycle consists of five data abstractions that are used to describe the data input and output of an analytical task. They are raw, aggregated, contextualized, transformed, and extracted data. The actual data-life cycle process will depend on the sequence of the analytical tasks designed to support an IoT application. We expect that different IoT applications will require specific data life-cycle processes, but will have similar data abstractions.

Definition 1. *(Raw Data)*: The data streams $\mathbb{D}$ generated by IoT devices can be defined as a sequence of tuples $T_i \subseteq (T_1, \ldots, T_n)$ that contain a set of attributes such as:

$$T_i = (S_i, x_i, y_i, t_i)$$

where

S_i*: is a set of attributes (i.e. measurements) obtained from an IoT device;*

x_i, y_i*: is the geographical location of an IoT device;*

t_i*: is the timestamp t when a measurement has occurred.*

These tuples represent the raw data in a data life-cycle and their main characteristics have been previously outlined by (Hernandez et al., 2017) as one of the following:

- They are potentially unbounded in size and they are transported using data packages according to a priori known time window.

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

- There is no control over the order in which a tuple arrives within a data package or across data streams; and the probability distribution of the unknown data generation process may change over time due to its non-stationary state.

- It is not feasible to locally store a stream in its entirety since the local resources are normally limited. This means that data tuples are active and stay only for a limited time period in memory locally.

Definition 2. *(Aggregated Data)*: is defined as a set of new data tuples Q that are created by an aggregation operation Φ executed on a selected attribute (or a set of selected attributes) of a set of original data tuples T.

$$\begin{cases} \forall T_i \in (T_1, T_2, \ldots, T_n) : T_i = (S_i, x_i, y_i, t_i) \\ \mathbb{D} = (T_1, \ldots, T_n) \xrightarrow[\text{on attribute } S]{\Phi} \widehat{\mathbb{D}} = (Q_1, \ldots, Q_m) \\ \forall Q_j \in (Q_1, \ldots, Q_m) : Q_j = (Agg\_value_1, Agg\_value_2, \ldots) \end{cases}$$

Aggregation is a mathematical operation (e.g. sum, average, count, minimum) that takes multiple attributes of many tuples and returns a single value. However, some challenges still remain and they are associated with how to determine the granularity level that is needed by an analytical task and how the data output should be structured to avoid overly aggregating the data. For example, *"Analytics Everywhere"* frameworks depend on the time granularity being used at

a compute node, which can be a priori defined (e.g. every day, every month) or can be event-based where the time granularity is defined by when an event occurs. Moreover, the heterogeneity of IoT devices brings a variety of granularity relationships among compute nodes within an *"Analytics Everywhere"* framework. Bettini et al. (1998) described them as being groups into, finer than, shift equivalent, groups periodically into. The challenge is to design an Analytical Everywhere framework that can handle these relationships meanwhile the tuples are being aggregated at different compute nodes.

Definition 3. *(Contextualized Data)*: is defined as a set of new data tuples P that are created throughout the contextualization process using contextualization operation Ψ to add new attributes to the original data tuples T.

$$\begin{cases} \forall T_i \in (T_1, T_2, \dots, T_n) : T_i = (S_i, x_i, y_i, t_i) \\ \mathbb{D} = (T_1, \dots, T_n) \xrightarrow{\Psi} \overline{\mathbb{D}} = (P_1, \dots, P_n) \\ \forall P_i \in (P_1, \dots, P_n) : P_i = (S_i, x_i, y_i, t_i, Context_1, Context_2, \dots) \end{cases}$$

Contextualization is the most complex step in a data life-cycle that is performed to enrich the tuples using high level concepts accordingly to a particular IoT application. It is crucial in transforming meaningless tuples generated by IoT devices into semantically enriched data that are needed as an input to analytical tasks. New attributes are added to each tuple that can actually represent a context that characterizes a situation and the surroundings of IoT devices.

Definition 4. *(Transformed Data)*: is defined as a set of new data tuples K that are created by a transformation operation Υ executed on a selected attribute

(or a set of selected attributes) of a set of original data tuples T.

$$\begin{cases} \forall T_i \in (T_1, T_2, \dots, T_n) : T_i = (S_i, x_i, y_i, t_i) \\ \mathbb{D} = (T_1, \dots, T_n) \xrightarrow{\Upsilon} \ddot{\mathbb{D}} = (K_1, \dots, K_n) \\ \forall K_i \in (K_1, \dots, K_n) : K_i = (Trans\_value_1, Trans\_value_2, \dots) \end{cases}$$

Transformation refers to the replacement of an attribute by a function since there is a need to change the scale of an attribute or standardize the values of this attribute that belongs to a tuple. In *"Analytics Everywhere"* frameworks, transformation plays an important role in using categories or bins to incrementally create new attributes that can help to advance the analytical tasks.

Definition 5. *(Extracted Data)*: is defined as a subset of data tuples that are extracted from a set of original data tuples T using extraction (filtering) operation Ω; or a set of data tuples L that are created by an extraction (filtering) operation Ω executed on a selected attribute (or a set of selected attributes) of a set of original data tuples T.

$$\begin{cases} \forall T_i \in (T_1, T_2, \dots, T_n) : T_i = (S_i, x_i, y_i, t_i) \\ \mathbb{D} = (T_1, \dots, T_n) \xrightarrow[\text{on attributes } (S\|x\|y\|t)]{\Omega} \mathbb{D}'' = (L_1, \dots, L_n) \\ \forall L_i \in (L_1, \dots, L_n) : L_i = (att_1, att_2, \dots), \quad \forall att \subset (S, x, y, t) \end{cases}$$

2.3.4 Data Life-cycles in relation to Resource and Analytical Capabilities

Determining how to map different analytical capabilities with the most appropriate computing resources based on a data life-cycle of an IoT application is far from being a trivial endeavour since several aspects must be taken into account. Not all

analytical tasks can run on all compute nodes due to the complexity of learning paradigms that currently exist such as deep learning, on-line learning, local learning, and anticipatory learning, to mention a few. Moreover, it is important to point out that an *"Analytics Everywhere"* framework will have limitations and real-world IoT applications will play an important role in providing empirical evidence to validate and improve such a framework.

In Figure 2.2 we provide an overview of our proposed *"Analytics Everywhere"* framework, where each cell of the grid represents the expected data life-cycle according to analytical and resource capabilities. Overall, descriptive analytics at the edge will be more likely to handle raw data and aggregated data; while diagnostic and predictive analytics will be impracticable at the edge. By comparison, descriptive analytics in the fog will require data contextualization tasks that will support further extraction and transformation of data in the cloud.

On the one hand, fog resources are aimed at scaling up the processing power of edge nodes since larger data sets will be aggregated, contextualized, and transformed as needed for the descriptive, diagnostic, or predictive analytical tasks. On the other hand, the data life-cycles in the cloud are dependent on the type of data analytics that is required by an IoT application. Fog resources are not expected to replace the cloud. In fact, predictive analytics in the cloud will deal with contextualized, transformed and extracted data as well. We also can observe how data aggregation will play a significant role in diagnostic analytical tasks.

One example of these permutations includes IoT applications where analytical tasks are expected to be running at edge and fog resources since network and cloud connections are not available. For example, only 1 percent of data from an oil rig with 30,000 sensors is currently being analyzed for anomaly detection and control rather than optimization and prediction (Manyika, 2015). Other IoT applications in

smart buildings and smart mobility will typically require different permutations at all three resource levels (edge, fog, and cloud). The transit application we discuss later in this paper is a typical example of this case.

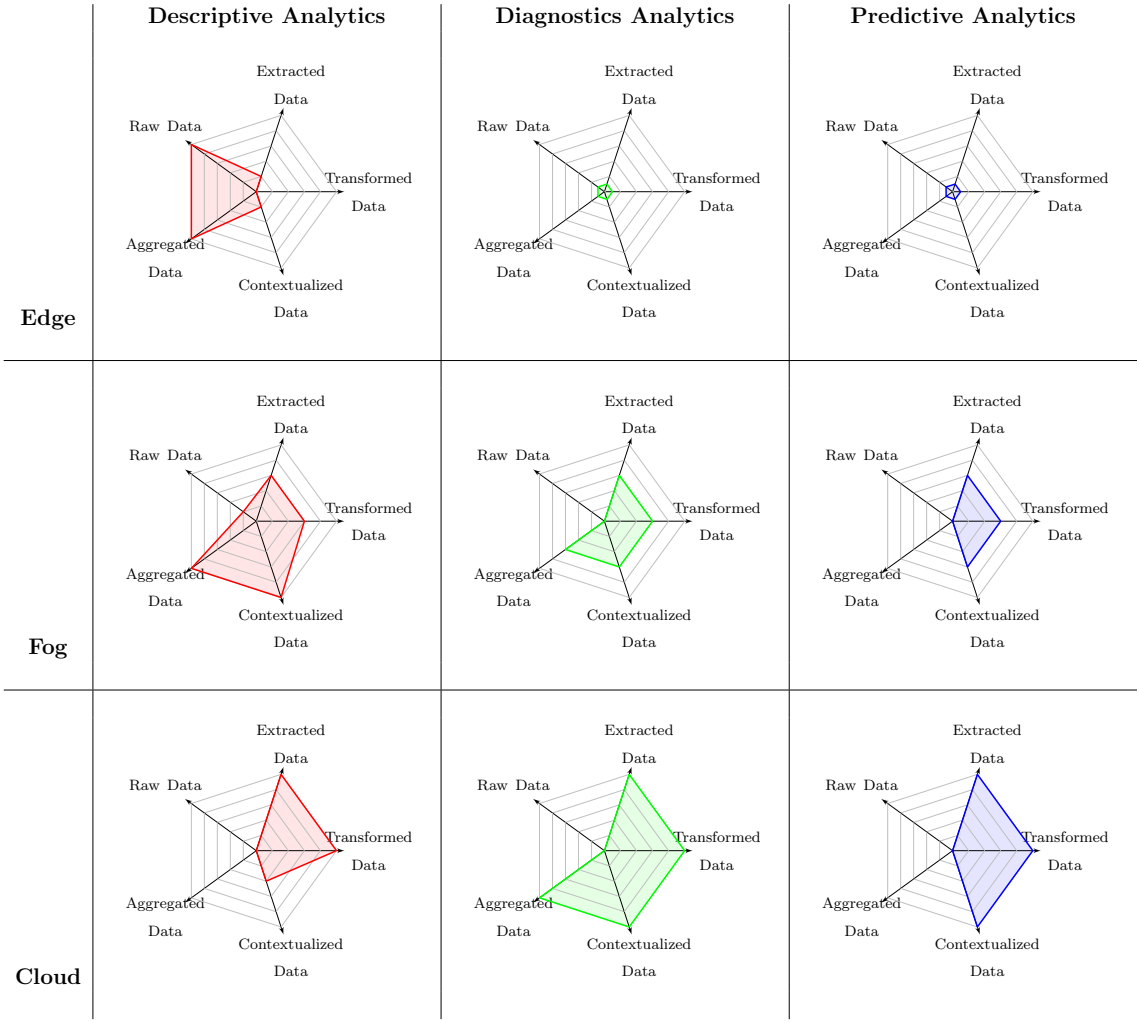


Figure 2.2: The matrix of data life cycle in relation to the analytical and resource capabilities.

2.4 Analytics Everywhere Architecture

We propose an architecture in which any analytical capability is mapped into and executed by a distributed resource architecture composed of a hierarchy of resources available at the edge, the fog, and the cloud. The proposed architecture is illustrated

in Figure 2.3. The aim is to support analytical tasks using a combination of different computation resources available at the edge nodes, the fog nodes and the cloud in order to provide meaningful information, actionable insights, and knowledge anytime and anywhere.

This section describes a general design guidance to implement an Analytical Everywhere framework. It consists of the following main components: networking, storage, computation/accelerators, controller/feedback, and data stream management/monitoring.

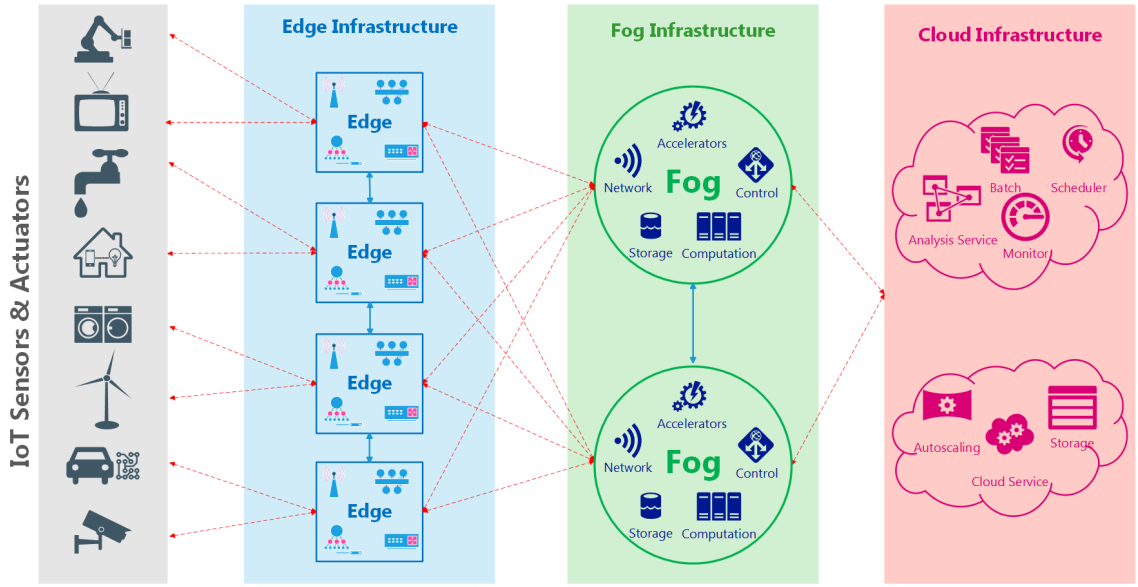


Figure 2.3: The proposed edge-fog-cloud architecture.

2.4.1 Networking

It is very important to choose the right networking technology for supporting a variety of IoT devices. Therefore, network standards, topology, and protocols should be considered carefully. Network developers need to consider various networking characteristics including throughput, fault tolerance, data rate, frequency band, power consumption per bit, number of nodes (hops) per network, and nominal range. In

order to balance the evaluations of these networking characteristics, a network topology is vital to outline the connections between the elements in the network (e.g. IoT devices, hub, gateways, edge nodes, fog nodes).

It is important to point out that due to the nature of our *"Analytics Everywhere"* framework, a comprehensive management of the entire network topology is required including wired and wireless, and seeking access and data transfer from the edge to core network elements. The networking connection between IoT devices and edge nodes can support many types of connections (e.g. Wi-Fi 802.11 a/b/g/n, LoRaWAN, Zigbee, 2G/3G/LTE Cellular) for rapid retrieval of tuples from the IoT devices themselves as well as a broadcasting service in which a forever loop of event time windows can be applied. One main requirement for implementing an *"Analytics Everywhere"* framework is to be able to guarantee that any unbounded size of raw generated tuples can be always transported independently from the type of an IoT device being used.

Once the management of the entire network topology is known, the appropriate communication protocols need to be selected. Figure 2.4 summarizes the most popular networking protocols and communication layers that are currently available. The protocol stack is described from low physical layers to high abstracted application layers.

The protocol selection will rely on the requirements related to what type of IoT devices are going to be used, how much realtime or near realtime versus batch processing is required, and what type of resource capabilities are available in the network. In other words, a one-protocol-fits-all approaches cannot be applied when implementing *"Analytics Everywhere"* frameworks.

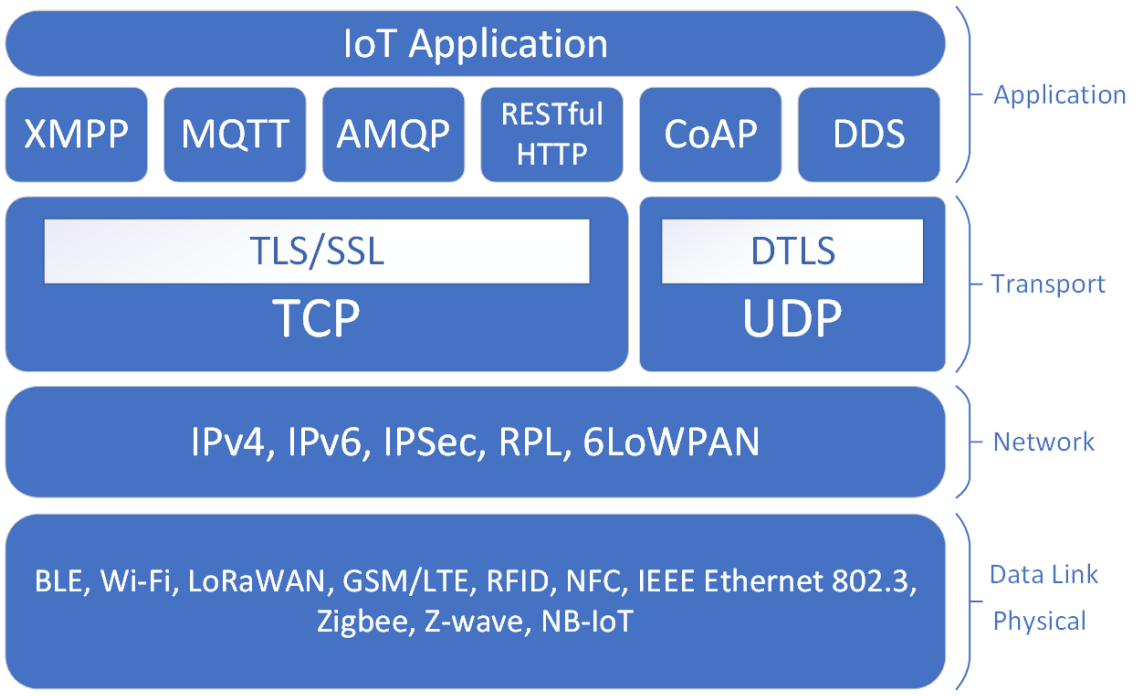


Figure 2.4: Current networking protocols supported by the *"Analytics Everywhere"* framework.

2.4.2 Storage

The second component of our system architecture that needs careful evaluation is the storage space. Indeed, the raw data tuples are constantly being generated by the IoT devices, transmitted over the network, and accumulated gradually over time. To find an optimal solution to storing the data is a non-trivial task when designing a system architecture for an *"Analytics Everywhere"* framework. The main design solutions are related to the following questions: *(1) which type of storage method should be applied? (2) where should the data be stored? (3) when is there a need to store data? (4) how can high availability be provided?*

The general guidelines are as follows:

- *In memory vs disk storage:* The mission-critical data tuples (hot data) that need to be accessed frequently by the analytical tasks should be stored in ways

offering fast retrieval and updates. Therefore, they should be kept in memory of the compute nodes, while less urgently accessed data (cold data) can be stored in a database, on disk, or in data files. Edge nodes in particular should be used to store in-memory data only.

- *Small vs medium vs large data:* Edge nodes are normally lightweight with low storage capabilities, while nodes at the fog have higher storage capability, and nodes in the cloud have the highest storage capability. Therefore, small, medium, and large data can be stored at the edge, fog, and cloud, respectively.

- *Nodes federation:* It is necessary to provide fault tolerance and high availability for data storage in our system architecture. All the compute nodes (at the edge, fog, and cloud) in the network can be used to aggregate and interconnect their storage environment as a unique place where data can be partitioned into many copy blocks and distributed everywhere in the IoT network.

2.4.3 Computation/Accelerators

The compute nodes are usually deployed covering a large geographical area and they can be static (i.e. a fog node deployed inside a building) or dynamic (i.e. an edge node deployed in a car). The core hardware of the compute nodes could be one or the combination of several processing units such as GPUs, CPUs, APUs, ASICs, FPGAs, and SoC accelerators. These computational devices can handle tasks either in independent style or in parallel, concurrent, or distributed styles. In this paper, three main types of shared resources based on the geo-distribution (at the edge, the fog, and the cloud) can be used to determine the type of compute nodes that are needed for the analytical tasks.

From the acceleration of data processing perspective, the processing power

of the compute nodes at the edge, fog, and cloud are sorted from low to medium to high. Therefore, nodes at the edge (static or dynamic) should be used to implement analytical algorithms for performing lightweight tasks such as descriptive analytics (in local scale) in order to generate new insights about the IoT device behaviors such as communication problems and low battery. Many IoT devices are expected to be connected to one or more edge nodes. However, high performance processing capabilities at the edge are prohibitive and may cause computational resource contention. Therefore, the accelerators at the fog can handle the heavier analytical tasks including descriptive (in regional scale) or diagnostic to reveal the patterns such as anomalies in the system. The highest computational capability in the cloud allows the nodes to handle the heaviest analytical tasks such as descriptive (in global scale), diagnostic (in long-term diagnosing), or predictive to forecast future changes in the system.

2.4.4 Controller/Feedback

The controller/feedback is an important component in this architecture. Once the analytical results of different analytical capabilities on the compute nodes at different places (edge, fog, cloud) are achieved, the actions of the IoT system need to be guided to optimize or adapt with the new change, new situation, new environment. Therefore, the feedback, which is a relevant result of the analytical capabilities, is pushed back from any compute nodes to order users or IoT actuators to take immediate actions. The controller/feedback can be real time, near real time or batch processing time depending on the place where it is computed. The criteria to choose the ramification (real time vs near real time vs batch processing time) of feedback is closely tied to the requirements of the application. For example, real time feedback detects anomalies in the operational behavior of the device at the edge, or

abnormal behavior in a traveling object's movement detected at the fog or the cloud.

2.4.5 Data Stream Management/Monitoring

In the *"Analytics Everywhere"* framework, there are two main options to select a data stream management engine: horizontal and vertical. The option chosen depends on the requirements of the application. Horizontal deployment means that the main components of a data stream management engine are horizontally deployed across remote nodes. Some examples include the open-source platforms such as Apache Flink, Apache Samza, Apache Apex, Apache Storm, Apache Spark Streaming[1]. In contrast, vertical deployment not only expands their services to the edge but also scales the data stream management components to the nodes close to the IoT devices. This latter deployment is a new trend so there are not many unique options available. However, some platforms can be considered such as Cisco Kinetic, IBM Watson IoT Platform Edge, Microsoft Azure IoT Edge, or Apache Edgent[2].

Streaming management can be either *stateful* or *stateless* depending on the analytical requirements of an IoT application. Stateless streaming management treats each event independently and creates the output only depending on the data tuples of that event. As an example, we can use a filtering operation to filter an incoming data stream of a transit network by a field (i.e.: busID) and write the filtered messages to their own stream. In contrast, stateful streaming management combines different events together and creates the output based on multiple data tuples taken from those events. A good example of this is counting the number of stops made at bus stations at which all buses in the transit network pull over during

[1]https://flink.apache.org, http://samza.apache.org/, https://apex.apache.org/, https://storm.apache.org/, https://spark.apache.org/streaming/

[2]https://www.cisco.com/c/en/us/solutions/internet-of-things/iotkinetic.html, https://console.bluemix.net/docs/services/IoT/edge/WIoTP_edge.html, https://azure.microsoft.com/en-ca/services/iot-edge/, https://edgent.apache.org/

a day. Moreover, developers can also specify a reliability mode or management semantics that guarantee it will provide for IoT data streaming across the entirety of the application architecture. It is worth noting that the guarantee is not only at the protocol level but it also can apply to the data stream management platforms. There are three main approaches as follows:

- At most once: At most once is a euphemism for there being no correctness guarantees that data tuples in a stream are guaranteed to be handled at most once by all streaming operators in the application. In other words, in the event of a failure, no additional attempts are made to re-handle these data tuples.
- At least once: At least once means that data tuples in a stream are guaranteed to be handled at least once by all operators in the application. If the failure happens, additional attempts will be made to re-handle these data tuples. This approach may cause unnecessary duplication of data tuples in the streams.
- Exactly once: Exactly once means that data tuples are guaranteed to be handled exactly the same as it would be in the failure-free scenario, even in the event of various failures.

2.5 Public Transit Scenario

2.5.1 Overview of the CODIAC Transpo Service

Public transport authorities must understand the performance of transit services to develop strategies for better transportation decision-making policies. Traditional solutions either failed to find the answers or have been too expensive to be widely deployed. Our *"Analytics Everywhere"* framework can provide automated analytical

capabilities that rely on the most appropriate computing resources. Moreover, the outcomes of our *"Analytics Everywhere"* framework can not only serve a transit authority, but it can also support a variety of user groups such as bus drivers and passengers who are seeking new insights to optimize their decisions and adjust their behaviors. For example, bus drivers might be interested in knowing how their driving performance has been for the last week while passengers would be interested in how frequently the services are delivered on-time.

In this section, we present the CODIAC Transpo as a public transit scenario to evaluate our proposed *"Analytics Everywhere"* framework. CODIAC Transpo serves the area of Greater Moncton, Canada[3]. Annually, CODIAC Transpo provides more than 2.3 million rides to transit users from Moncton, Dieppe and Riverview Area. The transit network currently operates 30 bus routes from Monday to Saturday, some of which have additional evening and Sunday services. Aiming to assist CODIAC Transpo in providing a safe, reliable, and professional transit service for passengers, we selected the following analytical capabilities:

- Descriptive Analytics: What is currently happening with the bus services in the CODIAC Transpo network?
- Diagnostic Analytics: Why have abnormal phenomena (e.g. congested, service interrupted, or normal events) happened to a bus service?
- Predictive Analytics: What will likely to happen to a bus service in the near future?

The CODIAC Transpo scenario can be described as each moving bus in the transit network generating realtime transit data feeds which are fetched by a mobile edge node installed directly in each bus. Here, descriptive analytical tasks are

[3]http://www.codiactranspo.ca/

running while the bus moves around a city. Once the analytical results are locally generated at the edge, they provide actionable information about what is happening to a moving bus. There are several transit hubs around the city where passengers and cargo are exchanged. At the transit hubs, the fog nodes are deployed to collect the cleaned data streams and the descriptive analytic results from different edge nodes whenever the buses gather there. At the fog resources, automated diagnostic analytic tasks are applied to understand why any abnormal phenomena have happened. Finally, a private cloud infrastructure is deployed in the transit headquarters aiming to summarize and handle the data streams from all the buses in the transit network. Figure 2.5 illustrates the scenario developed for the CODIAC Transpo network.

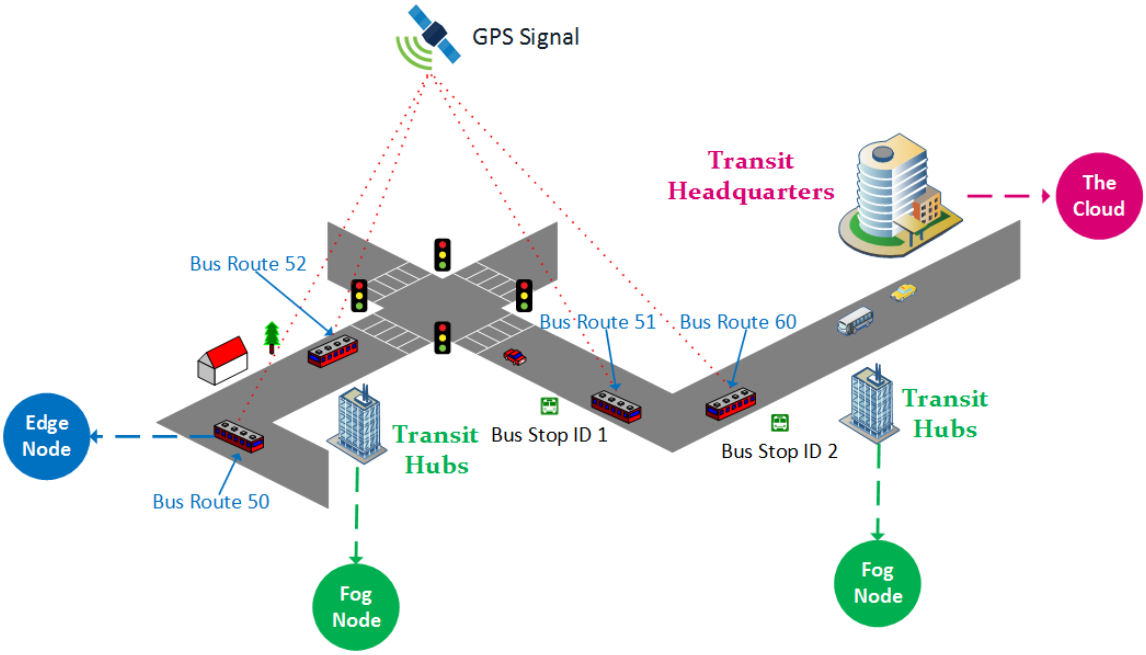


Figure 2.5: The CODIAC Transpo scenario.

2.5.1.1 The Transit Feeds

In this scenario, each bus is equipped with a mobile edge node that receives streaming transit feeds every 5 seconds containing the GPS position and telemetry data from sensors installed in the bus. These transit data feeds consist of a sequence T_1, ..., T_n

of out-of-order tuples containing attributes in the format:

$$T_i = (S_i, x_i, y_i, t_i)$$

where

S_i: is a set of attributes containing telemetry data such as the bus route identifier, the bus route number, the vehicle identifier, the trip identifier, the start time of a trip, and the end time of a trip. In this scenario we have a total of 17 attributes belonging to a tuple and they are listed in Table 2.2;

x_i, y_i, t_i: are the geographical coordinates x_i, y_i of the device at the sampling time t_i.

The bus route 51 was selected for evaluating our *"Analytics Everywhere"* framework because it has the highest trip density during a day. We have used 168,970 data tuples retrieved during a period of one week from 02/14/2017 to 02/20/2017. According to the transit schedule, there were 66 bus trips operating each day from Monday to Saturday and 23 bus trips on Sunday. As scheduled, each trip can take approximately 45 minutes.

2.5.1.2 Analytical Capabilities

The descriptive analytics are expected to reveal schedule adherence patterns which can be used by transit operators to adjust their operations such as route optimization, schedule modification, or bus maintenance. The diagnostic analytics also provide new insights that can assist bus drivers to change their driving behaviors to improve their scheduled adherence to the services. Finally, predictive analytics offer global insights

Table 2.2: The 17 attributes of the transit data feed.

ID	Attribute Name	Description
1.	vlr_id	The data point ID in the vehicle location report table.
2.	route_id_vlr	The route ID in the vehicle location report table.
3.	route_name	The route name.
4.	RouteID	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	End time of the trip.
11.	vehicle_id_vab	Vehicle ID.
12.	vehicle_id_vlr	Vehicle ID in the vehicle location report table.
13.	vehicle_id_vlr_ta	Descriptive name of the bus.
14.	bdescription	Bus description.
15.	lat	Latitude.
16.	lng	Longitude.
17.	timestamp	Timestamp of the data point.

on the whole transit network such as predicting trip behavior. Table 2.3 provides an overview of analytical capabilities and their corresponding techniques that have been implemented for the CODIAC Transpo scenario.

Table 2.3: Analytical capabilities of the CODIAC Transpo scenario.

Analytical Capability	Techniques	IoT application	Target Group of Users
Descriptive	- Statistics	- Schedule adherence	Transit Operators
Diagnostic	- Affinity Propagation Clustering	- Abnormalities detection	Bus Drivers
Predictive	- Random Forest	- Trip behaviors prediction	Passengers

2.5.1.3 Data Life-cycle

It consists of two cycles:

- Raw data arriving at an edge node, aggregated data are transported from the edge nodes to a fog node, and transformed data are transported from the fog

nodes to the cloud.

- Contextualized data are transported from the edge nodes directly to the cloud.

Figure 2.6 illustrates the data life-cycle implemented for the CODIAC Transpo scenario. The raw data tuples are generated every 5 seconds and the high volume of tuples, belonging to each sliding time window, is kept in-memory until it is transported to the fog node. The raw data tuples from the first time window are cleaned and pre-processed to remove errors, redundancies, and inconsistencies; the same tasks are performed for the next time windows in a sequential manner. The data tuples collected for the bus route trips were then contextualized at the mobile edge node to determine whether a bus is moving or stationary. These tuples have been further processed and analyzed at the edge using multiple descriptive statistical functions. From analytical results at the edge, the aggregated data were computed and passed through the fog for further diagnostic analytic tasks while the contextualized tuples were continuously sent to the cloud for prediction analytic tasks.

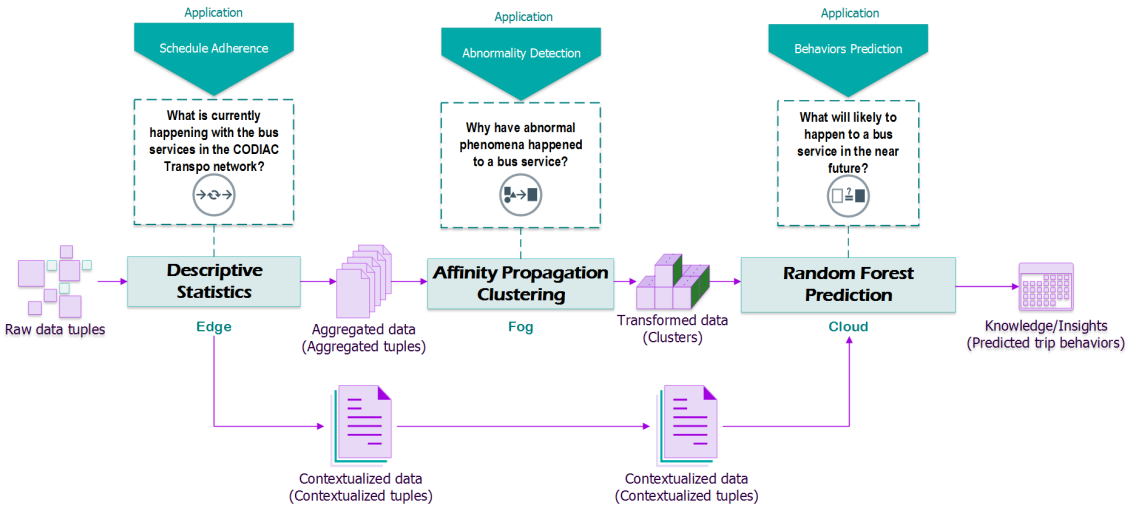


Figure 2.6: The knowledge/insights lifecycle from our public transit scenario.

Every 6 hours, all aggregated data were scheduled to arrive at the fog node. Here, we ran the affinity propagation clustering algorithm over the aggregated data

to transform them into clusters that can reveal abnormal trip behavior. Then, all transformed data (clusters) were also sent to the cloud for prediction analytic tasks.

The cloud receives the contextualized data tuples continuously being pushed from all the edge nodes as well as the transformed data resulting from the diagnostic analytical nodes. Both data sources (contextualized data tuples and transformed data) were used as input data of our random forest predicting model to predict the trip behaviors.

2.5.2 Analytics Everywhere Architecture

The system architecture is shown in Figure 2.7. For the data ingestion, an http POST, Wi-Fi and a 3G connection were used for rapid tuples retrieval from the IoT devices themselves as well as a broadcasting service in which a forever loop of event time windows can be applied. At the edge, the Cisco IR829 Industrial Integrated Services Router was used as a mobile edge node and was installed inside a bus. The router has an Intel Atom Processor C2308 (1M Cache, 1.25 GHz) Dual Core X86 64bit, 2GB DDR3 memory and Wi-Fi connection. This edge node handles all traffic routing, switching, and networking using an IOx operating system, running on a virtual machine that uses Linux Yocto (Cao et al., 2017). To collect the raw data tuples, Gateway Management Module (GMM) and Data Control Module (DCM), which are the integral parts of the Cisco Kinetic platform, were deployed on top of this mobile edge node. The Cisco Kinetic platform is a scalable, open system, and is adaptable for a variety of IoT applications. It can be used to extract, synchronize, compute, and move the data tuples to the right applications at the right time (Hernandez et al., 2017). A Message Broker was established at the edge to move the data from the edge to fog.

The fog node was implemented using the Cisco UCS 240 modular with a two rack-unit (2RU) server and 2 Intel Xeon processor E5-2600 CPUs, 24 double-data-rate 4 (DDR4) dual in-line memory (DIMMs) of up to 2400 MHz speeds, 6 PCI Express (PCIe) Generation 3 slots, and 12 large-form factor hard drives. It is managed by the Cisco Unified Computing System Manager Software. The fog node can host a virtual machine where an operating system can be run.

The cloud cluster is supported by Compute Canada which provides an IaaS where we have created and allocated cloud resources such as VMs, Servers, Storage, Load Balancers, IP addresses. Our cloud capabilities include a maximum of 5 Instances, 40 VCPUs, 150GB RAM, 2 Floating IPs, 5TB Volume Storage. In the cloud, we have the capability to handle the global geo-distribution of data (the whole transit network) and we have enough computing resources to perform complex analytical tasks. All necessary data needed for different analytical tasks are stored and are available in the cloud. The Hadoop ecosystem, in particular Apache HBase, Apache Zookeeper have been deployed in the cloud.

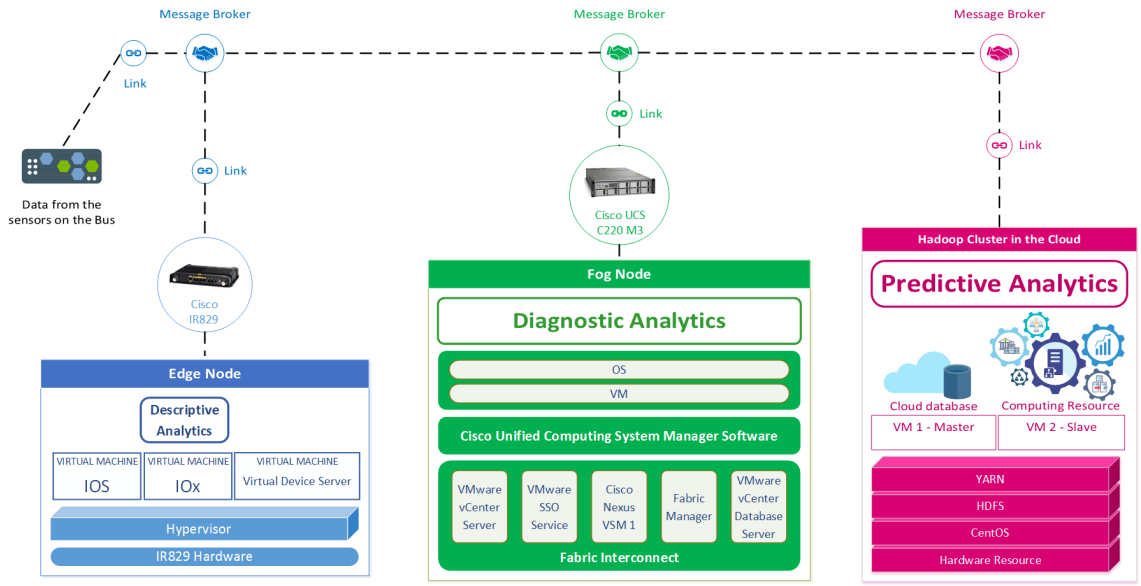


Figure 2.7: The *"Analytics Everywhere"* architecture implemented for our public transit scenario.

2.5.3 Descriptive Analytics

A contextualize function was implemented to interpret the status of a bus. The GPS coordinates were sent to the edge node every 5 seconds. A fixed distance value between two consecutive GPS positions of the bus was used for determining stops and moves. This value was empirically determined for the CODIAC Transpo network as being 15 meters. When the distance between the previous point and the current point is more than 15 meters, the bus is moving; therefore the current point is tagged as a move. In contrast, when the distance is less than 15 meters, the current point is tagged as a stop.

Additionally, a temporal aggregation function was used to compute (i) the actual time duration of a trip using the timestamps of the origin and destination points of each trip; (ii) the total number of stops during a trip; and (iii) the total number of moves during a trip. In summary, five data fields (Trip Id, Date, Start_Time, Move_Status, Stop_Status, Finish_Time) were used for the temporal computations. The following function was used to implement this step:

$$f(m,s,t) = \begin{cases} M = \sum_{i=1}^{n} m_i & \text{if } m_i \neq 0 \\ S = \sum_{i=1}^{n} s_i & \text{if } s_i \neq 0 \\ \Delta(t) = T_D - T_O \end{cases}$$

where

M, S: are the total number of moves and stops, respectively.

m_i, s_i: are the move and stop status in each tuple.

i = 1..n: is the index of the tuple in the data stream.

$\Delta(t)$: is the total time length of the trip.

T_D, T_O: are the timestamps of the destination and origin tuple.

Next, we computed the average trip time in the morning (5AM-12PM), afternoon (1PM-6PM), and evening (7PM-12AM). The average of the total number of moves and stops was computed for the different times of the day (i.e. morning, afternoon, evening) using the following function:

$$g(m, s, t) = \begin{cases} \overline{M} = \frac{\sum_{i=1}^{n} M_i}{n} \\ \overline{S} = \frac{\sum_{i=1}^{n} S_i}{n} \\ \overline{T} = \frac{\sum_{i=1}^{n} \Delta(t)_i}{n} \end{cases}$$

where

M_i, S_i, $\Delta(t)_i$: are the total moves, total stops, and total length of time for each trip.

n: is the number of trips during a period of time (morning, afternoon, evening).

2.5.4 Diagnostic Analytics

The goal was to demonstrate how it is possible to diagnose the causes of abnormalities, such as the interruption of services in near realtime. The affinity propagation clustering algorithm (Frey and Dueck, 2007) was selected to detect clusters. First, this algorithm automatically classified the clusters without prior knowledge about the number of clusters. Second, it can allow for non-metric dissimilarities. Therefore, we can handle non-metric space in our aggregated data. Also, the affinity propagation

clustering algorithm is deterministic over runs. The main idea behind this algorithm was to use a graph-based approach to let all data points collectively vote on their preferred 'exemplars', which are identified as those most representative of others. It is worth noting that implementing the affinity propagation clustering algorithm is a typical option of many options that we can choose for diagnostic analytics.

Algorithm 1 describes our implementation of the aggregated data pulled from the edge every 6 hours; its purpose is to discover any outliers that may indicate abnormal events (i.e.: traffic congestion). The input of this algorithm is a set of aggregated data points in which each data point contains 5 features *(TripID* $\langle Id_i \rangle$*, Start Time* $\langle St_i \rangle$*, Total_Move* $\langle \overline{M}_i \rangle$*, Total_Stop* $\langle \overline{S}_i \rangle$*, Total Trip Time* $\langle \overline{T}_i \rangle$*)* obtained from the edge computation after the end of each bus trip. The two most important features, Total_Move $\langle \overline{M}_i \rangle$ and Total_Stop $\langle \overline{S}_i \rangle$, are used as input for the clustering algorithm. At the end of this implementation process, the output will contain a set of original aggregated data points plus the cluster labels $\langle \widehat{C}_i \rangle$, which represent the aggregated information related to each trip, and a cluster that this set of data points belong to.

2.5.5 Predictive Analytics

We have used Random Forest (RF) to build a predictive model based on the performance benchmark carried out by Fernández-Delgado et al. (2014). Random Forest is an ensemble learning algorithm that can be used both for classification and regression problems by combining many small, weak decision trees in parallel to form a single, strong predictive model (Biau, 2012). Figure 2.8 depicts the predictive model showing a number of decision trees that were created during the training phase. Each decision tree contains a random subset of the most relevant features. When a new data tuple comes to the prediction model, it is predicted through each decision tree

Algorithm 1: Clustering algorithm using Affinity Propagation approach

Data: Set of $U = (U_1, U_2, U_3, ...)$ such that $U_i = (Id_i, St_i, \overline{M}_i, \overline{S}_i, \overline{T}_i)$ is the aggregated data point

Result: $Q = (Q_1, Q_2, ...)$ such that $Q_i = (Id_i, St_i, \overline{M}_i, \overline{S}_i, \overline{T}_i, \widehat{C}_i)$ in which $\widehat{C} = (\widehat{C}_1, ..., \widehat{C}_n), \widehat{C}_j = argmax[a(j,k) + r(j,k)]$

Initialize: The Similarity Matrix S $\forall j, k : s(j,k) = 0$; The Availability Matrix A $\forall j, k : a(j,k) = 0$; The Responsibility Matrix R $\forall j, k : r(j,k) = 0$;

Function `AP_Clustering(`U`):`

Compute Matrix S: $\forall j, k : s(j,k) \leftarrow -||V_j - V_k||^2$ where $V_j = (\overline{M}_j, \overline{S}_j)$ extracted from U_j; $V_k = (\overline{M}_k, \overline{S}_k)$ extracted from U_k;

repeat

Update Matrix R:

$$\forall j, k : r(j,k) \leftarrow s(j,k) - \max_{k' : k' \neq k} \{a(j,k') + s(j,k')\}$$

Update Matrix A:

$$\forall j, k : \begin{cases} a(j,k) \leftarrow \min\{0, r(k,k) \\ \qquad\qquad + \sum\limits_{j' : j' \notin \{j,k\}} \max\{0, r(j',k)\} \\ a(k,k) \leftarrow \sum\limits_{j' \neq k} \max(0, r(j',k)) \end{cases}$$

Cluster assignments:

$$\widehat{C} = (\widehat{C}_1, ..., \widehat{C}_n), \widehat{C}_j = argmax[a(j,k) + r(j,k)]$$

until *The Responsibility R and Availability Matrix A converge*;

$Q = U \rtimes \widehat{C}$

return Q;

and returns the target class label. A majority-voting function was utilized to vote the majority target class label and predict the label.

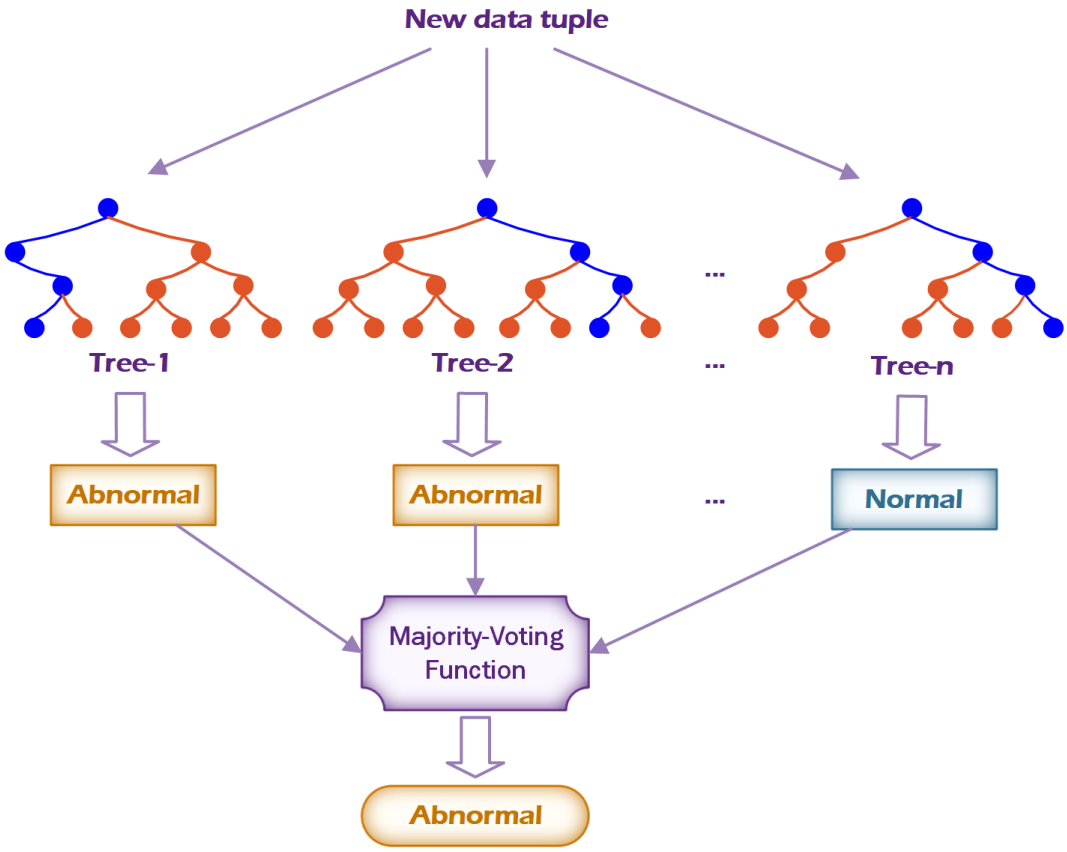


Figure 2.8: Random Forest model with majority voting.

Algorithm 2 provides details for the purpose of predicting trip behavior such as abnormal/normal events. The algorithm inputs are the clustering data pulled from the fog and the contextualized tuples received from the edge. The clustering data are a set Q of data points in which each data point contains 7 features *(TripID* $\langle Id_i \rangle$, *Start Time* $\langle St_i \rangle$, *Total_Move* $\langle \overline{M}_i \rangle$, *Total_Stop* $\langle \overline{S}_i \rangle$, *Total Trip Time* $\langle \overline{T}_i \rangle$, *Cluster Label* $\langle \widehat{C}_i \rangle$, *Behavior Label* $\langle Behavior_i \rangle$*)*. Meanwhile, the contextualized tuples belong to a set T' in which each tuple contains 17 features of the original tuple plus the new context feature.

The first step of Algorithm 2 is to merge the two datasets together to

Algorithm 2: Predicting algorithm using Random Forest

Data: Set of $T' = (T'_1, T'_2, ..)$ such that $T'_i = (S_i, x_i, y_i, t_i, context_i)$ is the contextualized tuples; Set of $Q = (Q_1, Q_2, ...)$ such that $Q_i = (Id_i, St_i, \overline{M}_i, \overline{S}_i, \overline{T}_i, \widehat{C}_i, Behavior_i)$ is clustering data

Result: Prediction model P

Function `Merge_Dataset(`T', Q`):`
  $G = T' \bowtie Q$ using TripID and Start Time $\langle Id_i, St_i \rangle$; `/* Left outer join 2 datasets */`
  $G = G.delete(\langle \overline{M}_i, \overline{S}_i, \overline{T}_i, \widehat{C}_i \rangle) = (G_1, G_2, ...)$ such that $G_i = S_i, x_i, y_i, t_i, context_i, Behavior_i)$;
  return G;
Function `Handle_Class_Imbalance(`G, *Method*`):`
  switch *the value of Method* **do**
    case *1* **do** Upsample the minority class;
    case *2* **do** Downsample the majority class;
    otherwise do Synthesize new minority class;
  end
  K-fold Cross-Validation (G) → Training set (G') and Testing set (G");
  return G', G'';
Initialize: Set number of small tree $Forest = int\_value$; Get number of features $F = Random\_number(2 : max\_no\_feature(G'))$;
Function `Build_Tree(`G', F`):`
  At each node:
    $f \leftarrow$ randomly select subset of Feature F;
    Split on best feature in f;
  return $Small\_Tree$;
Function `Random_Forest(`G', F`):`
  $P \leftarrow \emptyset$
  foreach $Tree_i \subseteq Forest$ **do**
    $\widehat{G'} \leftarrow$ A bootstrap sample from G'
    $p_i \leftarrow Build\_Tree(\widehat{G'}, F)$
    $P \leftarrow P \cup p_i$
  end
  return P;

form a unique dataset that can be used for the predictive model. For this purpose, the contextualized data tuples need to be indexed according to whether they have normal or abnormal behavior, based on the label provided by the clustering dataset. Therefore, we executed a left outer join operation on these datasets to form a new unique dataset. Then, we only keep the Behavior Label on this new dataset and eliminate the other features *(TripID* $\langle Id_i \rangle$*, Start Time* $\langle St_i \rangle$*, Total_Move* $\langle \overline{M}_i \rangle$*, Total_Stop* $\langle \overline{S}_i \rangle$*, Total Trip Time* $\langle \overline{T}_i \rangle$*, Cluster Label* $\langle \widehat{C}_i \rangle$*)* in order to avoid the impact on the predicted result since these other features are directly correlated to the Behavior Label.

Next, we handled another problem due to the data being outnumbered by normal behaviors with few instances of abnormal behaviors. This might cause bias towards the normal behaviors. Therefore, we used several solutions to balance the dataset; we used some methods such as upsampling the minority class (abnormal behaviors), downsampling the majority class (normal behaviors), or synthesizing a new minority class (abnormal behaviors) based on the existing samples. Then we applied cross validation procedure on the new dataset (training set G', testing set G'') to avoid overfitting or selection bias problems.

Once the class imbalance problem is handled, a predictive model is built based on the Random Forest approach: (i) A random number of decision trees are built in parallel. (ii) Each tree in the forest is built using a subset of features of the training set G' (the features are selected randomly among 17 features plus the context feature). (iii) Then, a bootstrap number of training samples from the training set G' are selected to form each tree in the forest. (iv) Finally, all the trees are combined together to form a single predictive model (see Algorithm 2).

2.5.6 Results and Discussion

2.5.6.1 Descriptive Analytical Results at the Edge

Figure 2.9 illustrates the existence of several missing trips that have been detected in realtime. The buses did not run on February 14th at 6 AM to 7 AM; and there were no trips at 10 PM on the 15th, 16th, 18th. Moreover, missing trips have also occurred on the 17th after 12 PM, on the 19th early in the morning (6 AM and 7 AM), and in the evening (6 PM to 10 PM). This is relevant information since it can generate warnings to the transit managers as well as passengers about the current state of the network at the trip level.

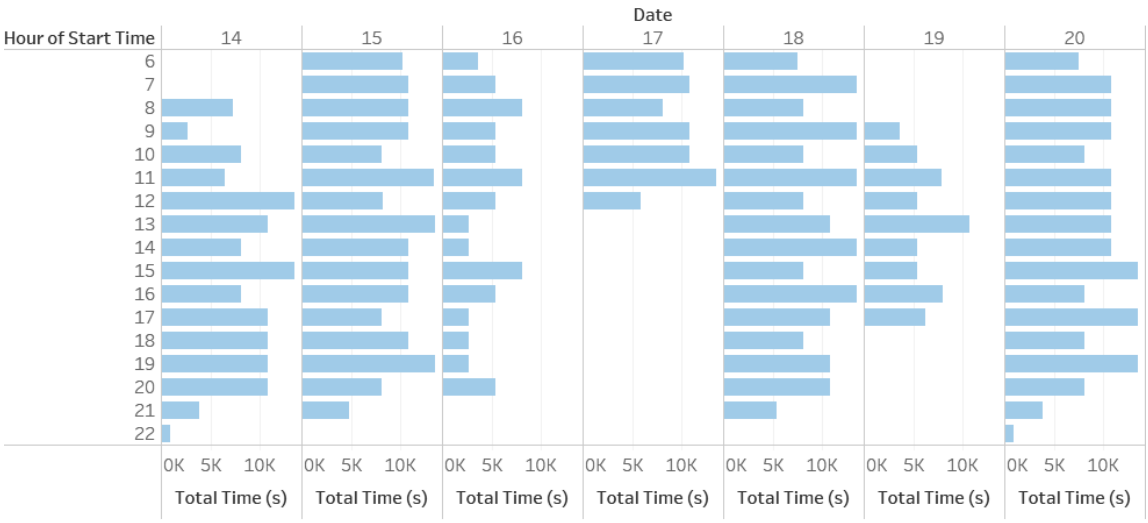


Figure 2.9: The distribution of the hourly trip times for each day of the week.

Moreover, computing the total trip time in realtime can provide relevant information to the transit manager about the abnormalities occurring with the bus service. For example, Figure 2.9 shows the total trip times from February 14th to February 20th. On February 14th, the shortest trip took 897 seconds (at 10 PM of the start time), meanwhile the longest trip took 13,468 seconds (at 12 PM of the start time). The weather conditions were fair on that day, making such an information

relevant as a feedback to the transit manager in order to identify the actual cause of these disruptions on the bus service. In contrast, on February 16th the bus service was erratic due to a snowstorm as shown by the different values of the total trips. This information is relevant as a feedback to be provided to the passengers in such a way that they would be able to make a decision to take a bus or to search for another mode of transportation.

To assess the mobility patterns of bus route 51 during the week, we selected 2 trips in the morning, 2 trips in the afternoon, and 2 trips in the evening, with each pair of trips starting at the same time in order to plot the total number of moves and total number of stops and compare the trips (see Figure 2.10). By comparing these two aggregation numbers of each trip during an operating date, we can find which trip is congested/unblocked based on pace behavior by reasonably assuming that the higher number of Stops will cause a congested trip. Figure 2.10 indicates that bus route 51 is a busy route based on the fact that the average number of Stops (273) in a trip is higher than Moves (189).

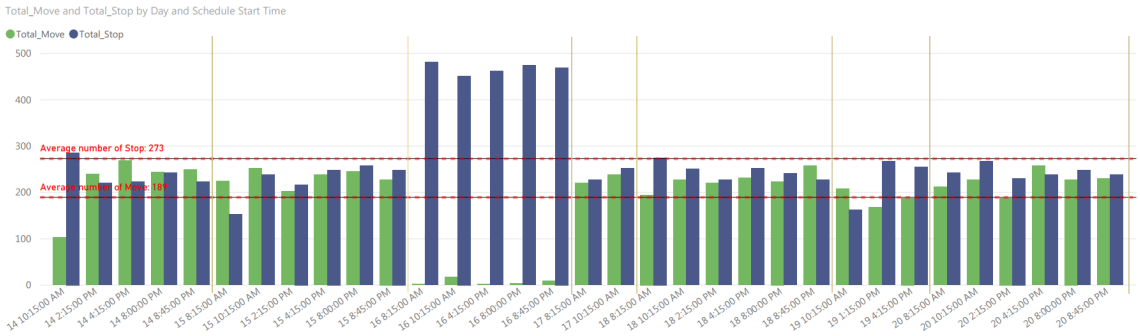


Figure 2.10: The comparison between the total number of Stops and Moves at different times during a week of observation.

2.5.6.2 Diagnostics Analytical Results at the Fog

Figure 2.11 illustrates the results obtained from running the clustering algorithm on the aggregated data. As we can notice in this figure, there are a total of 24 clusters

found from 419 trips accumulated from a week of data in this experiment. Most of them - which are located in the blue diamond box (see Figure 2.11) - adhered to the schedule, having ordinary pace behaviors. Therefore, they were labelled as the normal trips based on the identification of the transit managers. However, there were also some trips containing anomalous behaviors. For example, when the total number of Moves is outnumbered by the total number of Stops, this means that the total trip time is much shorter than usual. Hence, these trips were identified as the abnormal trips (shown as red circle of clusters in Figure 2.11).

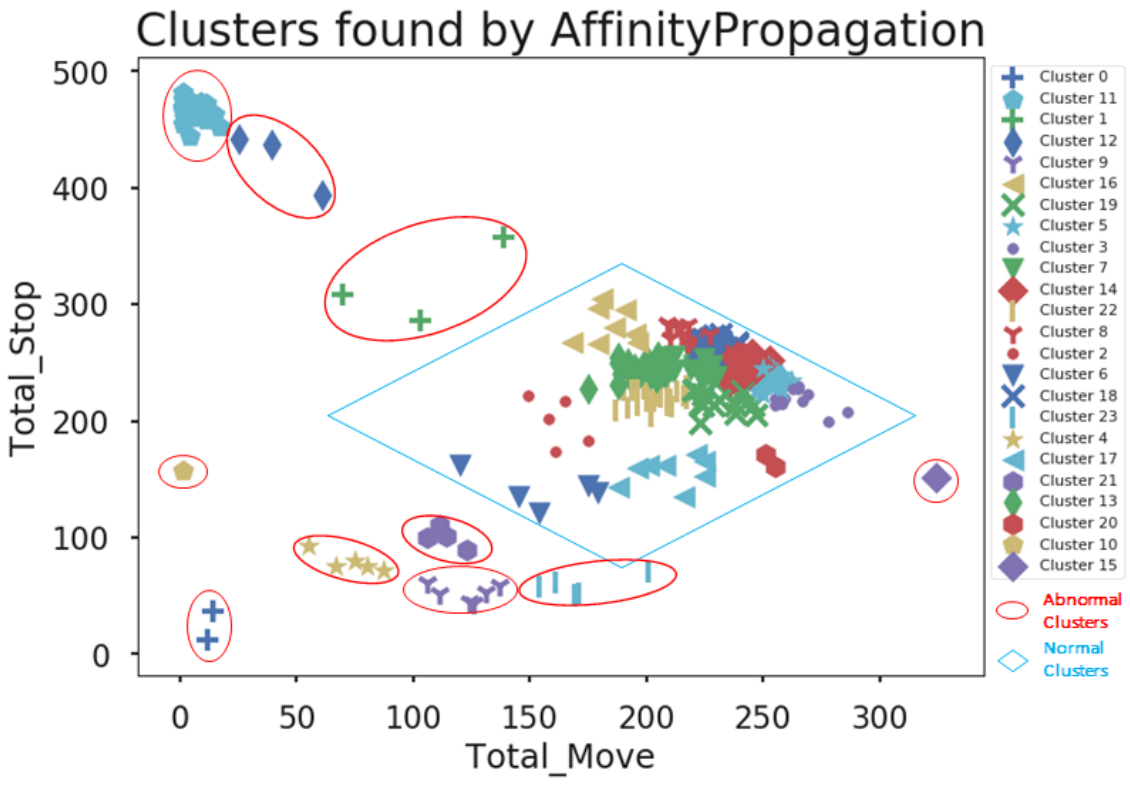


Figure 2.11: Overview of the clusters that were computed at the fog node.

After the clustering algorithm produced its results, a new data feature representing the behavior label (normal/abnormal) was added to the clustering dataset. Therefore we have now a dataset with 7 features *(TripID* $\langle Id_i \rangle$*, Start Time* $\langle St_i \rangle$*, Total_Move* $\langle \overline{M}_i \rangle$*, Total_Stop* $\langle \overline{S}_i \rangle$*, Total Trip Time* $\langle \overline{T}_i \rangle$*, Cluster Label* $\langle \widehat{C}_i \rangle$*, Behavior Label* $\langle Behavior_i \rangle$*)*). This clustering dataset was finally transmitted to our

cloud environment for further predictive analytics.

2.5.6.3 Predictive Analytical Results in the Cloud

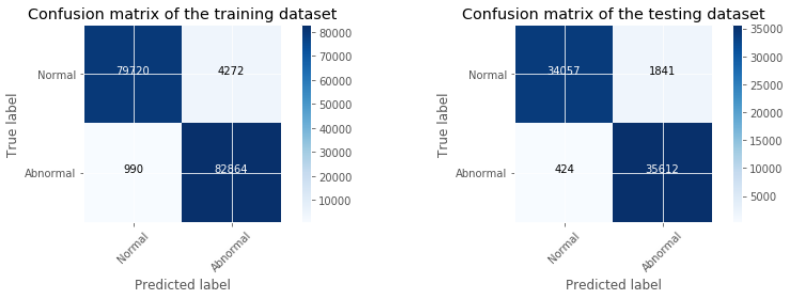
We evaluated our predictive model using 10-fold cross validation. There were a total of 239,780 tuples used to build this model, of which 2/3 are used for the training while the 1/3 remaining tuples are used for the testing. We then computed the average accuracy of the model. Table 2.4 shows the several main evaluation metrics such as accuracy, precision, recall, F1 score, and Area under the ROC Curve (AUC) on both training and testing datasets. In comparison, the accuracy of both sets is very similar, accounting for 96.86% (training set) and 96.85% (testing set). Similarly, the precision score of the training set is not very different from the one of the testing set (95.10% vs 95.08%). Also, while the recall and F1 score are the same, the AUC differed by only 0.02% on both sets.

Table 2.4: The evaluation of our prediction model.

	Accuracy	Precision	Recall	F1 Score	AUC	Support
Training Set	0.9686	0.9510	0.9882	0.9692	0.9687	167846
Testing Set	0.9685	0.9508	0.9882	0.9692	0.9685	71934

Figure 2.12 illustrates the confusion matrices on both sets. As can be seen, the type I and type II errors on both sets are very low, while the predicted condition positive and predicted condition negative values remain very high.

We also studied to find the importance of each feature that affects the predictive results of this model. Therefore, we visualized the importance score of each feature in the training set. Figure 2.13 indicates some important points to improve our model. First, the latitude, longitude, and the timestamp of a tuple are the 3 most important features that highly influence the predictive results in our model.



(a) (b)

Figure 2.12: Confusion matrices.

Second, the first 4 features *(RouteID, route_id_vlr, route_name, route_nickname)* in Figure 2.13 are almost unimportant to our predictive model. Therefore, they can be removed during the training phase to improve our predictive results since keeping them can introduce some noise in our model.

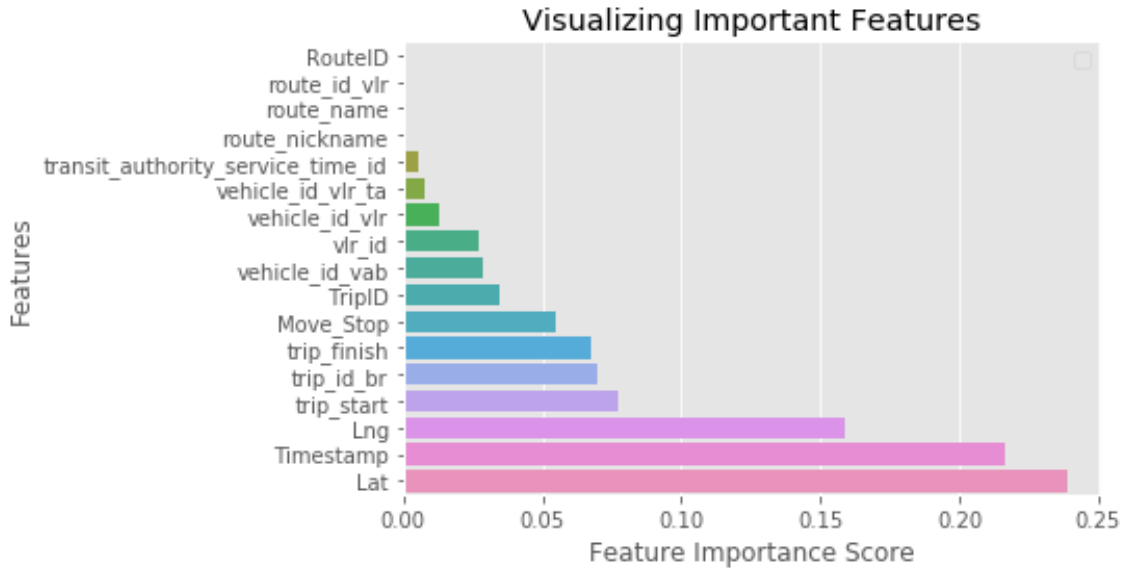


Figure 2.13: List of the most influential attributes in the prediction model.

To evaluate how the accuracy of the prediction model changes as a function of the training set size, we have plotted the accuracy curve as shown in Figure 2.14 . This plot indicates that, not surprisingly, when training data samples increase, the accuracy of our predictive model increases.

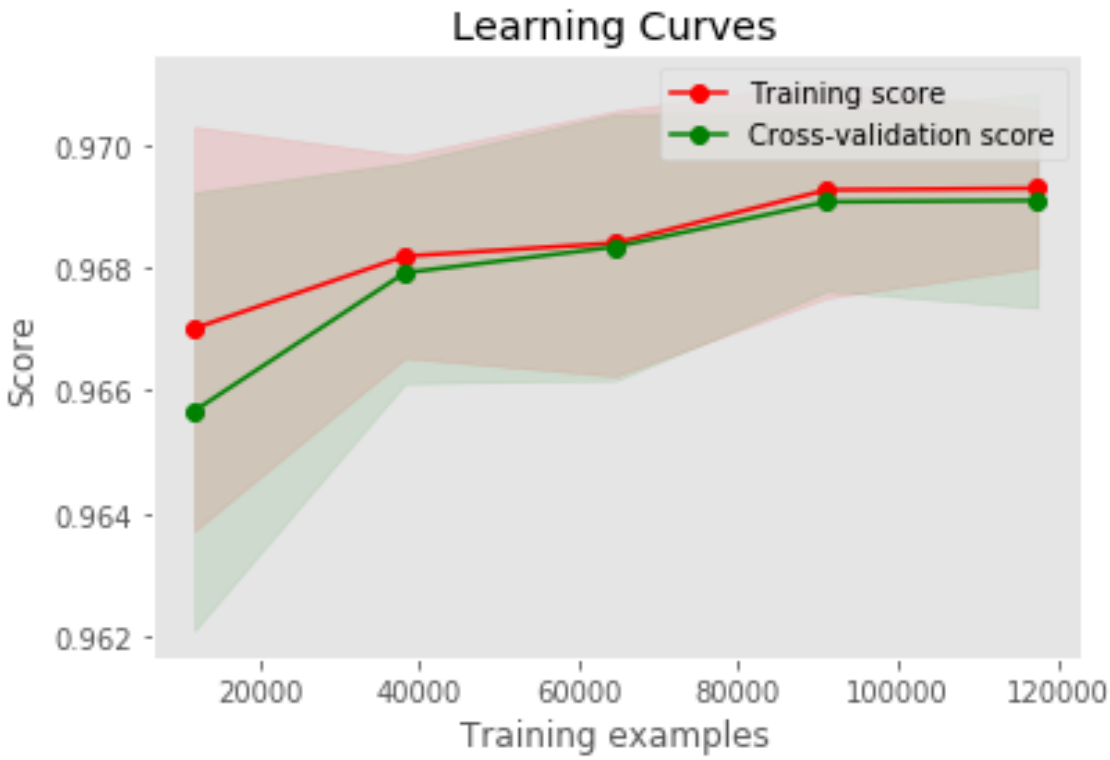


Figure 2.14: Accuracy of the prediction based on number of training items.

Moreover, Figure 2.15 shows the area under the ROC curve to measure the quality of our predictive model. As can be seen, our predictive model has a very high AUC score (0.97) indicating that it performs well as a general measure of predictive accuracy.

At the end of the computation in the cloud, the predicted values become the historical feedback for the transit managers, bus drivers, and passengers in order to understand how efficient the bus service is at the transit network level during a long period of time. In this experiment we have only used the data generated by one bus route as an example; however, the predictive model can be applied to the whole transit network. It is also worth noting that our model can continuously retrain and update itself with the new datasets that are consecutively sent to the cloud and will be used to offer better predictive results.

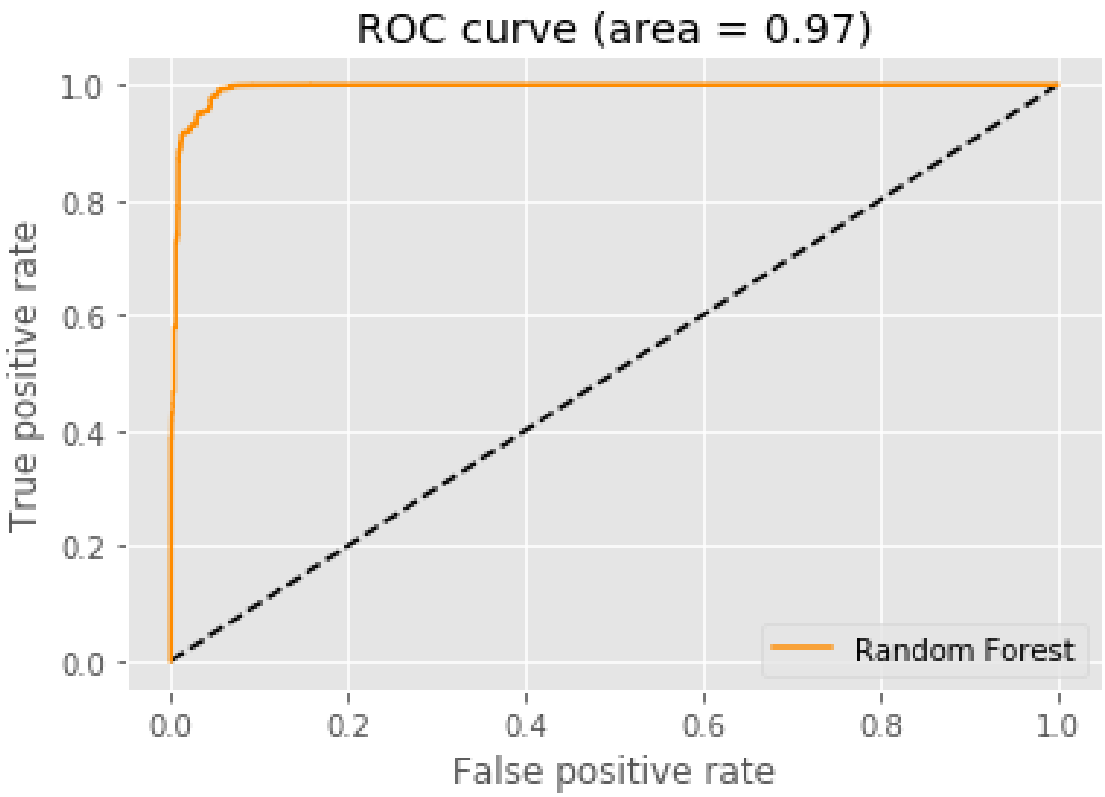


Figure 2.15: Area under the ROC Curve of our predictive model.

2.5.6.4 Discussion

We can evaluate the performance of this proposal using the Service Delivery Time (SDT) metric. SDT is computed as

$$SDT = T_I + \sum_{i=1}^{n} T_{P_i} + \sum_{i=1}^{n} T_{A_i} + T_F$$

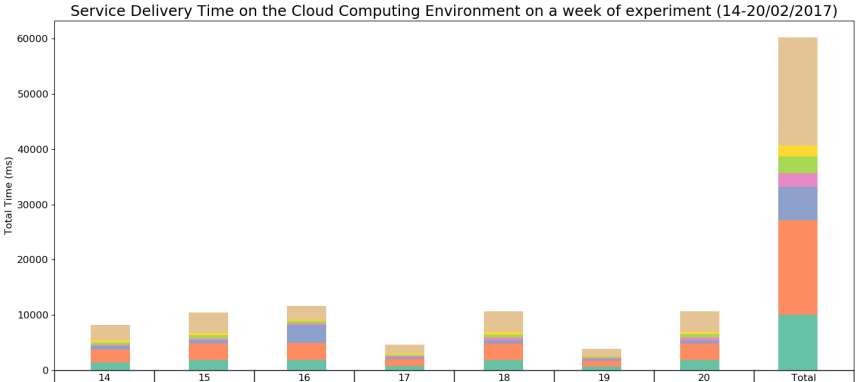
where

- T_I: Total time the data streams are ingested in the system
- T_{P_i}: The processing time of the task i^{th} in the system
- T_{A_i}: The analytical time of the task i^{th} in the system

- T_F: The feedback time that the system emits the actionable insights to the users or devices.

Figure 2.16 illustrates the detailed performance during a week of experiments of 7 tasks to delivery the service in the cloud. They include the ingestion time I, processing time P *(*$P1$*: Eliminating Redundant Data,* $P2$*: Removing Duplicated Data,* $P3$*: Normalizing Missing Value)*, analytical time A *(*$A1$*: Extracting Value,* $A2$*: Sorting,* $A3$*: Computing Stop/Move)*. At the current stage, we have not reached the level of fully computing the feedback time yet, but we could assume that the feedback time will take $\delta(t)$ (ms). Therefore, the service delivery time on our cloud computing environment can be computed by $SDT = T_I + \sum_{i=1}^{3} T_{P_i} + \sum_{i=1}^{3} T_{A_i} + \delta(t)$.

From our experience, it is not worth gathering all the data streams to the cloud then processing and analyzing them in batch since (1) A massive number of data tuples contain errors and inconsist information; almost half of the tuples used in our implementation (Cao and Wachowicz, 2019) were deleted. In fact, processing time in Figure 2.16 accounts for about 40% of service delivery time in the cloud. (2) With such a large amount of unnecessary data arriving in our system, there is a burden on our system in terms of energy consumption, bandwidth contention, and maintenance cost. Therefore, our new *"Analytics Everywhere"* framework is a fresh step forward to tackle these issues. Although further empirical experiments at the edge and the fog need to be done in the near future, it is expected that the data ingestion time T_I will be less than shown in Figure 2.16 because we will move some processing and analytical tasks close to the data source. Also, the data processing time is expected to be reduce as well as the new feedback time $\delta(t') < \delta(t)$ since the data processing and analytical tasks happen close to the data source instead of being sent to the cloud.



	14	15	16	17	18	19	20	Total
A3: Computing Stop/Move	2842.9	3752.2	2512.0	1609.7	3811.8	1304.3	3797.6	19630.4
A2: Sorting	280.8	370.6	248.1	159.0	376.5	128.8	375.1	1939.0
A1: Extracting Value	431.7	569.8	381.5	244.4	578.9	198.1	576.7	2981.0
P3: Normalizing Missing Value	278.5	347.1	363.0	232.9	551.5	188.7	549.4	2511.0
P2: Remove Duplicated Data	618.9	566.9	3251.0	268.8	510.6	317.0	546.8	6080.0
P1: Eliminate Redundant Data	2380.9	3058.3	3112.0	1316.2	3072.0	1104.5	3076.0	17120.0
I: Ingesting Data Time	1393.6	1790.2	1821.6	770.5	1798.2	646.5	1800.5	10021.0

Figure 2.16: Performance results based on service delivery time.

2.6 Conclusions and Future Work

This paper presents an *"Analytics Everywhere"* framework in the context of a composite architectural paradigm that includes edge, fog, and cloud resources for analyzing data streams generated from the Internet of Things. The framework aims to facilitate the design of IoT applications, bringing together in the same conceptual framework the computational capabilities of resources and analytical tasks, taking into account the characteristics of data life-cycles. The framework is based on the idea that IoT applications are convenient to push the computation toward the edge while trying to keep most of the data as close as possible to where it originated. This presents immediate advantages that would be favourable for today's IoT applications. It can support data privacy to a certain extent, reduce the cost to transfer large amounts of data to data centers, and make it possible to transmit feedback quickly to a variety of users. In contrast, it creates data management issues ranging from data governance, data heterogeneity, to data integrity.

We have applied the proposed framework on an actual real-world scenario

for the management of a public transit. Our lesson learned is that if any of the edge/-fog/cloud resources of the system architecture are considered in isolation , they would not be able to manage the IoT application, without compromising on functionalities or performance. Still, using a combination of edge, fog, and cloud resources requires careful coordination and a precise allocation of analytical capabilities. That is why the a-priori mapping between analytical capabilities with the appropriate computation resources should be set up by a developer; we do not expect that a user will take this role. Failing to achieve this mapping will have a negative impact on the performance and accuracy of the analytics performed. More research work is needed to determine this impact on over fitting our analytical models.

Despite the fact that PaaS/IaaS models are still an open issue in edge/-fog/cloud computing environments in an IoT ecosystem, our prototype has outlined the interchanging major components as being resource capability.

For future research work, we plan to extend the framework by considering security, latency, fault tolerance, and privacy requirements of IoT applications. Regarding the IoT application, we plan to increase the requirements in the cloud resources by adding a data visualization component, such as Kibana or Grafana. Our current prototype is not capable of accommodating dynamic task sharing, but this is definitely our next step. It is important to point out that our Analytical Everywhere framework does not need to be modified to support dynamic task sharing since it relies on the assumption that tasks should be *a priori* allocated, exploiting the different resources, regardless of workload balancing. Finally, more research is needed to understand the balance between supervised versus unsupervised learning for future reinforcement and federated learning.

References

Aburukba, R., Al-Ali, A., Kandil, N., and AbuDamis, D. (2016). Configurable zigbee-based control system for people with multiple disabilities in smart homes. In *2016 International Conference on Industrial Informatics and Computer Systems (CIICS)*, pages 1–5. IEEE.

Akpakwu, G. A., Silva, B. J., Hancke, G. P., and Abu-Mahfouz, A. M. (2018). A survey on 5g networks for the internet of things: Communication technologies and challenges. *IEEE Access*, 6:3619–3647.

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

Anastasi, G. F., Carlini, E., Coppola, M., and Dazzi, P. (2017). Qos-aware genetic cloud brokering. *Future Generation Computer Systems*, 75:1–13.

Atzmueller, M., Fries, B., and 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*, pages 1239–1246. ACM.

Banerjee, P., Friedrich, R., Bash, C., Goldsack, P., Huberman, B., Manley, J., Patel, C., Ranganathan, P., and Veitch, A. (2011). Everything as a service: Powering the new information economy. *Computer*, 44(3):36–43.

Bettini, C., Dyreson, C. E., Evans, W. S., Snodgrass, R. T., and Wang, X. S. (1998). A glossary of time granularity concepts. In *Temporal databases: Research and practice*, pages 406–413. Springer.

Biau, G. (2012). Analysis of a random forests model. *Journal of Machine Learning Research*, 13(Apr):1063–1095.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H. B., et al. (2019). Towards federated learning at scale: System design. In *Proceedings of the 2nd SysML Conference*.

Bonomi, F., Milito, R., Natarajan, P., and Zhu, J. (2014). Fog computing: A platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*, pages 169–186. Springer.

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM.

Borthakur, D., Dubey, H., Constant, N., Mahler, L., and Mankodiya, K. (2017). Smart fog: Fog computing framework for unsupervised clustering analytics in wearable internet of things. In *Signal and Information Processing (GlobalSIP), 2017 IEEE Global Conference on*, pages 472–476. IEEE.

Botta, A., De Donato, W., Persico, V., and Pescapé, A. (2014). On the integration of cloud computing and internet of things. In *Future internet of things and cloud (FiCloud), 2014 international conference on*, pages 23–30. IEEE.

Cao, H. and Wachowicz, M. (2019). The design of an iot-gis platform for performing automated analytical tasks. *Computers, Environment and Urban Systems*, 74:23–40.

Cao, H., Wachowicz, M., and Cha, S. (2017). Developing an edge computing platform for real-time descriptive analytics. In *Big Data (Big Data), 2017 IEEE International Conference on*, pages 4546–4554. IEEE.

Cha, S., Ruiz, M. P., Wachowicz, M., Tran, L. H., Cao, H., and Maduako, I. (2016). The role of an iot platform in the design of real-time recommender systems. In *2016 IEEE 3rd world forum on internet of things (WF-iot)*, pages 448–453. IEEE.

Chen, M., Mao, S., and Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications*, 19(2):171–209.

Clemente, J., Valero, M., Mohammadpour, J., Li, X., and Song, W. (2017). Fog computing middleware for distributed cooperative data analytics. In *Fog World Congress (FWC), 2017 IEEE*, pages 1–6. IEEE.

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

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15(1):3133–3181.

Frey, B. J. and Dueck, D. (2007). Clustering by passing messages between data points. *science*, 315(5814):972–976.

Galache, J. A., Yonezawa, T., Gurgen, L., Pavia, D., Grella, M., and Maeomichi, H. (2014). ClouT: Leveraging Cloud Computing Techniques for Improving Management of Massive IoT Data. In *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, pages 324–327. IEEE.

Harth, N., Delakouridis, K., and Anagnostopoulos, C. (2018). Convey intelligence to edge aggregation analytics. In *Studies in Computational Intelligence*, volume 715, pages 25–44. Springer, Cham.

Hernandez, L., Cao, H., and Wachowicz, M. (2017). Implementing an edge-fog-cloud architecture for stream data management. In *Fog World Congress (FWC), 2017 IEEE*, pages 1–6. IEEE.

Herrera-Quintero, L. F., Banse, K., Vega-Alfonso, J., and Venegas-Sanchez, A. (2016). Smart its sensor for the transportation planning using the iot and big-data approaches to produce its cloud services. In *Telematics and Information Systems (EATIS), 2016 8th Euro American Conference on*, pages 1–7. IEEE.

Huh, J.-H. and Seo, K. (2017). An indoor location-based control system using bluetooth beacons for iot systems. *Sensors*, 17(12):2917.

Karunaratne, P., Karunasekera, S., and Harwood, A. (2017). Distributed stream clustering using micro-clusters on Apache Storm. *Journal of Parallel and Distributed Computing*, 108:74–84.

Khan, A. N., Kiah, M. M., Khan, S. U., and Madani, S. A. (2013). Towards secure mobile cloud computing: A survey. *Future Generation Computer Systems*, 29(5):1278–1299.

Kliazovich, D., Pecero, J. E., Tchernykh, A., Bouvry, P., Khan, S. U., and Zomaya, A. Y. (2016). Ca-dag: Modeling communication-aware applications for scheduling in cloud computing. *Journal of Grid Computing*, 14(1):23–39.

Krause, A., Smailagic, A., and Siewiorek, D. P. (2006). Context-aware mobile computing: Learning context-dependent personal preferences from a wearable sensor array. *IEEE Transactions on Mobile Computing*, 5(2):113–127.

Lee, G., Saad, W., and Bennis, M. (2017). Online optimization for low-latency computational caching in fog networks. In *Fog World Congress (FWC), 2017 IEEE*, pages 1–6. IEEE.

Li, S., Da Xu, L., and Zhao, S. (2018). 5g internet of things: A survey. *Journal of Industrial Information Integration*, 10:1–9.

Liao, Y., Loures, E. d. F. R., and Deschamps, F. (2018). Industrial internet of things: A systematic literature review and insights. *IEEE Internet of Things Journal*, 5(6):4515–4525.

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

Manyika, J. (2015). *The Internet of Things: Mapping the value beyond the hype*. McKinsey Global Institute.

Marjani, M., Nasaruddin, F., Gani, A., Karim, A., Hashem, I. A. T., Siddiqa, A., and Yaqoob, I. (2017). Big iot data analytics: architecture, opportunities, and open research challenges. *IEEE Access*, 5:5247–5261.

Mekki, K., Bajic, E., Chaxel, F., and Meyer, F. (2019). A comparative study of lpwan technologies for large-scale iot deployment. *ICT express*, 5(1):1–7.

Molina-Masegosa, R. and Gozalvez, J. (2017). Lte-v for sidelink 5g v2x vehicular communications: A new 5g technology for short-range vehicle-to-everything communications. *IEEE Vehicular Technology Magazine*, 12(4):30–39.

Morabito, R., Cozzolino, V., Ding, A. Y., Beijar, N., and Ott, J. (2018). Consolidate iot edge computing with lightweight virtualization. *IEEE Network*, 32(1):102–111.

Mukherjee, A., Paul, H. S., Dey, S., and Banerjee, A. (2014). ANGELS for distributed analytics in IoT. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 565–570. IEEE.

Nahrstedt, K., Li, H., Nguyen, P., Chang, S., and Vu, L. (2016). Internet of mobile things: Mobility-driven challenges, designs and implementations. In *Internet-*

of-Things Design and Implementation (IoTDI), 2016 IEEE First International Conference on, pages 25–36. IEEE.

Nokia and Intel (2014). Increasing Mobile Operators Value Proposition With Edge Computing. `https://www.intel.co.id/content/dam/www/public/us/en/documents/technology-briefs/edge-computing-tech-brief.pdf`. [Online; accessed on 15-November-2017].

Patel, P., Intizar Ali, M., and Sheth, A. (2017). On Using the Intelligent Edge for IoT Analytics. *IEEE Intelligent Systems*, 32(5):64–69.

Qi, B., Kang, L., and Banerjee, S. (2017). A vehicle-based edge computing platform for transit and human mobility analytics. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, page 1. ACM.

Rahmani, A. M., Gia, T. N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., and Liljeberg, P. (2018). Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Future Generation Computer Systems*, 78:641–658.

Rao, B. B. P., Saluia, P., Sharma, N., Mittal, A., and Sharma, S. V. (2012). Cloud computing for Internet of Things & sensing based applications. In *2012 Sixth International Conference on Sensing Technology (ICST)*, pages 374–380. IEEE.

Raza, U., Kulkarni, P., and Sooriyabandara, M. (2017). Low power wide area networks: An overview. *IEEE Communications Surveys & Tutorials*, 19(2):855–873.

Ren, W., Ren, Y., Wu, M.-E., and Lee, C.-J. (2015). A Robust and Flexible Access Control Scheme for Cloud-IoT Paradigm with Application to Remote Mobile Medical Monitoring. In *2015 Third International Conference on Robot, Vision and Signal Processing (RVSP)*, pages 130–133. IEEE.

Rose, D. P., Ratterman, M. E., Griffin, D. K., Hou, L., Kelley-Loughnane, N., Naik, R. R., Hagen, J. A., Papautsky, I., and Heikenfeld, J. C. (2015). Adhesive rfid sensor patch for monitoring of sweat electrolytes. *IEEE Transactions on Biomedical Engineering*, 62(6):1457–1465.

Satyanarayanan, M. (2017). The emergence of edge computing. *Computer*, 50(1):30–39.

Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23.

Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646.

Sicari, S., Rizzardi, A., Grieco, L. A., and Coen-Porisini, A. (2015). Security, privacy and trust in internet of things: The road ahead. *Computer networks*, 76:146–164.

Sinha, R. S., Wei, Y., and Hwang, S.-H. (2017). A survey on lpwa technology: Lora and nb-iot. *Ict Express*, 3(1):14–21.

Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. (2017). Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434.

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

Stojmenovic, I. and Wen, S. (2014). The fog computing paradigm: Scenarios and security issues. In *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, pages 1–8. IEEE.

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*, pages 101–106. IEEE.

Sun, X. and Ansari, N. (2016). EdgeIoT: Mobile Edge Computing for the Internet of Things. *IEEE Communications Magazine*, 54(12):22–29.

Taneja, M., Byabazaire, J., Davy, A., and Olariu, C. (2018). Fog assisted application support for animal behaviour analysis and health monitoring in dairy farming. In *Internet of Things (WF-IoT), 2018 IEEE 4th World Forum on*, pages 819–824. IEEE.

van Dam, J.-F., Bißmeyer, N., Zimmermann, C., and Eckert, K. (2019). Security in hybrid vehicular communication based on its g5, lte-v, and mobile edge computing. In *Fahrerassistenzsysteme 2018*, pages 80–91. Springer.

Vieira, M. R., Barbosa, L., Kormáksson, M., and Zadrozny, B. (2015). Usapiens: A system for urban trajectory data analytics. In *Mobile Data Management (MDM), 2015 16th IEEE International Conference on*, volume 1, pages 255–262. IEEE.

Wang, C., Jiang, T., and Zhang, Q. (2016). *ZigBee® network protocols and applications*. Auerbach Publications.

Wang, T., Cardone, G., Corradi, A., Torresani, L., and Campbell, A. T. (2012). Walksafe: a pedestrian safety app for mobile phone users who walk and talk while crossing roads. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, page 5. ACM.

Welbourne, E., Battle, L., Cole, G., Gould, K., Rector, K., Raymer, S., Balazinska,

M., and Borriello, G. (2009). Building the internet of things using rfid: the rfid ecosystem experience. *IEEE Internet computing*, 13(3):48–55.

Yang, W., Wang, M., Zhang, J., Zou, J., Hua, M., Xia, T., and You, X. (2017). Narrowband wireless access for low-power massive internet of things: A bandwidth perspective. *IEEE wireless communications*, 24(3):138–145.

Yi, S., Li, C., and Li, Q. (2015). A survey of fog computing: concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*, pages 37–42. ACM.

Zaharia, M., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., Stoica, I., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., and Venkataraman, S. (2016). Apache Spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65.

Zhang, Y., Wang, H., and Xie, Y. (2017). An intelligent hybrid model for power flow optimization in the cloud-IOT electrical distribution network. *Cluster Computing*, pages 1–10.

Chapter 3

An edge-fog-cloud architecture of streaming analytics for IoT applications

This chapter has been published in the Special Issue Edge/Fog/Cloud Computing in the Internet of Things of Sensors Journal. The full citation of this published article is:

Cao, H., & Wachowicz, M. (2019). An Edge-Fog-Cloud Architecture of Streaming Analytics for Internet of Things Applications. Special Issue Edge/-Fog/Cloud Computing in the Internet of Things. *Sensors*, *19*(16), 3594.

Abstract

Exploring Internet of Things (IoT) data streams generated by smart cities means not only transforming data into better business decisions in a timely way but also

generating long-term location intelligence for developing new forms of urban governance and organization policies. This paper proposes a new architecture based on the edge-fog-cloud continuum to analyze IoT data streams for delivering data-driven insights in a smart parking scenario.

3.1 Introduction

Internet of Things (IoT) devices are usually equipped with many sensors, ranging from accelerometers and gyroscopes to proximity, light, and ambient sensors, as well as microphones and cameras. For smart cities, these devices are geographically distributed and can produce an overwhelming amount of data that poses a challenge for capturing, managing, processing and analyzing these data within a responsive acceptable time. In particular, analyzing IoT data streams generates location intelligence for many IoT applications in smart cities to engage actively with their citizens and enhance the city performance and reduce operational costs. However, this is a non-trivial process since we need a completely new IoT architecture that is capable of performing streaming analytical tasks running in parallel to provide timely approximate and accurate results.

Previous research has focused on pushing the data streams generated by IoT devices directly to a cloud environment, despite the inherited issues such as high latency, high data rates, low fault-tolerance and the unbounded order of incoming data streams (Cao and Wachowicz, 2019). Marz and Warren (2015) proposed the Lambda Architecture, a cloud architecture that provides scalability and fault tolerance for integration of data stream processing. The main purpose of this architecture was to cope with both *"volume"* and *"velocity"* dimensions of big data, which require complex computation-intensive processes to integrate streaming ana-

lytical tasks, making it unsuitable for IoT applications (Lin, 2017). Searching for simplicity, the Kappa Architecture was introduced to avoid using a batch processor by replacing it with a streaming processor able to handle data streams as an extended cache of the data flow into a cloud environment (Kreps, 2014). This cloud architecture may require larger in-memory storage space but it can be effective for IoT applications because it can handle fast data rates and handle retention times of the order of weeks (Wingerath et al., 2016).

However, IoT applications bring further fundamental and technological challenges. First, the time is ripe to rethink whether cloud computing is the only architecture able to support IoT applications, especially in the case of smart cities, where static and mobile IoT devices will be widely embedded in city infrastructure. It is worth investigating an overall orchestration of the computational resources available today that can take advantage of the edge-fog-cloud continuum to guarantee a seamless execution of automated analytical tasks without compromising the accuracy of their outcomes. Second, managing retention times between automated analytical tasks is critical for handling high/low latency of existing data life-cycles that are encountered when supporting IoT applications. However, the real advantage is not at all about latency versus throughput but rather about allowing smart cities to develop, test, debug and operate their IoT applications on top of a single analytical framework.

This paper proposes an *"Analytics Everywhere"* framework that encompasses the edge-fog-cloud continuum to support streaming analytics for maximizing the potential insights from IoT data streams. A new IoT architecture is proposed based on a conceptual framework that is particularly useful for integrating IoT devices using the edge-fog-cloud continuum. It consists of three elements that can be considered as the main criteria to take into account in order to determine whether an

edge-fog-cloud environment is required by an IoT application. They can be described as follows:

- Resource capability: This element consists of organizing distributed compute nodes (i.e., cloud, fog and edge nodes) that will provide a message broker, data link, IoT device connector, data flow editor, parser, Machine Learning (ML) libraries , in-memory data storage and power for the execution of streaming tasks. Geographically adjacent compute nodes deployed at the edge, fog and cloud will be usually connected through a plethora of communication networks.
- Analytical capability: This element selects the best practice methods/algorithms for the orchestrated execution of analytical tasks that are vital to meet the requirements of IoT applications. The compute nodes are needed to perform *a priori* known analytical tasks to collect, contextualize, process and analyze data from IoT devices.
- Data life-cycle: This component describes the changes that data streams go through during the execution of analytical tasks.

The scientific contributions of this paper can be summarized as follows:

- Most of the IoT architectures rely on a cloud environment in which n-tiers of horizontal layers are designed to perform analytical tasks. Our approach proposes a new architecture based on an integrated fabric of compute nodes that are designed to work together to perform many analytical tasks, which are triggered by IoT data streams transported through an edge-fog-cloud continuum.
- Automated analytics for IoT data streams is still in its infancy and applications usually require a diverse number of outputs having different temporal

granularities. There has been very little research reported on the impact of analytical tasks in the IoT architectures. The scientific contribution of our research is therefore to ascertain this impact using a smart parking scenario.

The remainder of this paper is organized as follows: Section 3.2 reviews the existing architectures, processing and analytical frameworks for handling IoT data streams. Section 3.3 introduces the main concepts of our proposed *"Analytics Everywhere"* framework. Section 3.4 describes the developed IoT architecture for analyzing the incoming data at anywhere and in anytime. Section 3.5 describes the smart parking scenario used to validate the proposed architecture. The main outcomes of the analytical tasks are shown in Section 3.6. Section 3.7 concludes our research and discusses further research.

3.2 Related Work

It is challenging to analyze vast amounts of incoming IoT data streams. Over 400 architectures have been proposed in the literature to handle incoming IoT data streams using different strategies such as *stream*, *micro-batch* and *batch processing* (Wingerath et al., 2016; Cao and Wachowicz, 2017). The most important issue in selecting an IoT architecture is to balance the trade off between throughput and latency. However, most approaches to handle this trade off are based on a cloud computing environment where IoT data streams are pushed to and accumulated over a long period of time and are later processed and analyzed in batches.

Batch-oriented processing frameworks have been efficiently used for processing large amounts of historical IoT data with high throughput but also with high latency. For example, one of the most common and widely used cloud architectures

for batch-oriented processing that supports distributed storage across many clusters of commodity servers is the Hadoop MapReduce framework (Dittrich and Quiané-Ruiz, 2012). Another example is Spark (Zaharia et al., 2016) which has the ability to perform large-scale batch processing in memory using resilient distributed data sets.

Aiming to increase efficiency, *micro-batch frameworks* buffer and process IoT data streams in batch. For example, Spark Streaming restricts the batch size in a processor where each batch contains a set of events that arrived online over the batch period (regardless of the event's time). However, it will obviously increase the time the data streams spend in the data pipeline. In contrast, *stream-oriented frameworks* typically provide time-sensitive computations but have relatively high data processing costs on a continuous stream of IoT data. Stream-oriented processing architectures usually avoid putting data at rest. Instead, they minimize the time a single tuple should spend in a processing pipeline. Some typical stream processing frameworks are Storm, Samza and Flink (Toshniwal et al., 2014; Carbone et al., 2015; Noghabi et al., 2017).

From an analytics perspective, IoT data streams that are accumulated for a long period of time can be analyzed in batches using traditional algorithms in machine learning and data mining such as clustering, classification, regression and dimensionality reduction, to name a few. For example, Ismail et al. (2018) proposed a MapReduce based mining algorithm to facilitate Parallel Productive Periodic Frequent Pattern mining of health sensors data. Ta-Shma et al. (2018) also described an attempt to ingest and analyze IoT data streams using open source components. Their simplified architecture is a combination of several instances that install an event processing framework, a batch analytics framework, a data storage framework and a message broker to handle both batch and streaming data flows. Santos et al.

(2018) proposed an e-health monitoring architecture based on sensors that rely on cloud and fog infrastructures.

Recently, a paradigm shift has emerged in the evolution of IoT architectures aiming at analytics, software and platform configuration. Streaming analytics algorithms are being developed to extract value from IoT data streams as soon as they arrive at a computational resource. However, it is a non-trivial task to extract insights online, since the nature (or distributions) of IoT data streams change over time due to the geographical location of IoT devices (De Francisci Morales et al., 2016). Moreover, streaming analytical algorithms must work within limited resources (time and memory). Some open source frameworks for IoT data stream analytics are being developed including MOA, SAMOA and skit-multiflow (Montiel et al., 2018; Morales and Bifet, 2015; Bifet et al., 2011) using only streaming processors.

Our proposed architecture is a step forward in finding a unique solution that combines the advantages of different computational resources into an integrated edge-fog-cloud fabric that is capable of capturing, managing, processing, analyzing and visualizing IoT data streams. This fabric of computational resources is designed to work towards an asynchronous approach for supporting an *"Analytics Everywhere"* framework (Cao et al., 2019) making the development, deployment and maintenance more pragmatic and scalable. By breaking down the processing and analytical capabilities into a network of streaming tasks and distributing them into an edge-fog-cloud computing environment, our proposed architecture can support streaming descriptive, diagnostic and predictive analytics.

3.3 Analytics Everywhere Framework

We propose an *"Analytical Everywhere"* concept as a conceptual framework that integrates a variety of computational resources for a flexible orchestration platform that has functionality around containers. The primary goal is a seamless and automated execution of automated analytical tasks founded on a data life-cycle of an IoT application. This framework consists of three elements: Resource Capability, Analytical Capability and a Data Life-cycle which are described in more detail in the following sections.

3.3.1 Resource Capability

In general, an IoT application will require a combination of different compute nodes running at the edge, fog and/or cloud. The main criteria to take into account when selecting a compute node have been first introduced by Cao et al. (2019). They are described as follows:

- Vicinity: The geographical proximity of compute nodes to an IoT device is an important criterion to take into consideration for an IoT application. Since IoT devices can be static (i.e., deployed inside a building) or mobile (e.g., deployed in a car) and their distance to a compute node might vary, our *"Analytics Everywhere"* framework is based on the principle that IoT devices are related to everything else. Therefore, compute nodes near IoT devices are more closely related than distant ones (First Law of Geography). In particular, edge nodes should be located near the IoT devices and should use short-range networks for the data streams.

- Reachability: The time to reach a compute node via a network varies accordingly to the type of IoT devices and the communication network. Typically, if a compute node is connected to the Internet with a fixed IP address, this can be considered a highly reachable resource (i.e., it takes relatively little time to reach the compute code), rather than if it is connected using a private network and behind a NAT.

- In-memory and storage: This criterion handles the amount of data in a compute node that should be kept in memory or stored in a database. The retention time of IoT data streams is expected to vary according to the IoT application requirements as well as the available memory size. The final decision will also depend on the bandwidth and the throughput required by an IoT application. The actual amount of data which has been transmitted varies, as there could be many different factors (i.e., latency affecting throughput). The latency is clearly low at the edge due to the proximity to the IoT devices and increases as we move to the cloud.

- Computation: The amount of processing power available at a compute node for performing a set of automated analytical tasks. Taking into account the IoT application requirements can help in making a decision about which compute node to use for executing these tasks.

- Standardization: This aspect represents the most important criterion yet to be met in the implementation of IoT applications. Different standards can be applied in an IoT application ranging from network protocols and data-aggregation to security and privacy standards.

While computation and memory capabilities can increment as the analytical tasks are executing from the edge to the cloud, reachability must be always

considered for an analytical task. Reachability is a critical dimension that requires analytical tasks to return results in a timely way, dependently of computational resources. Because fog nodes play the role of the intermediate resources that seamlessly integrate edge and cloud resources, the resource contention in the compute nodes and the communication links can be easily eliminated. In contrast, the proximity of the edge nodes to IoT devices can assist the necessary scaling of IoT applications, turning them into an essential computational resource for supporting near or real-time data analytics. Nevertheless, the immaturity of standards in edge resources and IoT devices are currently impeding the implementation of IoT applications.

3.3.2 Analytical Capability

We propose an *"Analytics Everywhere"* framework that can be applied to develop a variety of analytical tasks to perform descriptive, diagnostic and predictive analytics using IoT data streams. Streaming analytics are used to provide higher-level information about IoT data streams at the edge, the fog or the cloud. The aim is to generate new insights as demanded by an IoT application in order to answer the questions: *"What is happening in the real-world?"* (Streaming Descriptive Analytics); *"Why is it happening?"* (Streaming Diagnostic Analytics) and *"What will happen?"* (Streaming Predictive Analytics). The main goal of our *"Analytics Everywhere"* framework is to automate *a priori* known analytical tasks that will be executed at the edge, the fog and the cloud in order to answer these questions.

Figure 3.1 illustrates some analytical tasks that may be required for supporting an IoT application (green node: analytical tasks performed at the edge; orange node: analytical tasks performed at the fog; blue node: analytical tasks performed at the cloud). The analytical tasks have different levels of complexity and require a suitable data life-cycle to support multiple paths of computation ranging

from data cleaning and data aggregation tasks that require a continuous stream of data, to more complex tasks such as data contextualization and data summarization tasks that require accumulated data streams for time-sensitive results. Streaming descriptive analytics may be performed at the edge, the fog and the cloud; however, we anticipate that they will more often be executed at the edge because *(i) IoT data streams have tiny volume at the edge* and *(ii) many IoT applications will prevent data from being moved to a cloud due to privacy and costs concerns.*

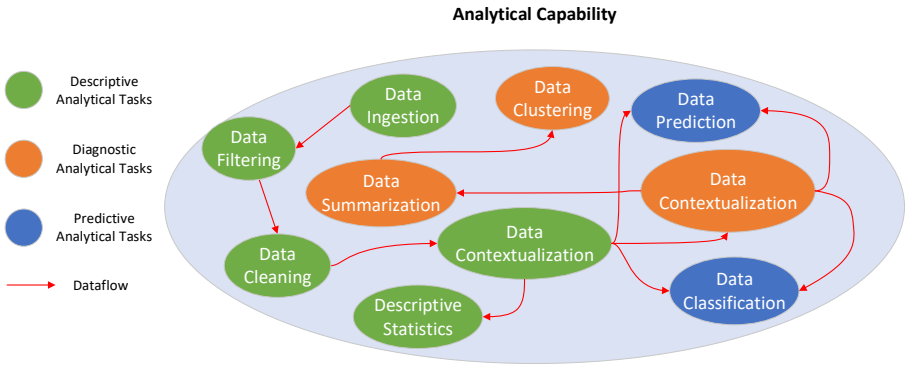


Figure 3.1: Overview of streaming tasks

Streaming diagnostic analytics can be executed near to or far from an IoT device, depending on where it is more feasible to install relatively powerful computational resources. Streaming diagnostic analytical tasks are usually supported by a few on-line algorithms, stream clustering algorithms, ad-hoc queries and continuous queries. Fog and cloud resources are expected to be used to perform streaming diagnostic analytics since they provide computation, storage and accelerator resources that are more suitable than edge nodes to perform the streaming tasks. Fog and cloud computing can improve the accuracy and reduce the computational complexity of the automated tasks in near real-time. Streaming predictive analytics requires on-demand analytical tasks with high availability and rapid elasticity through the virtually unlimited resources of the cloud; the analytical tasks are expected to use a huge amount of historical IoT data that need to be processed according to the

nature of IoT applications.

3.3.3 Data Life-Cycle

We expect many types of data life-cycles depending on the types of analytical tasks and compute nodes needed by an IoT application. Therefore, a data life-cycle can be either stateful or stateless depending on the orchestration requirements of an IoT application. A stateless data life-cycle treats each analytical task independently and creates output data tuples depending only on the input data tuples of that analytical task. On the contrary, stateful data life-cycles combine different analytical tasks together and create the output data tuples based on multiple input data tuples taken from those analytical tasks. Moreover, data scientists must also specify a reliability mode that can follow three approaches:

- At most once: There is no guarantee that data tuples in a stream are being handled at most once by a streaming task of an IoT application. If a failure takes place at the edge, fog or cloud nodes, no additional attempts are made to re-handle these data tuples. The assumption is that the throughput (i.e., the actual amount of data that has been transmitted between compute nodes) exceeds the maximum bandwidth. In other words, there could be different factors such as latency affecting throughput.

- At least once: The data tuples in a stream are guaranteed to be each handled at least once by all streaming tasks of an IoT application. If a failure happens, additional attempts are needed to to re-handle these data tuples. This approach may cause unnecessary duplication of data tuples in the streams but it has been widely adopted for cloud processing (e.g., Storm and Samza).

- Exactly once: Exactly once means that data tuples are guaranteed to be handled exactly the same as they would be in the failure-free scenario, even in the event of various failures.

The edge-fog-cloud continuum brings a high complexity in connecting and in orchestrating several compute nodes; therefore our *"Analytics Everywhere"* framework currently supports a stateless data life-cycle having the "at most once" approach for guaranteeing low latency for running analytical tasks of an IoT application. Two main computation paths can be found in our data life-cycle:

- Computation Path 1: analytical tasks that need user-defined Windows (batches) for accumulating data streams in order to generate outputs. Data aggregation and clustering are examples of analytical tasks that require this type of path, also called batch processing.
- Computation Path 2: analytical tasks that run using continuous data streams in order to generate outputs. Some examples of analytical tasks include data cleaning, data filtering and data duplication. This path has also been previously referred to as stream processing.

More information about how these computation paths have been applied to a Smart Parking application can be found in Section 3.5.2.

3.4 The Streaming IoT Architecture

Resource capabilities play an important role in designing an IoT architecture that relies on the edge-fog-cloud continuum for running automated analytical tasks that

have a data life-cycle with streaming data tuples as input and output. We propose a geographically distributed network of compute nodes that have a combination of modules including Admin/Control, Stream Processing & Analytics, Run Time, Provision & Orchestration, and Security & Governance (Figure 3.2). Our IoT architecture enables micro-services to run at various compute nodes in such a way that each micro-service can perform a specific analytical task depending on which module it belongs to. It is important to point out the essential role of the Admin/-Control module of our IoT architecture, since it optimizes the data flow in order to implement a data life-cycle that takes into account the individual requirements of an IoT application. Therefore, we have also integrated data management, visualization, orchestration, and security modules in our IoT architecture.

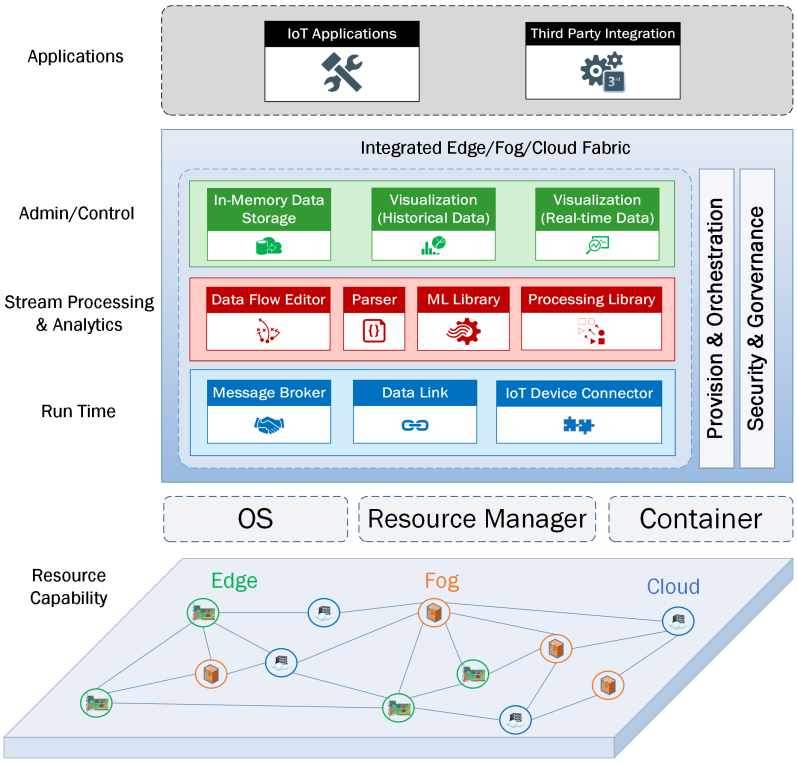


Figure 3.2: The proposed Internet of Things (IoT) architecture for our *"Analytics Everywhere"* framework.

3.4.1 Stream Data Tuples

We propose an IoT architecture focusing on processing data streams which are defined as a sequence of tuples that usually contain 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_n when a measurement has occurred.

The main characteristics of tuples can be described as one of the following:

- Each tuple in a stream arrives online. An effective architecture begins by prioritizing routing the streaming data tuples to the distributed compute nodes. This is achieved by keeping records of the ingestion times when a tuple arrives at compute nodes located at the edge, the fog or the cloud.
- An architecture has no control over the order in which a tuple arrives at a compute node. When an analytical task is automated and continuous queries are needed by an IoT application,the ingestion times play an important role in making sure all streaming data tuples implicitly belong to a user-defined window. In other words, the order of the tuples coming from an IoT device does not matter; however the order of the ingestion timestamps matter because we should not have a tuple arriving at the cloud and having an earlier ingestion timestamp from a tuple arriving at the edge.

3.4.2 Main Processing Modules

The main modules can be categorized as Run Time, Stream Processing & Analytics, and Admin/Control.

3.4.2.1 Run Time

Message Broker: In our IoT architecture, the message broker is a software/middleware computer program module that reliably routes messages between clients using a formal messaging protocol and providing metadata about connected clients such as the data they are streaming and/or the actions they expose with guaranteed QoS delivery. They can also communicate with other modules, such as queries, Data Flow Editor, In-memory Databases and applications such as enterprise apps or analytical dashboards.

Data Link: A data link is a wrapper with a domain-specific library or functionality, that is exposed to the communication network. A data link provides an interface to access streaming data tuples from different data sources and sinks into and out of the compute nodes. It can be a device link, bridge link or an engine link. The device data links allow the capability to connect specific IoT devices together (e.g., WeMo devices, beacons, sensors). The bridge data links offer two-way communications with other publish-subscribe protocols (e.g., MQTT, AMQP, STOMP). The engine data links contain logic functions/drivers or provide access to the processes that provide specific functionality (e.g., JDBC, ODBC).

IoT Device Connector: This module manages the network connection

between IoT devices and compute nodes. There are two main options to deploy device connector modules depending on the requirements of an IoT application: they can be described as a horizontal or as a vertical option. In the horizontal device connectors, the main components of a data stream management platform are horizontally deployed across remote nodes. In contrast, vertical device connectors not only expand their services to the edge but also scale the data stream management components to the nodes close to the IoT devices. In our architecture, we combine both horizontal and vertical options to guarantee a unique architecture based on a network of IoT devices and compute nodes.

3.4.2.2 Stream Processing & Analytics

Data Flow Editor: The data flow editor is a visual data manipulation environment for wiring together IoT devices, APIs and services. It allows developers to create a data-flow model based on a set of programming blocks that perform the assigned analytical tasks when requirements are met. A data-flow model can be considered as a broker client because it can subscribe to data from different data sources and publishes results to the broker. Therefore, the data flow editor is designed to support a data-flow model to be deployed to the run time in a convenient manner.

Parser: The streaming data tuples can continuously bounce from one compute node to another. The goal of the parser module is to transform or serialize the tuples into a series of bytes to be stored or transmitted across a network then reverse or de-serialize them back to their original form when they reach their final destinations. Therefore, the data streams need a syntax for storing and exchanging data that is not only

convenient for developers to read and write but also easy for machines to parse and generate.

Machine Learning Library: The main element of this module is the Online Learning Library. In contrast to batch machine learning which trains the input data, builds and evaluates the model as a bundle, the Online Learning Library is used to evaluate the current streaming data on-the-fly as they enter the compute node, and to gradually build the learning model based on the incoming data tuples over time.

Processing Library: This engine mainly deals with the continuous arrival of data tuples. It includes the Complex Event Processing (CEP) component and Structured Streams Processing (SSP) component to manage and transform the raw data tuples. The SSP component is used to build programs that implement operations and analytical tasks on data tuples (e.g., cleaning, filtering, updating state, defining windows, aggregating). The CEP component allows us to detect event patterns in an endless stream of events.

3.4.2.3 Admin/Control

Data Visualization: This module provides two main services: the monitoring service and the exploring service. The monitoring service is used to plot real-time data whenever they arrive at our system, with the aim of early detection of abnormalities. The exploring service plots processed/historical data with the aim of assisting us with analysis and discovering new insights.

In-Memory Data Storage: The in-memory storage space is where the incoming data tuples and/or the results of the analytical operations reside. The storage space can be different types of in-memory databases (e.g., document-based store, key-value store) or an in-memory file system.

3.5 Validating the Proposed Architecture

In this section, we validate our proposed architecture using a smart parking application. We describe in detail the software components used to implement the main modules in the integrated edge-fog-cloud fabric. Also, the data life-cycle implementation and the IoT data streaming mechanism between each nodes in the architecture are explained in detail.

3.5.1 Smart Parking Application

A smart parking application was selected to evaluate our IoT architecture because it combines communication and information technology to help drivers find available parking spaces. Studies have shown that integrating smart parking into the city framework can shorten parking search time, reduce emissions and fuel consumption and decrease traffic congestion. The application consists of IoT data streams generated in real-time whenever a driver parks his/her car and uses the mobile application of the HotSpot Parking system which is being used in the city of Saint John, NB, Canada (Figure 3.3). The data streams are fetched by the edge nodes which are geographically installed close to the pay station facilities in the city. Afterward, the data streams are sent to a fog node located at City Hall. Finally, the data arrives at a Data Center provided by Compute Canada West Cloud as the IaaS resource,

located in Vancouver. They are configured to communicate together as a network of nodes. The detailed specifications of each compute node are available in Table 3.1.

Table 3.1: The overview of the compute nodes.

	Edge Node	**Fog Node**	**Cloud Node**
OS	Ubuntu Mate	Window Server	CentOS 7.0 (x86_64)
CPU	ARM Cortex-A53	Intel Xeon E5-2623 v3	Intel Xeon E5-2650 v2
# of Core	4 (1.4 GHz 64-bit)	4 (3.00 GHz 64-bit)	8 (2.60 GHz 64-bit)
RAM	1 GB	30 GB	30 GB
Disk	32 GB	1 TB	1 TB
Hardware	Raspberry Pi 3 B+	Commodity Server	Virtual Machine

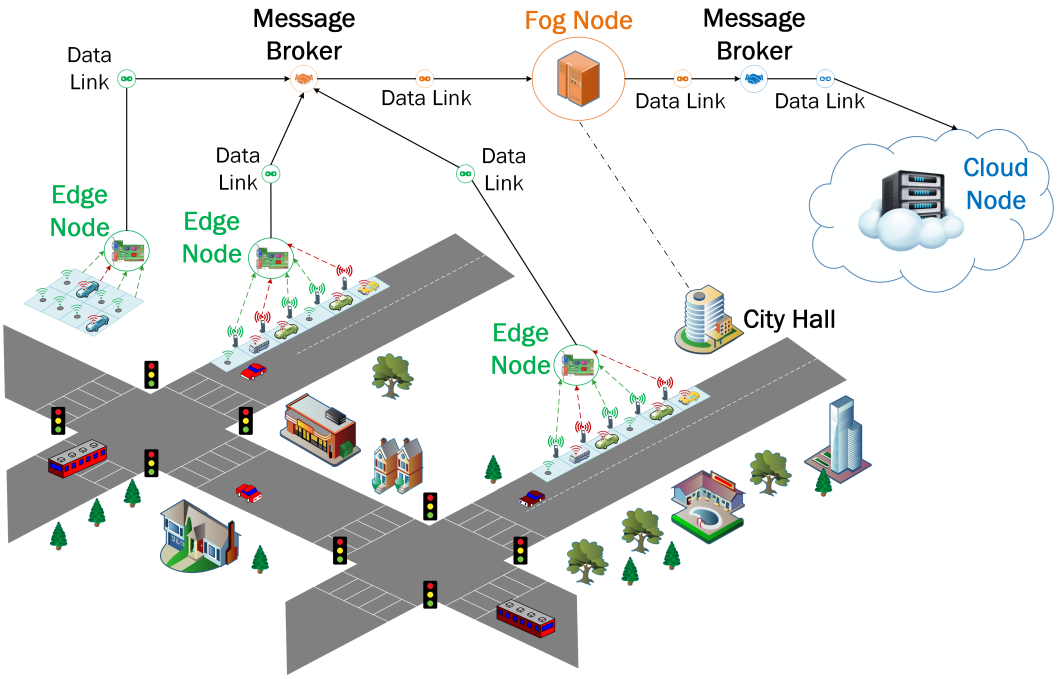


Figure 3.3: Geographical Distribution of the edge-fog-cloud nodes for the smart parking application.

Different modules have been used to implement an integrated edge-fog-cloud fabric of compute nodes for the Smart Parking application. We have implemented a variety of open source modules and commercial software packages to deploy the proposed IoT architecture. Each of them plays an important role as a module in the overall architecture. The implementation is illustrated in Figure 3.4.

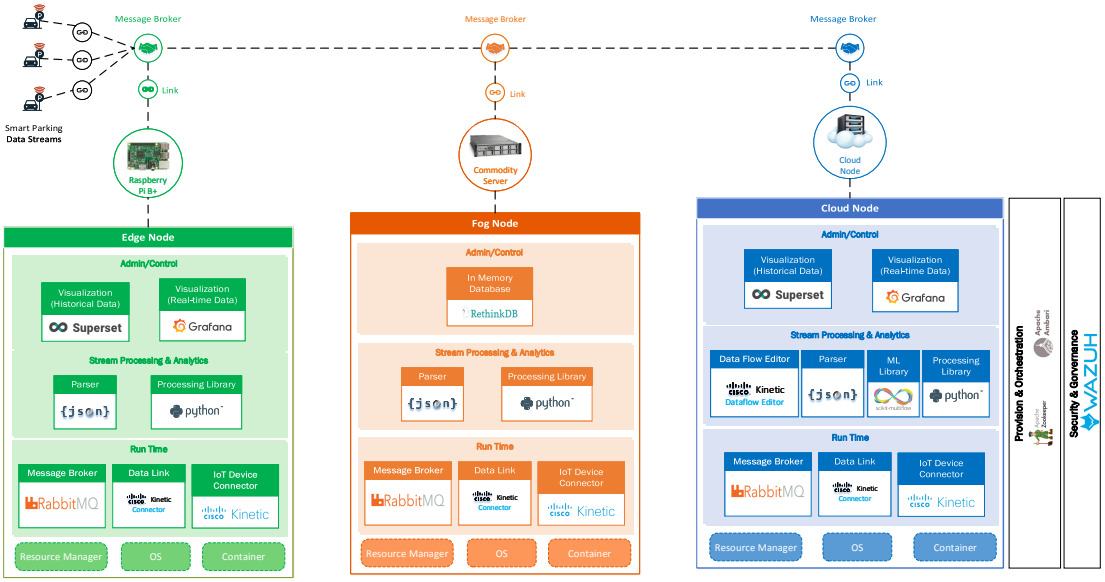


Figure 3.4: Implementation of the architecture for the smart parking application.

The software used for this implementation are summarized in Table 3.2 below.

Table 3.2: Software used for the modules implementation.

Run Time	*Message Broker*	**RabbitMQ**: It is an open source streaming platform that supports different message brokers to provide fault-tolerant, scalable, high-throughput and low-latency data pipelines of queuing real time IoT data streams using a publish-subscribe mechanism.
	IoT Device Connector	**Cisco Kinetic**: This is a scalable, secure commercial system that can be used to extract, compute and move the data tuples to the right applications at the right time. There are three integral parts of the Cisco Kinetic platform: Edge & Fog Processing Module (EFM), Gateway Management Module (GMM) and Data Control Module (DCM).
	Data Link	**Cisco Kinetic Connector**: As a feature of EFM, Cisco Kinetic Connector provides a wide array of data links developed by Cisco, Third Party, and Open Source Community. It supports connectivity between compute nodes, message brokers, and IoT devices.
Stream Processing & Analytics	*Data Flow Editor*	**Cisco Kinetic Dataflow Editor**: This is a feature in EFM that can be used to customize, modify, and manage data flows with a graphical layout. It also offers a convenient interface to create and debug data flows.
	Parser	**JSON parser**: JSON objects are mainly exchanged between the compute nodes in our system. Therefore, the parser is used to encode the data structures to JSON strings and decode them back to dictionary, list, tuple, boolean or other numerical data types.
	Stream ML Library	**Scikit-Multiflow**: It offers main packages to assist the users with handling and learning from their data streams such as stream generators, learning methods, change detectors, and evaluation methods.
	Processing library	**Python**: For dealing with structured incoming data streams and detecting different data patterns, we have developed the algorithms to take action when the events happen. A variety of built-in Python libraries, such as numpy and scipy, were used to develop our algorithms.
Admin/Control	*In-memory Database*	**RethinkDB**: It is an open-source, distributed document-oriented database for real time changing feeds. It allows the developers to push the continuous queries to retrieve the results in real time using ReQL query language.
	Visualization (Historical Data)	**Superset**: Aiming to extract the insights from the historical/processed data, we have employed Superset, which is a new ongoing incubation at the Apache Software Foundation.
	Visualization (Real-time Data)	**Grafana**: It is an open-source platform capable of monitoring and analyzing the dynamic data incoming from IoT devices, which we used for our streaming real-time data.
Provision & Orchestration		Aiming to mitigate difficulties in managing, distributing and updating the system, we have installed Apache Ambari and Apache Zookeeper in our network of compute nodes. The Apache Ambari package is then used to configure and install the other main modules of our IoT architecture.
Security & Governance		For the security, we have also configured Wazuh which is an open source system for integrity monitoring, and threat and intrusion detection to protect our compute nodes. It consists of many functions such as security analytics, vulnerability detection, file integrity monitoring, and configuration assessment.

The *"Analytics Everywhere"* framework is implemented to assist Hotspot in providing a more convenient, reliable, and professional parking service for drivers, and to assist the City of Saint John, Canada improve their parking facilities. We have selected the following analytical capabilities:

- Streaming Descriptive Analytics: What is the problem with smart parking in Saint John?
- Streaming Diagnostic Analytics: Why are these parking usage/frequency patterns an issue in Saint John?
- Streaming Predictive Analytics: What could be improved in the future?

3.5.2 Data Life-Cycle Implementation

Mapping between analytical tasks and compute nodes (edge-fog-cloud continuum) for executing a data life-cycle is a non trivial task because it requires careful orchestration and a precise allocation of resources. To ease the complexity of the mapping process, a data life-cycle describes the changes that stream data tuples go through during the automated execution of analytical tasks. The Smart Parking application requires a unique data life-cycle as shown in Figure 3.5. Moreover, all of the analytical tasks are fully triggered and performed in an automated manner as soon as the stream data tuples arrive at any compute node.

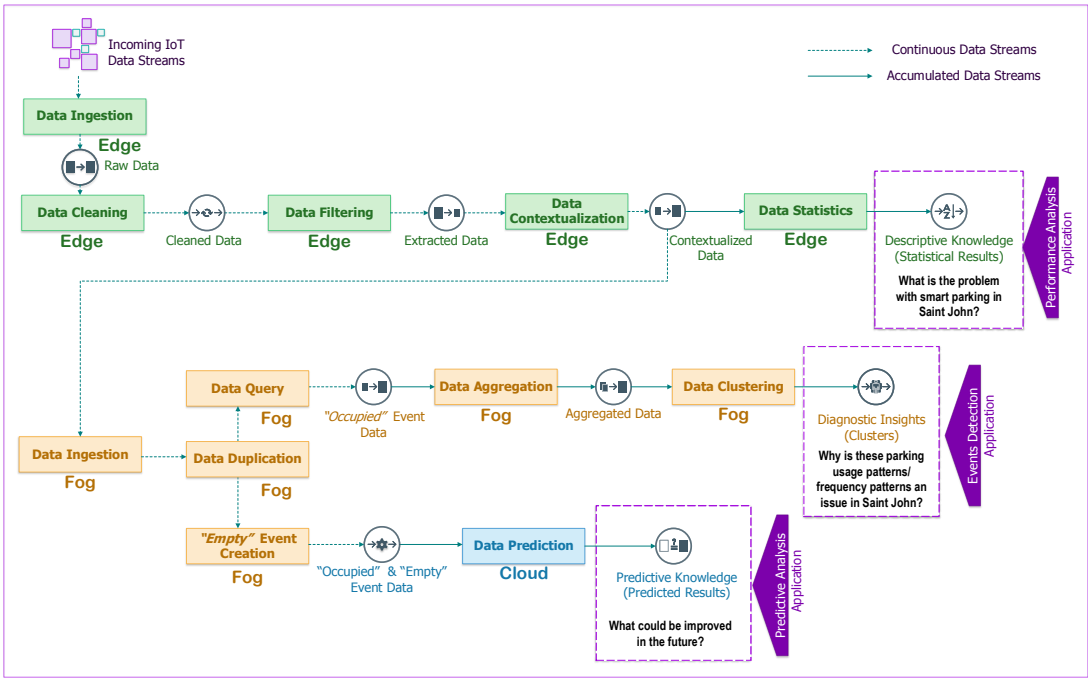


Figure 3.5: Overview of the implemented data life-cycle.

3.5.2.1 Analytical Tasks in Continuous Data Streams

The data ingestion task deployed at an edge node will retrieve parking data by defining a forever loop to iteratively trigger this task every 5 s. A raw streaming data tuple is considered a parking space event which will be sent to the closest edge node. The parking data streams consist of a set $\{T_1, \ldots, T_n\}$ of out-of-order tuples containing attributes in the format:

$$T_i = \langle PE_i, SE_i \rangle \tag{3.1}$$

where:

- PE_i: a specific parking event containing 4 attributes {*spot_id, length, startTime, vehicle_id*} described in Table 3.3.
- SE_i: a parking spot entity where the parking event is happening. It contains

3 attributes {*lat, long, spot_name*} described in Table 3.3.

Table 3.3: The description of data tuples.

Data Fields	**Attribute Name**	**Description**
Parking Event	spot_id	The parking spot ID in the parking event table
	length	Total parking duration (hours) when a driver parks his/her vehicle
	startTime	A timestamp indicating the start time of the parking process
	vehicle_id	Vehicle ID in the parking event table
Spot Entity	lat	Latitude of the parking spot
	long	Longitude of the parking spot
	spot_name	The conventional name of the parking spot given by the City

The raw data tuples obtained after the data ingestion task will be forwarded to the data cleaning task, which consists of a sequence of operations including assessment, detection, repairing, and validation. The assessment process can detect and identify errors, redundancies, missing values, incomplete tuples, and inaccurate data fields. The tuples are re-organized, replaced, repaired or removed using adaptive integrity constraints in a dynamic sense to ensure data quality. Finally, validating the accuracy of the data tuples once they have been cleaned is an important operation before passing them to the next analytical task.

The attributes of a cleaned data tuple are later grouped into two new data fields (Parking Event and Spot Entity). Our new data tuple now becomes a set of attributes $\{T'_1, \ldots, T'_n\}$ in which each $T'_i = \langle s_1, \ldots, s_7 \rangle | i$ contains a vector of 7 corresponding attributes {*spot_id, length, startTime, vehicle_id, lat, long, spot_name*}.

We have implemented an autonomous script to logistically apply adaptive integrity constraints to handle missing attribute's values and tuples, to remove duplicate tuples and redundant attributes, and to repair incorrect attribute values. The cleaned tuples are then transferred to the data filtering task as illustrated in Figure 3.5.

The data filtering automatically derives a subset of data from the original

one using a set of criteria or extraction (filtering) operations. After finishing the data filtering task, the extracted data will be transferred to the data contextualization task to create new attributes and attach them to the original data tuples T using a contextualization operation Ψ as in Equation (2).

$$\begin{cases} \forall T'_i \in (T'_1, T'_2, \ldots, T'_n) : T'_i = \langle s_1, \ldots, s_7 \rangle | i \\ \mathbb{D} = (T'_1, \ldots, T'_n) \xrightarrow{\Psi} \overline{\mathbb{D}} = (P_1, \ldots, P_n) \\ \forall P_i \in (P_1, \ldots, P_n) : P_i = (S_i, x_i, y_i, t_i, Context_1, Context_2, \ldots) \end{cases} \tag{3.2}$$

The data contextualization task has been implemented at the edge to handle the current incoming data tuples and at the fog node to handle the outdated data tuples as described in Figure 3.5. A function was implemented to interpret the status (occupied or empty) of a parking spot whenever a driver parked his/her car.

$$f(T') = \begin{cases} T' = T' \cup s_8 & \text{where } s_8 = Occupied \\ T' = T' \cup s_9 & \text{where } s_9 = startTime + length \\ T' = T' \cup s_{10} & \text{where } s_{10} = edge\_arrivingTime \end{cases}$$

- Whenever a tuple arrives at the edge, we create an event label as *Occupied* and attach to the original tuple to mark that a parking spot is in use.
- We compute the *endTime* using the *startTime* and the parking duration *length*. The parking duration is the one paid by the customer.
- We also add the arriving time *edge_arrivingTime* whenever a tuple arrives at an edge node.

After the contextualization task at the edge has been executed, three new attributes s_8, s_9, s_{10} are attached to the original tuple. The contextualized tuples become $T'_i = \langle s_1, \ldots, s_{10} \rangle | i$ containing a vector of 10 attributes {*spot_id, length, start-*

Time, vehicle_id, lat, long, spot_name, ***event, endTime, edge_arrivingTime***}. This new contextualized tuple will be transmitted to the fog node where a new attribute, s_{11}, will be added for registering the ingestion time. At the fog, this *Occupied* data tuple is duplicated for two main purposes: (1) one copy of the *Occupied* data tuple is transmitted to accumulated data streams for further analytical tasks; (2) the other *Occupied* data tuple copy temporarily resides at the in-memory database for deducing other events.

In this smart parking application, outdated and current incoming *Occupied* data tuple are the important elements to determine the status of a parking event whenever a driver parked his/her car. We aim to infer whether an *Empty* event or an *Occupied* event is occurring at a specific parking spot.

The *Empty* event is also computed at a fog node as shown in Figure 3.6. The computation consists of the following steps:

- When a contextualized tuple T'_{i-1} with an *Occupied* status arrives at the fog, it is treated as an outdated tuple and retained in database (RethinkDB) until a new tuple T'_i of the same parking spot arrives. To detect the changes in our real time database, we have implemented an adhoc query using ReQL language to continuously monitor the incoming tuple as follows.

Description	**ReQL Statement**
Monitoring the feed if any new object changes on a table	r.**db**('spdb'). **table**('raw_historical_table'). **changes**.**run**(conn). each{|change| p(change)}

- The new tuple T' with an *Empty* status is initially computed by mirroring some static attributes from the incoming tuple T'_i including {*spot_id, lat, long, spot_name, edge_arrivingTime*}. Then, the *startTime* of tuple T' is assigned

by the *endTime* of tuple T'_{i-1} while the *endTime* of tuple T' is assigned by the *startTime* of tuple T'_i. The *length* of tuple T' is then computed by subtracting its *endTime* from its *startTime*. Finally, the *fog_arrivingTime* of tuple T' is attached at the end of the *Empty* tuple creation task. The following query command is used to retrieve the outdated *Occupied* tuple that temporarily resided in RethinkDB for this task.

Description	**ReQL Statement**
Query the outdated *"Occupied"* tuple that temporarily resided in RethinkDB	r.**db**('spdb'). **table**('raw_historical_table'). **without**('id', 'edge_arrivingTime', 'fog_arrivingTime'). **filter**({"spot_id": str(item['spot_id']), "event": "Occupied"}). **order_by**(r.desc('startTime')).limit(1).run(conn)

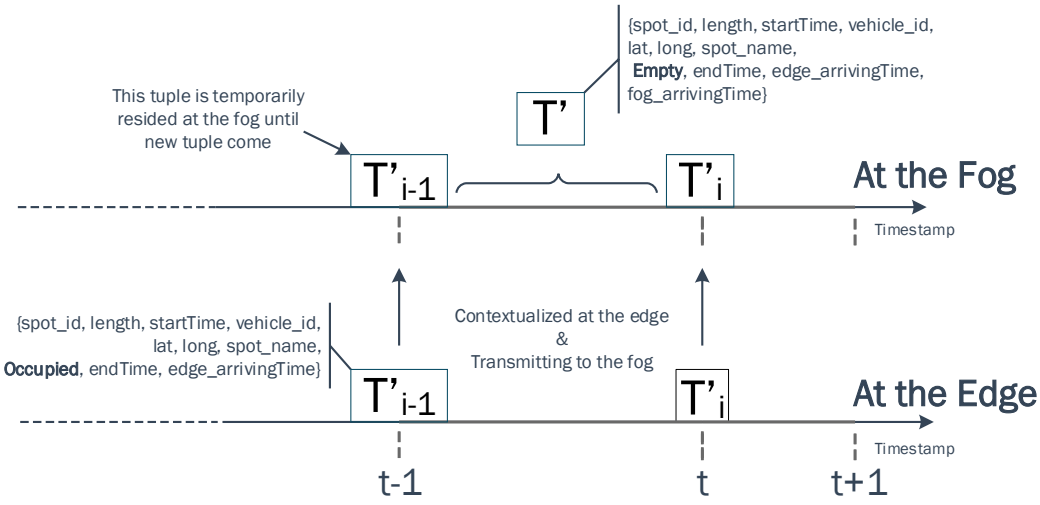


Figure 3.6: The process of computing an *Empty* event at the fog.

Once the data query task and the *Empty* event creation task at the fog are completed, all outdated *Occupied* data tuples, current incoming and new tuple will contain a vector of 11 attributes $\langle s_1, \ldots, s_{10}, s_{11} \rangle$ corresponding with {*spot_id, length, startTime, vehicle_id, lat, long, spot_name,* ***event, endTime, edge_arrivingTime, fog_arrivingTime***}. These event data tuples will be transmitted to the data summarization task at the fog and the data prediction task in the cloud for further

analytics as indicated in the lifecycle in Figure 3.5.

3.5.2.2 Analytical Tasks in Accumulated Data Streams

As aforementioned in Section 3.3, streaming descriptive statistics task can be implemented using frequency measurement, central tendency measurement, dispersion or variation measurement, and position measurement. We chose the first approach, which implements the analytical task using frequency measurement for the smart parking application. The aim of this task is to show how often the parking event occurs by showing the parking frequency at each *spot_id* grouped by *vehicle_id*. We also analyze the parking behavior of the driver by statistically computing the parking usage of each vehicle. At the edge, the data stream can be configured to be accumulated at different time granularity (i.e., every 10 min).

The data aggregation task is executed at the fog in order to count how many times each parking spot was occupied every hour, day or month. We have implemented a Python script to trigger the data aggregation task. For example, after each hour, a set of individual summaries $\{Q_1, Q_2, \ldots, Q_k\}$ will be produced in which each Q contains 4 main attributes including {*spot_id, lat, long, parking_frequency*}. The aggregated data of this task are pushed to the data clustering task for further analytics.

The aim of the data clustering task is to demonstrate how it is possible to diagnose if an incident or event occur at the fog in near real time manner. To detect an occurrence, we build an algorithm based on the Hierarchical Agglomerate Clustering (HAC) (Müllner, 2011) approach to cluster the temporal dimensions from the incoming aggregated data. We choose to implement this unsupervised learning method at the fog because it can work independently and automatically without any

human interference. The HAC method starts by partitioning a chunk of the data stream and place each data tuple into its own singleton cluster. Then, it merges the current pair of mutually closest clusters to form a new cluster. Finally, it repeats step by step until there is one final cluster left, which comprises the entire chunk of data stream.

The input of our clustering algorithm is a set of aggregated data tuples in which each data point contains 4 features {*spot_id, lat, long, parking_frequency*}. The aggregated data tuples are continuously pushing to the fog every hour. At the fog, we configure a user-defined window *weekly*. At the end of each time window, we trigger a data restructure function to sort the data so that each parking spot has not only its geo-information but also its parking frequency information at each hour during a week time window. Then, we apply the Principal Component Analysis (PCA) to select the best attributes to feed the clustering algorithm. The clustering algorithm is executed as shown in Algorithm 3.

There are many criteria to measure the distance between two clusters, u and v, such as single linkage, complete linkage, average linkage, weighted linkage, centroid linkage or median linkage. In our algorithm, we use Ward linkage since it can efficiently handle noise. In this case, the distance between two clusters is measured as the following equation.

$$d(u,v) = \sqrt{\frac{(n_u + n_s)d(u,s) + (n_v + n_s)d(v,s) - n_s d(u,v)}{n_u + n_s + n_v}} \tag{3.3}$$

where u, v are two joined cluster, and s is any other cluster; n_u, n_v and n_s are the size of cluster u, v, s, respectively.

Algorithm 3: Data clustering implementation for aggregated data based on Agglomerate Hierarchical Clustering approach

Data: Set of $Q = (Q_1, Q_2, Q_3, ...)$ such that $Q_i = (q_1, q_2, q_3, q_4)$ corresponding with {*spot_id, lat, long, parking_frequency*} is the aggregated data tuple; A distance function *DIST(u,v)*

Result: Set of clusters Y

Function `Data_Restructure(`U`):`
- $P \leftarrow Select_{distinct}(U)^{[id,lat,long]}$ `// Select a distinct list of parking spot`
- **foreach** $Hour$ **do**
  - $X \leftarrow Select(U)^{[parking\_frequency]}$ `// Select a list of parking frequency for every 1 h interval`
  - $G = P \bowtie X$ using SpotID $\langle Id \rangle$ `// Left outer join the parking frequency column every 1 h to the list of parking spot`
- **end**

Function `Clustering(`Q`):`
- $C \leftarrow \emptyset$ `// Initialize the set of clusters`
- **for** $i \leftarrow 0$ **to** n **do**
  - $C \leftarrow C \cup Q_i$ `// Place each data tuple into its own singleton cluster`
- **end**
- **while** $(|C| > 1)$ **do** `// Loop until there is 1 cluster remain`
  - **forall** $u, v \in C$ **do**
    - $d_j \leftarrow DIST(u, v)$ `// Compute the distance of all pair of clusters`
  - **end**
  - $\{c_i, c_j\} \widehat{=} min(d_j)$ `// Select a pair of clusters` $\{c_i, c_j\}$ `corresponding to the best distance` d_j
  - $C \leftarrow (C \setminus c_i) \setminus c_j$ `// Eliminate them from the active set`
  - $C \leftarrow C \cup (c_i \cup c_j)$ `// Group them to form a new cluster`
- **end**
- **return** C;

Function `Main(`Q`):`
- $W \leftarrow Begin\_Time$
- **while** *True* **do**
  - $W \leftarrow W + \Delta(t)$ `// Set a time window to process the incoming data`
  - $U \leftarrow \emptyset$
  - **repeat** `// Accumulate the incoming tuples until the end of each time window`
    - $U \leftarrow U \cup$ incoming aggregated tuple Q_i
  - **until** *(current_time == W)*;
  - $V \leftarrow Data\_Restructure(U)$ `// Restructure the data`
  - $Z \leftarrow PCA(V)$ `// Apply principal component analysis to reduce the number of attributes`
  - $Y \leftarrow Clustering(Z)$
- **end**
- **return** Y;

Recently, Gomes et al. (2017) proposed the algorithm Adaptive Random Forest (ARF) to make predictions on data streams. In our smart parking application, we have implemented our data prediction task for continuous incoming data tuples in the cloud based on this ARF algorithm. According to the data life-cycle in Figure 3.5, the contextualized data streams created by the data contextualization task will become the input data for the data prediction task. From the contextualized data stream, we receive a sequence of contextualized tuples $\{T'_1, \ldots, T'_n\}$ pushing from the fog in which each $T'_i = \langle s_1, \ldots, s_{11} \rangle | i$ corresponding with $\{$*spot_id, length, startTime, vehicle_id, lat, long, spot_name, event, endTime, edge_arrivingTime, fog_arrivingTime*$\}$. For each tuple, we use the attribute $\boldsymbol{event} = \{Occupied \mid Empty\}$ as the corresponding predictive target label when it is inputted to the ARF algorithm. It is worth noting that the ARF algorithm works based on the assumption that the tuples of input data stream are independent and identically distributed (iid). In our contextualized data stream, each data tuple T'_i is individualistic and it does not influenced to or is influenced by tuple T'_{i+1}. Also, the data contextualization task have deduce the *event* when each tuple arrive at the fog node. Therefore, the ground truth target label $T'_i\langle event \rangle$ corresponding with the other attributes in tuple T'_i is always available before the next tuple T'_{i+1} is presented to the learning algorithm.

Algorithm 4 illustrate the procedure to implement the ARF algorithm in the cloud to predict the event from the incoming contextualized data stream. Different from batch random forest algorithm, where all data instances are available for training; in stream learning, training is performed incrementally as new data tuple T'_i is available. In the process of growing trees over the current incoming data tuple T'_i, Algorithm 4 is able to detect whenever a concept drift happen in a tree and start to replace by its respective background tree. Performance P of the ARF model is computed according to some loss function that evaluates the difference between the set of expected target labels $T'_i\langle event \rangle$ and the predicted ones $\widehat{T}'_i\langle event \rangle$.

Algorithm 4: Data prediction implementation using the Adaptive Random Forest over the contextualized data stream in the cloud

Data: Set of $T' = (T'_1, T'_2, ..)$ such that $T'_i = (s_1, \ldots, s_{10}, event)$ is the incoming contextualized data tuples;

Result: Prediction model P

Initialize: Set number of small tree $N = int\_value$; Randomly set number of attributes $F = Random\_number(2 : max\_no\_feature(T'))$;

Function `Tree_Grow(`*Tree, T', F*`):`

  $k \leftarrow Poisson(\lambda)$ `// Set` λ `value for bagging according to Poisson distribution`

  if *(k>0)* **then**

    $L \leftarrow FindLeaf(Tree, T')$

    $UpdateLeafCounts(L, T')$ `// update counts on attribute values at L using T'` **if** $(TuplesSeen(L) >= Leaf_{min})$ **then** `// Minimum number of tuples a leaf`

      $Split\_on\_Best\_Attribute(L, F)$

    end

  end

Function `Random_Forest(`*N, F,* T'_i`):`

  $W \leftarrow InitWeights(N)$ `// Initialize the weight for N trees`

  $Tree \leftarrow CreateTrees(N)$ `// Create N trees`

  $P \leftarrow \emptyset$

  while T'_i *in pipeline* **do**

    forall $Tree_k \subseteq Forest(N)$ **do**

      $\widehat{T'_i}\langle event\rangle \leftarrow predict(T'_i)$ `// Predict a parking spot (Empty/Occupied) based on incoming tuple` T'_i

      $W(Tree_k) \leftarrow P(W(Tree_k), \widehat{T'_i}\langle event\rangle, T'_i\langle event\rangle)$ `// Assign new weight to` $Tree_k$

      $p_k \leftarrow Tree\_Grow(Tree_k, T'_i, F)$ `// Grow` $Tree_k$ `on current incoming tuple` T'_i

      $P \leftarrow P \cup p_k$

      if $(Detect(concept\_drift) == True)$ **then** `// Concept drift detected!!!`

        $p_j \leftarrow Tree\_Grow(Tree_j, T'_i, F)$ `// Replace tree by its new base tree` $Tree_j$

        $P \leftarrow P \cup p_j$

      end

    end

  end

  return P;

3.5.3 Streaming IoT Messages

As aforementioned in Section 3.4, the middleware brokers are integrated into our architecture to assist us into stream the incoming data seamlessly in our system. This section aims to illustrate the details of the data streaming mechanism between the edge, the fog, and the cloud nodes in our edge-fog-cloud continuum via the AMQP protocol. This protocol allows conforming client applications at different nodes in the network of resource to communicate with each other via conforming message brokers. A node in the network of resource can play the role of a producer or a consumer. A producer is the application which can broadcast the messages to a message exchange of a broker, while a consumer is the application which can retrieve messages from the message queue. The data stream is not transmitted directly to a message queue, instead, the producer streams data to an exchange. At a broker, an exchange will route the data stream to the different queues.

Figure 3.7 delineates a sequence diagram of transmitting the IoT data stream from the devices to the edge, then from the edge to the fog, and from the fog to the cloud using this protocol. First, the IoT devices connect to the first message broker and publish their generated data to the *(/raw_data)* topic. Consumer applications at the edge will connect to the same message broker and subscribe the *(/raw_data)* topic to ingest the data. At the edge, different analytical tasks can be executed before a producer application communicate with the second message broker and publish the processed data to the *(/contextualized_data)* topic. The same process happening at the fog as the consumer applications will connect to the second message broker and receive the data from the *(/contextualized_data)* topic. Again, analytical tasks are executed to diagnose the event from the data in near real time. Finally, a producer application establish a new connection with the third message broker to transmit data to the cloud. In the cloud, the consumer applications

will communicate with the third message broker and retrieve data from its message queues for the predictive analytical task. Algorithm 5 depicts the sample pseudo code for the producer and consumer to exchange the data via the brokers (Watch the demo of streaming data from the edge to the fog, then to the cloud here: `https://www.youtube.com/playlist?list=PL-hcE-LoSl0uMQy12yanDS8MEl5QLwp3d`).

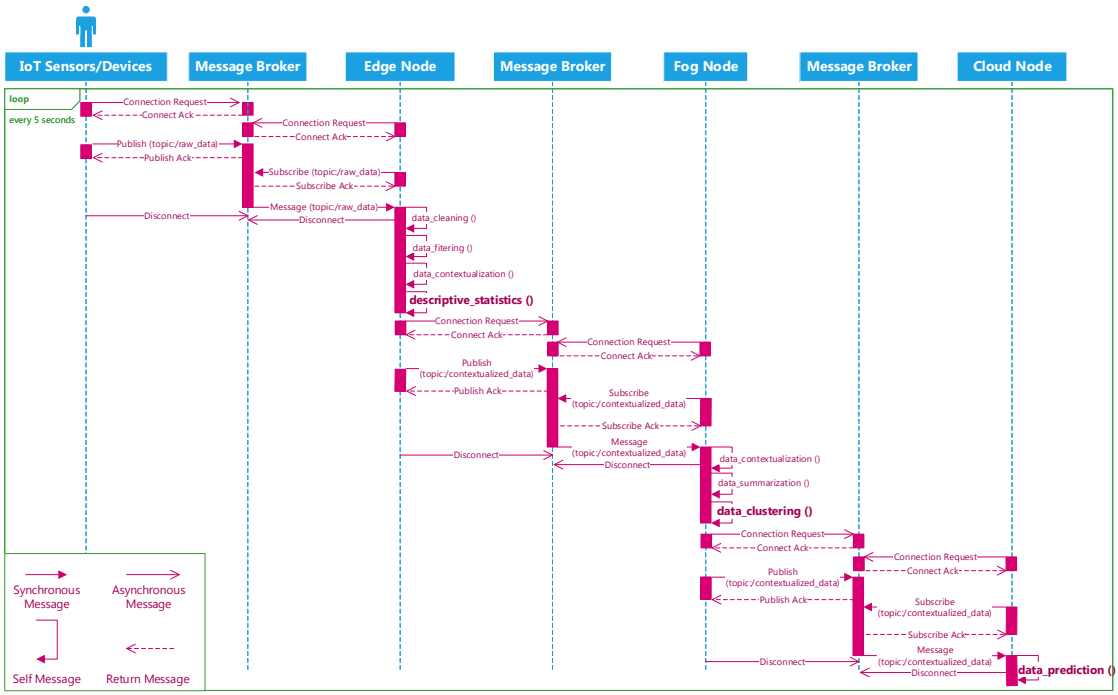


Figure 3.7: Sequence diagram for pushing the IoT data stream to the edge, fog, and cloud using Advanced Message Queuing Protocol (AMQP) protocol.

3.6 Discussion of the Results

This section describes the outcomes of the data life-cycle of our proposed *"Analytics Everywhere"* framework by showing examples of the results that emerged from our streaming descriptive, diagnostic and predictive analytics for the smart parking scenario. First, we discuss the performance of the our proposed architecture based on the latency of the data stream and the memory consumption metric. Second, we

Algorithm 5: Sample code to stream data from Producer to Consumer

```
1  Initialize: server = "hostname"
2              port = 5672
3              vhost = "virtualhost"
4              username = "username"
5              password = "password"
6              exchangeName = "Exchange"
7              queueName = "Queue"
8  Function Producer():
9      while True do
10         connection = create_connection(server, port, vhost, username,
             password)
11         channel = connection.create_channel()
12         channel.send_message(exchangeName, data)
13         connection.close()
14         delay(5 s)
15     end
16
17 Function Consumer():
18     while True do
19         connection = create_connection(server, port, vhost, username,
             password)
20         channel = connection.connect_channel(exchangeName)
21         channel.consume(queueName, data)
22         connection.close()
23         delay(5 s)
24     end
```

explore insights from the analytics at the edge, the fog, and the cloud.

3.6.1 Architecture Evaluation

In order to evaluate our proposed architecture, we have monitored the latency of the data streams when they arrived at our compute nodes. To compute the latency metric, we have collected samples every 10 min and registered the arrival times of the data streams at the edge, the fog, and the cloud. Figure 3.8 illustrates the patterns of the arrival time at different compute nodes.

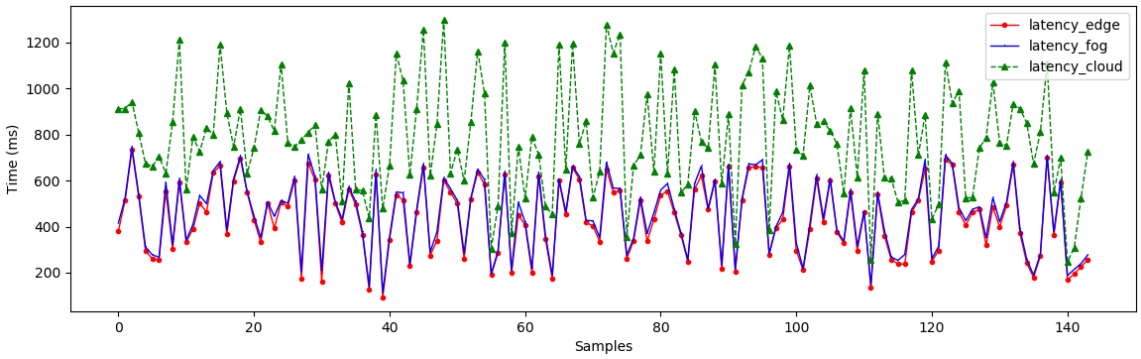


Figure 3.8: Latency Patterns.

As we can see, the latency at the edge and the fog are not significantly different. In contrast, there is a significant difference between them and the latency in the cloud. In fact, the latency at the edge and fog fluctuated around 150 $\rightarrow$ 800 (ms), while the latency in the cloud ranged from 200 $\rightarrow$ 1300 (ms). Although we can see similar latency patterns, a delay is clearly apparent when the data streams arrive in the cloud. This can be explained because we have deployed the edge and the fog nodes geographically close to each other using WSN in our smart parking scenario. But the data is streamed to the cloud later using the core network. These latency outcomes in Figure 3.8 have provided us with new insight on the crucial role of *a priori* mapping between analytical tasks with the appropriate resource capabilities.

Aiming to test the ability of our proposed IoT architecture to handle the streaming traffic going through different hops in our architecture, we have computed the memory consumption details of the brokers in Figure 3.9. Note that the memory details shown here have been only updated on request because they could be too expensive to calculate every few seconds on a busy compute node. As we can see, the total amount of memory used was around 75 MB including allocated memory for queues, binaries, connections, tables, processes and system. It was accounted for approximate 76.5% of run time allocated for this broker during the last updated request. This result indicates that there is still a lot of room in our system to perform more heavier analytical tasks. It also shows the stability of our architecture during the IoT data streaming operations.

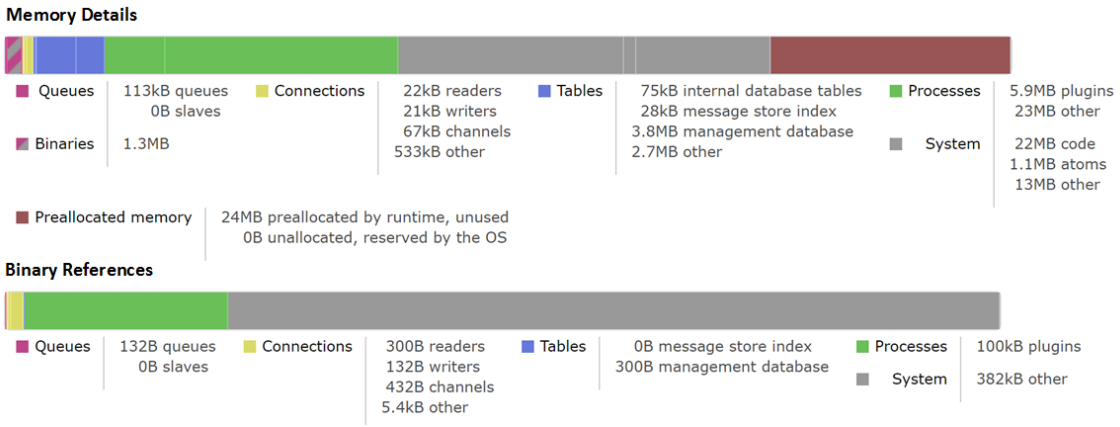


Figure 3.9: Memory Consumption Overview.

3.6.2 What Is the Problem with Smart Parking in Saint John?

In this section we describe the effectiveness of our architecture based on the proposed data life-cycle consisting of monitoring the usage patterns (i.e., counts of how many times the parking spot is occupied or occupancy frequency) at each parking spot

every 10 min using the edge nodes to continuously process the IoT data streams.

Figure 3.10 presents the usage patterns of 25 most used parking spots during a specific day of observation (13 May 2019). By comparing the total parking duration and the average parking time with the frequency of an *Occupied* event for each parking spot, we can infer that although the parking frequency is high, the average parking time at each spot is relatively low, approximately 1 h to 1.7 h. Only 2 parking spots (id 9339 & 9342) are less than 1 h. Note that if a point in Figure 3.10 is close to the origin coordinate, it signifies that the parking spot is usually used for short duration of time. However, the parking usage pattern was different the following day (14 May 2019); the average parking time increased to about 1.5 to 3.6 h (See Appendix A Figure A.1). The data visualization analysis during a week of observation can be found in this link (`https://youtu.be/YwlOWXK9F3I`).

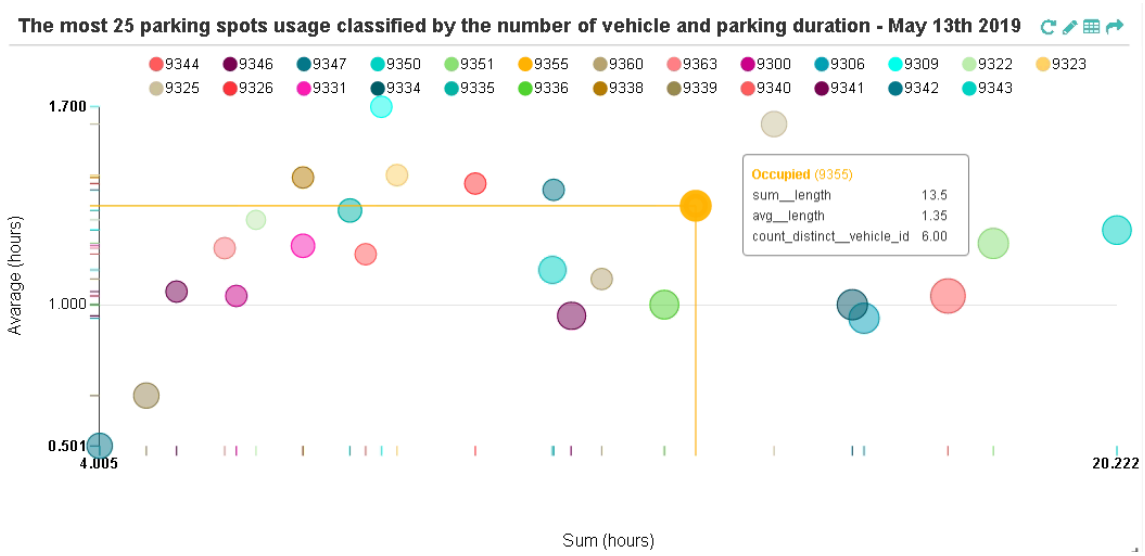


Figure 3.10: Parking usage pattern of the 25 most used parking spots.

Figure 3.11 shows the statistical information about the total parking hours of the top 50 vehicles using the parking service in the city during 2 weeks of observations (13 May 2019 to 26 May 2019).

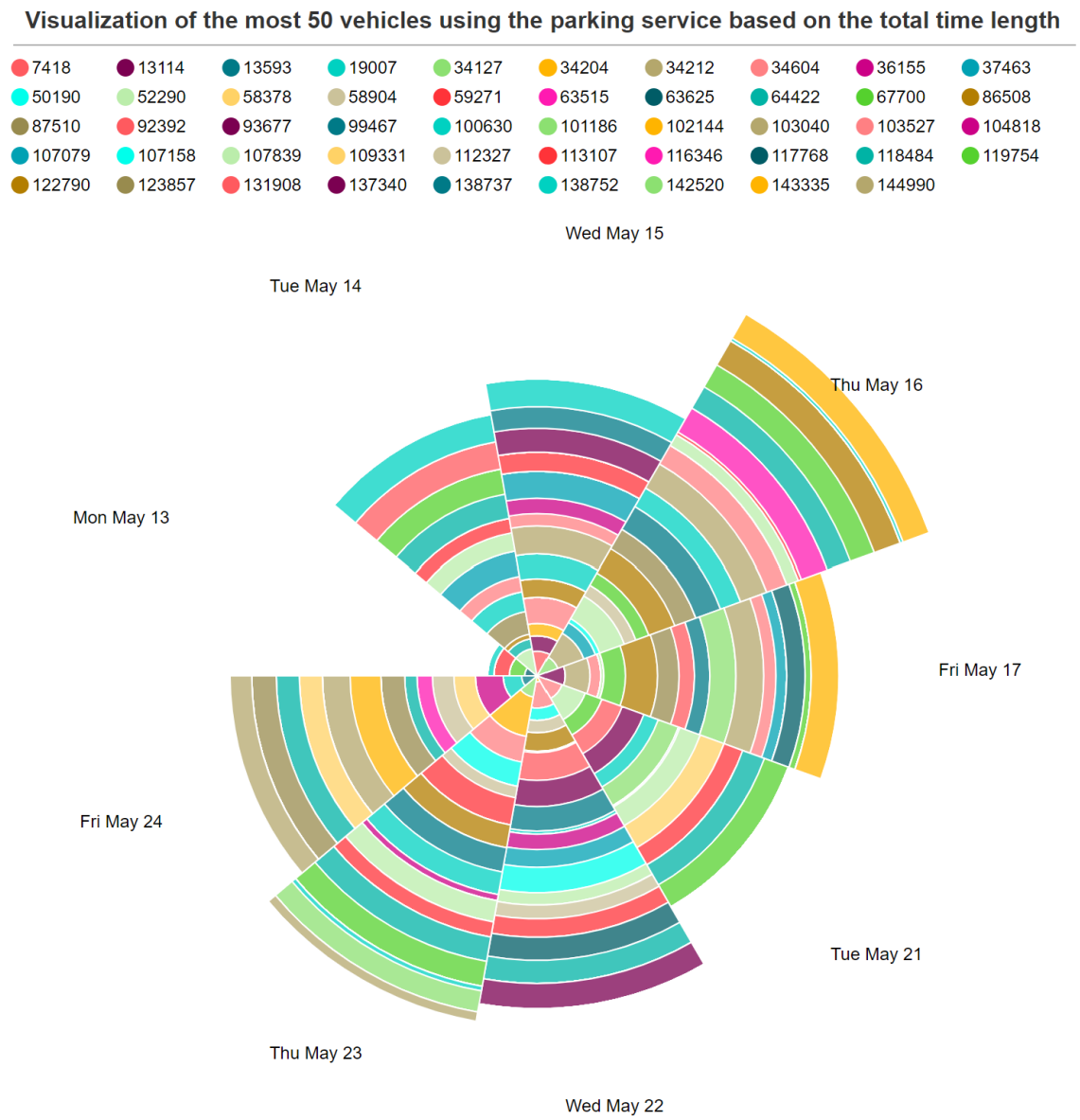


Figure 3.11: Usage patterns of the top 50 vehicles.

These preliminary results are already point out the under-utilization of parking spots in the city, since the frequency patterns can show often and how long the parking sports are being used.

3.6.3 Why Are These Usage Patterns an Issue in Saint John?

At the end of the data clustering task in Section 3.5.2.2, we are able to analyze to diagnose the event/incident from the clustering algorithm results. In this smart parking application, we have observed the continuous incoming aggregated data at the fog for about 5 weeks (from 13 May 2019 to 16 June 2019). Whenever an aggregated data tuple arrives at the fog, it will be accumulated and analyzed weekly. Figure 3.12 illustrates the dendrogram of the first observation week. The dendrogram is a tree-form visualization of the clustering algorithm showing the order and distances of merged clusters. One advancement of the HAC is that we do not need to choose a number of clusters k in advance. Instead, we can determine the number of clusters after the algorithm has been executed based on a *cut-off* distance of the dendrogram. As can be seen clearly in Figure 3.12, four main groups of instances are congregated into the clusters. Therefore, we have configured the cut-off distance equal to 19 (the black horizontal line in Figure 3.12). The dendrograms of the remaining observation weeks are depicted in Appendix B.1 (Figure B.1).

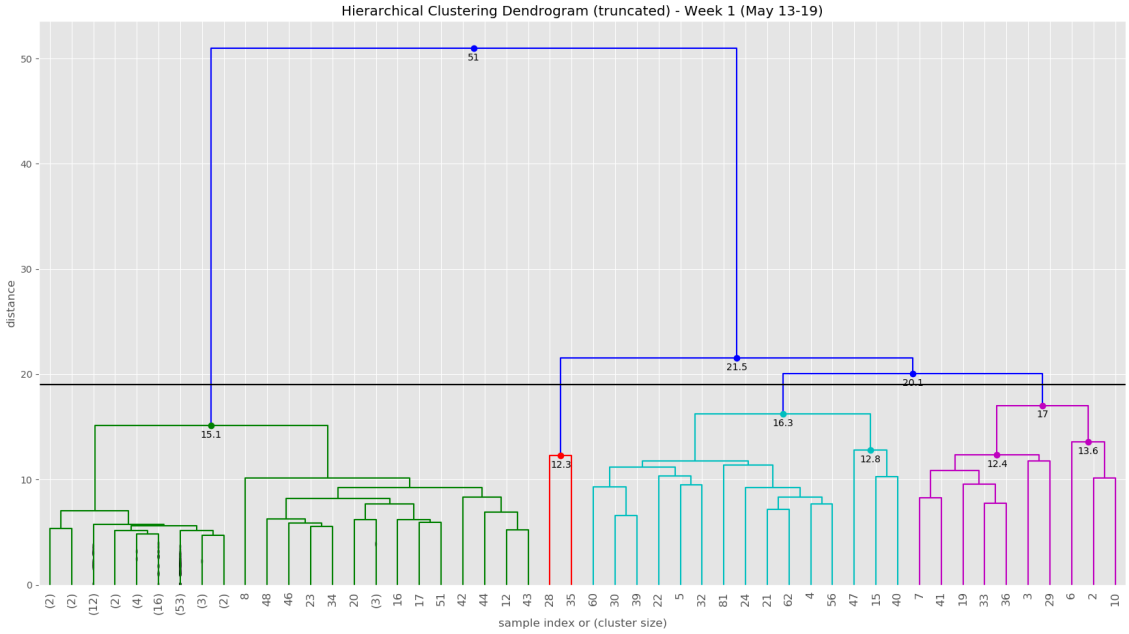


Figure 3.12: The dendrogram of the first observation week (13 May–19 May).

Figure 3.13 show the clustering results of the temporal attributes during the first observation week. The clustering results of the remaining observation weeks are depicted in Appendix B.2 (Figures B.2–B.5). The top right sub figure of Figure 3.13 of illustrates the 4 main temporal clusters found by the HAC algorithm using a Ward distance and cut-off distance equal to 19, while the top left sub figure represents the geographical location of the parking spots for the corresponding clusters. Similarly, the bottom sub figures show the resulting temporal clusters and corresponding parking locations using the cut-off distance set equal to 16.

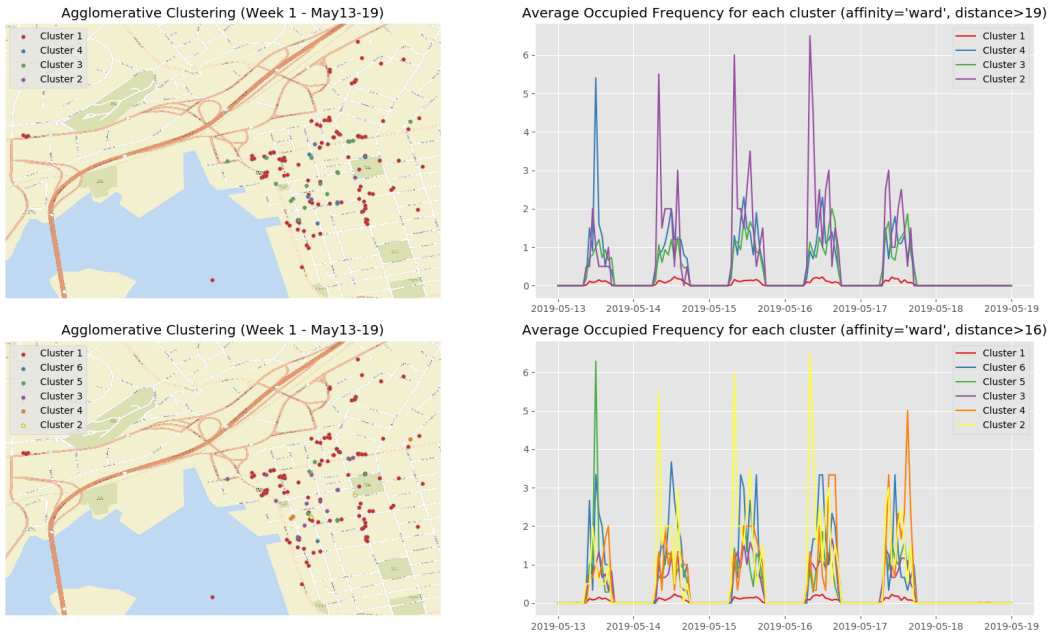


Figure 3.13: Clustering result of the first observation week (13 May–19 May).

We used aggregated data tuples arriving at the fog hourly as the algorithm input attributes. Therefore, we have $7\ days \times 24\ h \times 1\ tuple = 168\ attributes$ for each parking spot in the city during a week of accumulating aggregated data. Hence, we would like to perform dimension reduction using PCA on these attributes to see if any improvement on the clustering result with lower dimensionality. Figure 3.14 delineates the singular values of the principle components and the variance explained with these components. From this figure, we can see that the first principle compo-

nent can explain approximately 60% of the data variance, and the top 5 principles together can explains nearly 80% of the data variance. Thus, we keep the first 5 components to re-implement the HAC algorithm again.

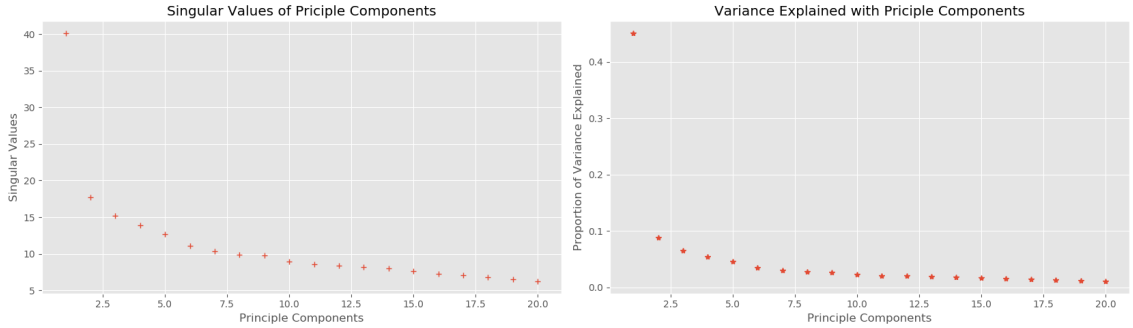


Figure 3.14: Principle Component Analysis over the aggregated data.

Figure 3.15 shows the dendrogram of the first observation week when we apply the HAC algorithm on the first 5 principle components. Although there is a slight difference in the leaf nodes compared to Figure 3.12, the group of clusters are very similar.

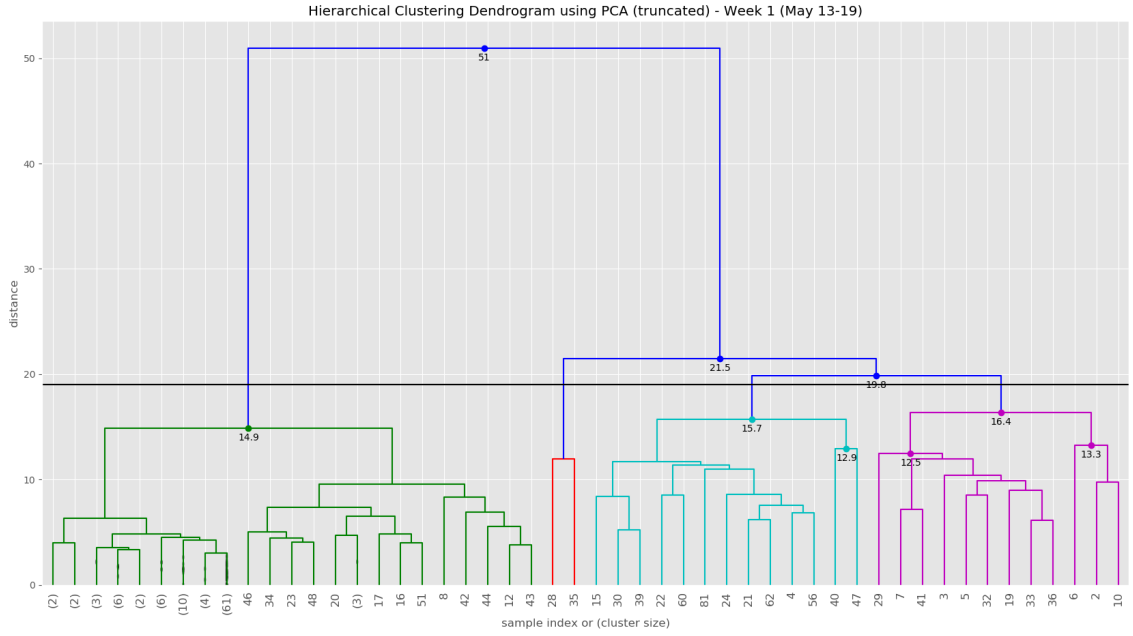


Figure 3.15: The dendrogram of the first observation week for the top 5 principle components.

Figure 3.16 shows the clustering results of the first observation week when we apply the HAC algorithm on the top 5 principle components. The clustering results on the top 5 principle components of the rest observation weeks are depicted in Appendix B.3 (Figure B.6–B.9) (See the evolution of the clusters here (1) `https://youtu.be/XDUtc9XJWc8` (2) `https://youtu.be/iS6-WxYHkv8`).

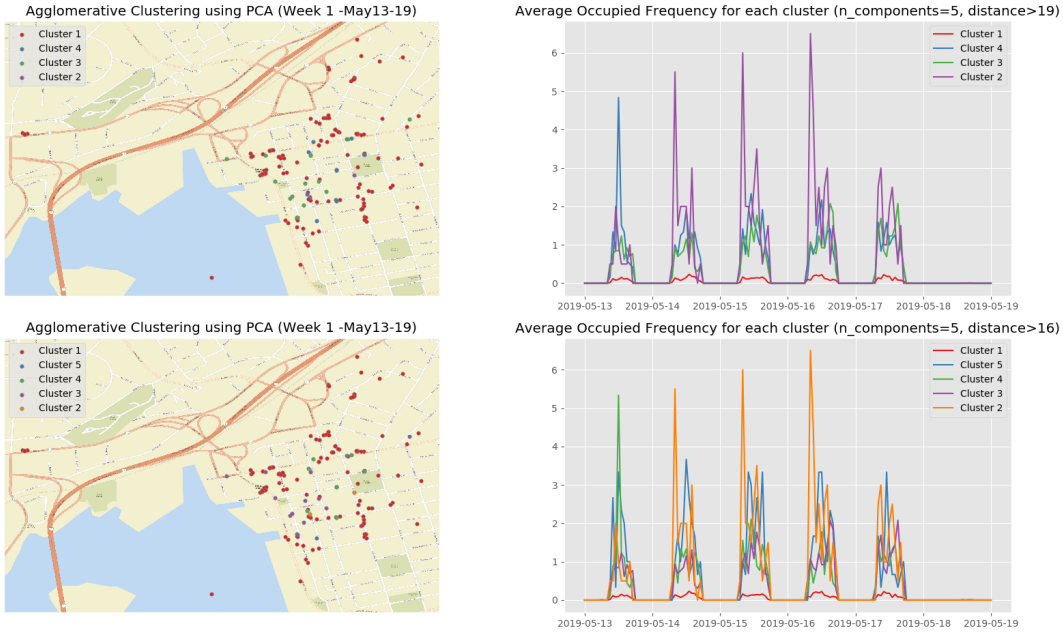


Figure 3.16: Temporal patterns of occupied/empty events that were computed at the fog node.

Observations & Comparisons: From the clustering results (see Figures 3.16 and B.6–B.9), we are able to diagnose some events/incidents based on the following observations and comparisons:

- We can clearly discover 3 types of parking spots and where they are:
  - the busiest parking spots (Cluster 2 in Figure 3.16, Cluster 3 in Figure B.6, Clusters 4 and 5 in Figure B.7, Cluster 2 in Figure B.8, and Cluster 2 in Figure B.9) with the parking frequency from 2 to 4 times per hour, which are all in the downtown core of Saint John City.

  - the ordinary parking spots (Clusters 3 and 4 in Figure 3.16, Cluster 2 in Figure B.6, Clusters 2 and 3 in Figure B.7, Cluster 3 in Figure B.8, and Cluster 3 in Figure B.9) with the parking frequency from 1 to 2 times per hour, which are mainly in the downtown core of Saint John City and surrounding areas.
  - the unpopular parking spots (Cluster 1 in Figure 3.16, Cluster 1 in Figure B.6, Cluster 1 in Figure B.7, Cluster 1 in Figure B.8, and Cluster 1 in Figure B.9) with the parking frequency from 0 to 1 time per hour and are often located in the areas far from the the downtown core of Saint John City.

- Based on these diagnostic analytical results, we can observe that there are almost no parking event during the weekend. However, this may, in fact, be inaccurate because the smart parking facilities are free for usage during the weekend.

- Based on the clustering results, we identified that a special event/festival had taken place during second observation week. Figure B.6 shows that there are not many parking behaviors on Monday, 20 May 2019. We noted that this date was a Canadian holiday, Victoria Day. However, interesting insights can be discovered from a cluster since a small number of people still paid for the parking facilities even though it was free on this day.

- Comparing the clustering results of the third observation week (Figure B.7) with the other observation weeks, we discovered that an abnormal event that occurred on Wednesday, 29 May 2019 since the parking frequency reached a peek in Cluster 5. More context (e.g., a city events/festival schedule) is needed in order to explain this phenomenon.

In summary, the clustering results alone were inconclusive to identify the reasons for the under utilization of parking spots in the city. Other factors may have played a role in generating the observed clustering patterns such as parking costs and/or availability.

3.6.4 What Could Be Improved in the Future?

The incremental predictive learning model implemented in Section 3.5.2.2 aims to anticipate whether the status of a parking spot is *Empty* or *Occupied* in the future by training the incoming contextualized data tuples. We evaluate our model mainly based on the accuracy metric and the kappa metric. Although the accuracy metric is useful on a binary classification, it does not provide a complete picture of the performance of our predictive model. Our training contextualized data tuples contain imbalanced number of *Occupied* and *Empty* class, therefore, the kappa metric (Bifet et al., 2015) is utilized alongside with the accuracy metric to avoid misleading predictive performance results. This is defined as the following equation:

$$\kappa = \frac{p_O - p_E}{1 - p_E} \tag{3.4}$$

where p_O is the predictive model's prequential accuracy and p_E is the probability of an expected random chance accuracy.

Figure 3.17 delineates the accuracy and kappa performance of our predictive model during the 5 weeks of incremental training (from 13 May 2019 to 16 June 2019) (See the full process of incremental training model (1) `https://youtu.be/AJvxM69AFps` (2) `https://youtu.be/RQEaoF4WkXo`. As can be seen from this figure, more accurate prediction results can be achieved by increasing the number of data tuples used produce the predictive model. Moreover, the kappa score was increased

to 0.8, indicating that our predictive model was improved, compared to a random chance model.

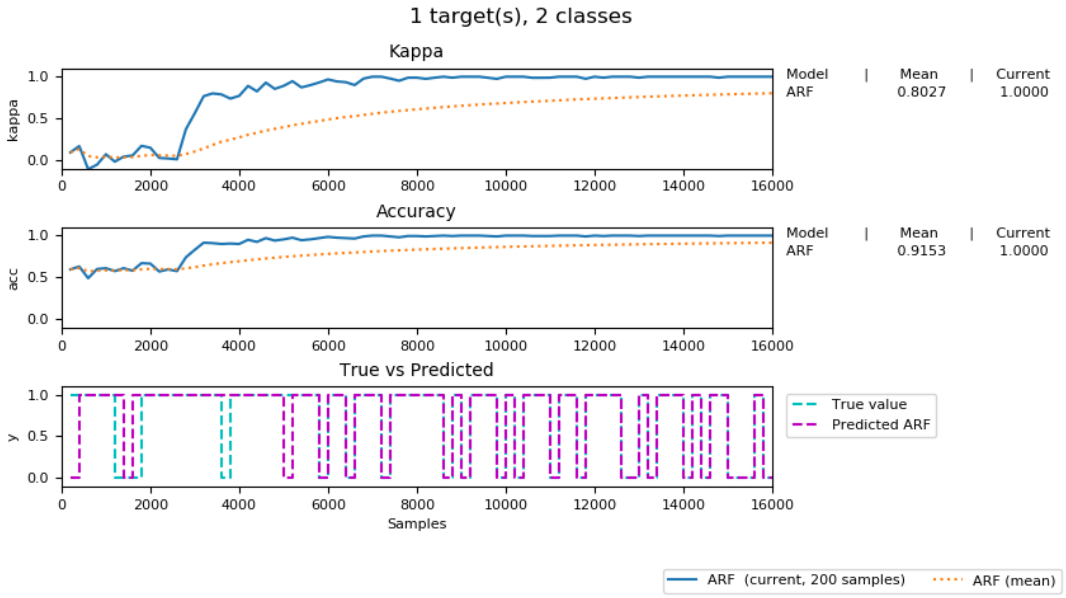


Figure 3.17: Incremental predictive learning results.

Figure 3.18 shows the F1, Precision, and Recall score of our adaptive predicting model. It shows the high score of these three measurements and an increasing trend, similar to the one shown in Figure 3.17.

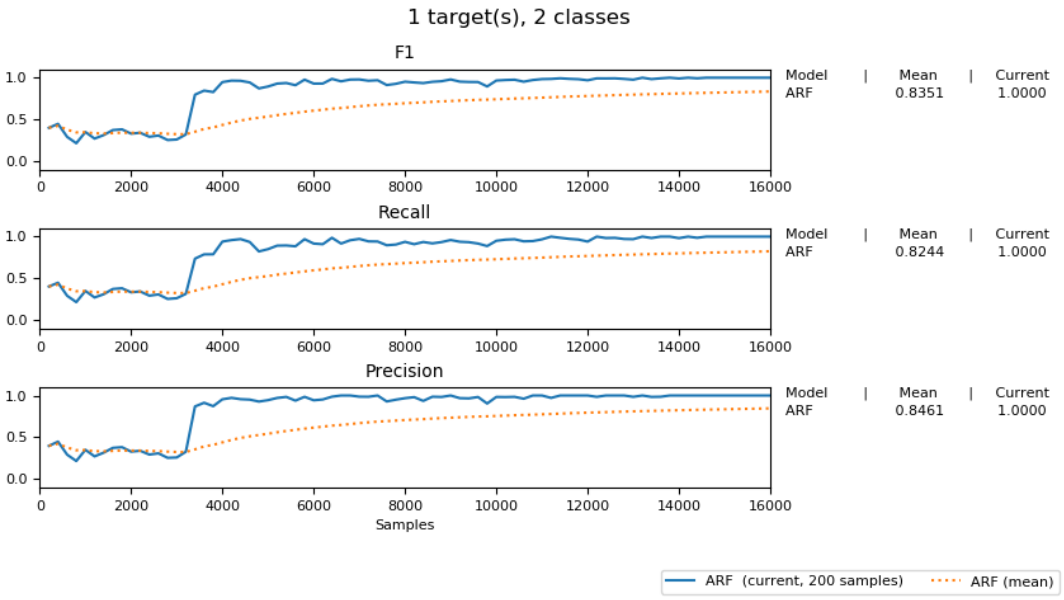


Figure 3.18: F1, Precision and Recall score of our predictive model.

In general, we have built a fairly good prediction model that is able to anticipate the incoming IoT data stream. The more data to arrive in our system, the better prediction accuracy we can achieve.

3.7 Conclusions

This paper describes our preliminary results in evaluating an IoT architecture where edge, fog, and cloud resources are used to support streaming analytics for a smart parking application. The latency and memory consumption metrics have pointed out that more research is needed to develop new metrics to evaluate IoT architectures in the future. These metrics are fundamental to design the best IoT architecture in order to account for the specific requirements of the IoT applications.

Moreover, the *"Analytics Everywhere"* framework will also play an important role in generating better results for the IoT applications. The selection of an-

alytical tasks (e.g., clustering versus classification) and performance metrics (e.g., latency) still need to be further investigated to provide more empirical results that can be used to improve our architecture.

We do not expect that one IoT architecture will fit all IoT applications. The smart parking scenario has proven that streaming analytics will always require a-priori mapping between analytical tasks and computational resources using a data life-cycle. Our future research work will also focus on developing a transfer learning process for our *"Analytics Everywhere"* framework.

Our main lesson learned after developing our IoT architecture is that if any of the edge-fog-cloud resource is considered in isolation, it would not be able to manage the data life-cycles of IoT applications without compromising the functionality or performance. However, many threats to validity of the proposed architecture using the edge-fog-computing might arise in other IoT applications. We will work with other IoT applications in smart cities, specifically those that require the analysis of time-series data and not only events.

References

Bifet, A., de Francisci Morales, G., Read, J., Holmes, G., and Pfahringer, B. (2015). Efficient online evaluation of big data stream classifiers. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 59–68. ACM.

Bifet, A., Holmes, G., Pfahringer, B., Read, J., Kranen, P., Kremer, H., Jansen, T., and Seidl, T. (2011). Moa: a real-time analytics open source framework. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 617–620. Springer.

Cao, H. and Wachowicz, M. (2017). The design of a streaming analytical workflow for processing massive transit feeds. In *The 2nd International Symposium on Spatiotemporal Computing*.

Cao, H. and Wachowicz, M. (2019). The design of an iot-gis platform for performing automated analytical tasks. *Computers, Environment and Urban Systems*, 74:23–40.

Cao, H., Wachowicz, M., Renso, C., and Carlini, E. (2019). Analytics everywhere: generating insights from the internet of things. *IEEE Access*, 7:71749–71769.

Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., and Tzoumas, K. (2015). Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 36(4).

De Francisci Morales, G., Bifet, A., Khan, L., Gama, J., and Fan, W. (2016). Iot big data stream mining. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2119–2120. ACM.

Dittrich, J. and Quiané-Ruiz, J.-A. (2012). Efficient big data processing in hadoop mapreduce. *Proceedings of the VLDB Endowment*, 5(12):2014–2015.

Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., Holmes, G., and Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9-10):1469–1495.

Ismail, W. N., Hassan, M. M., and Alsalamah, H. A. (2018). Mining of productive periodic-frequent patterns for iot data analytics. *Future Generation Computer Systems*, 88:512–523.

Kreps, J. (2014). Questioning the lambda architecture. *Online article, July*.

Lin, J. (2017). The lambda and the kappa. *IEEE Internet Computing*, 21(5):60–66.

Marz, N. and Warren, J. (2015). *Big Data: Principles and best practices of scalable real-time data systems*. New York; Manning Publications Co.

Montiel, J., Read, J., Bifet, A., and Abdessalem, T. (2018). Scikit-multiflow: a multi-output streaming framework. *The Journal of Machine Learning Research*, 19(1):2915–2914.

Morales, G. D. F. and Bifet, A. (2015). Samoa: scalable advanced massive online analysis. *Journal of Machine Learning Research*, 16(1):149–153.

Müllner, D. (2011). Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*.

Noghabi, S. A., Paramasivam, K., Pan, Y., Ramesh, N., Bringhurst, J., Gupta, I., and Campbell, R. H. (2017). Samza: stateful scalable stream processing at linkedin. *Proceedings of the VLDB Endowment*, 10(12):1634–1645.

Santos, G. L., Endo, P. T., da Silva Lisboa, M. F. F., da Silva, L. G. F., Sadok, D., Kelner, J., Lynn, T., et al. (2018). Analyzing the availability and performance of an e-health system integrated with edge, fog and cloud infrastructures. *Journal of Cloud Computing*, 7(1):16.

Ta-Shma, P., Akbar, A., Gerson-Golan, G., Hadash, G., Carrez, F., and Moessner, K. (2018). An ingestion and analytics architecture for iot applied to smart city use cases. *IEEE Internet of Things Journal*, 5(2):765–774.

Toshniwal, A., Taneja, S., Shukla, A., Ramasamy, K., Patel, J. M., Kulkarni, S., Jackson, J., Gade, K., Fu, M., Donham, J., et al. (2014). Storm@ twitter. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 147–156. ACM.

Wingerath, W., Gessert, F., Friedrich, S., and Ritter, N. (2016). Real-time stream processing for big data. *it-Information Technology*, 58(4):186–194.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., et al. (2016). Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65.

Chapter 4

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

This chapter has been published in the Computers, Environment and Urban Systems Journal. The full citation of this published article is:

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.

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.

4.1 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 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 4.2, related works in GIS platforms previously developed for smart transit applications are reviewed and the existing IoT platforms are described. In Section 4.3, the IoT-GIS platform is presented, including the details of the automated tasks, specifications, and requirements. Section 4.4 is dedicated to describing the cloud architecture used to implement the IoT-GIS platform. Section 4.5 describes in detail the experiment of implementing our IoT-GIS platform for the Codiac Transit Agency. Section 4.6 discusses both the performance of the proposed platform and the experiment analysis results. Section 4.7 concludes the paper and discusses further research.

4.2 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).

4.3 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 4.1 illustrates the overview of the automated analytical tasks.

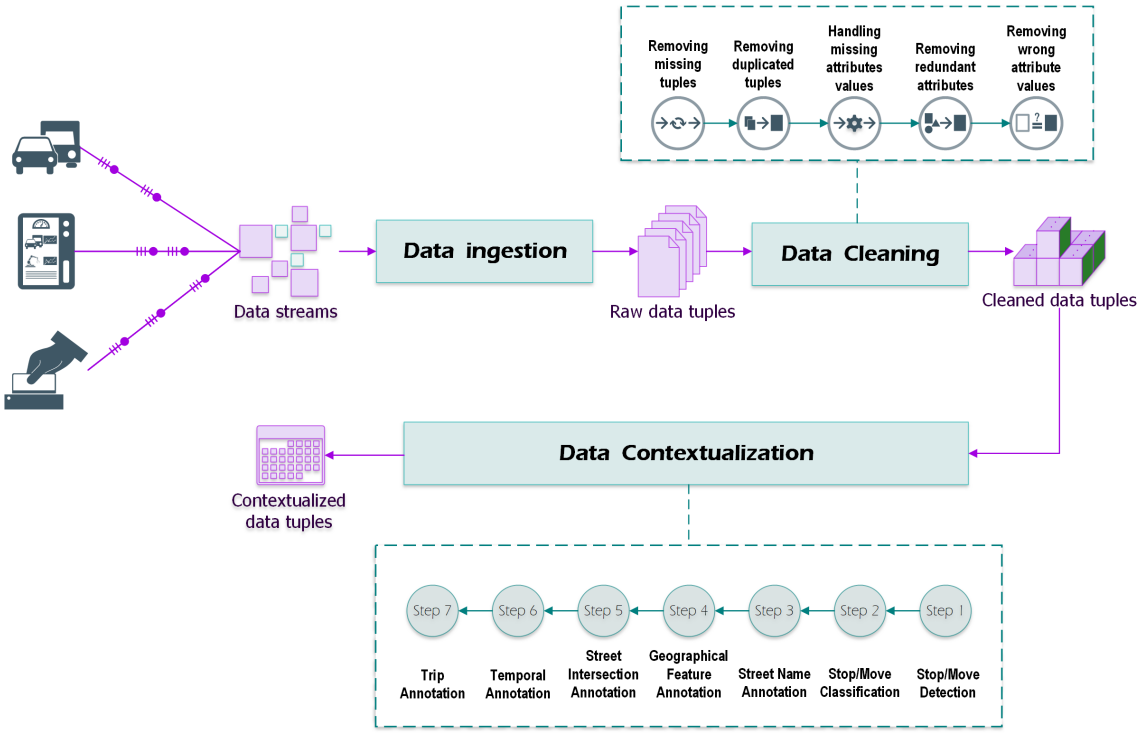


Figure 4.1: Automated tasks our IoT-GIS platform.

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.

4.3.1 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 net-

work (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.

4.3.2 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.

4.3.3 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.

4.4 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 4.2 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.

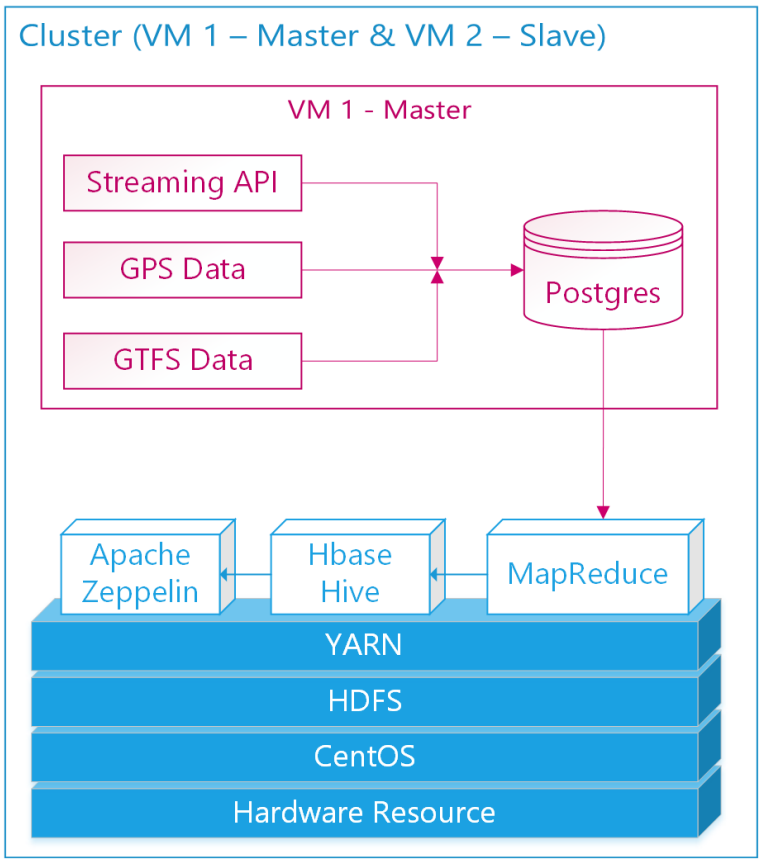


Figure 4.2: Overview of the cloud architecture developed for our IoT-GIS platform.

4.4.1 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.

4.4.2 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 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 4.1.

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 4.1: Overview of the Hadoop Specifications.

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 4.3.

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 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

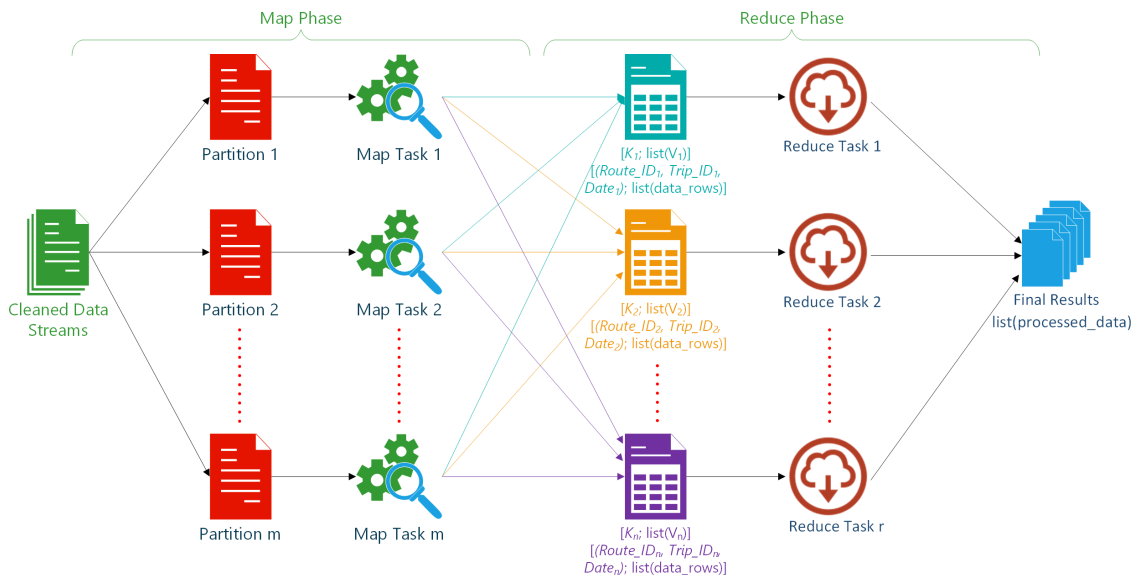


Figure 4.3: Logical view of the contextualization steps using MapReduce.

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.

4.5 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 4.4

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.

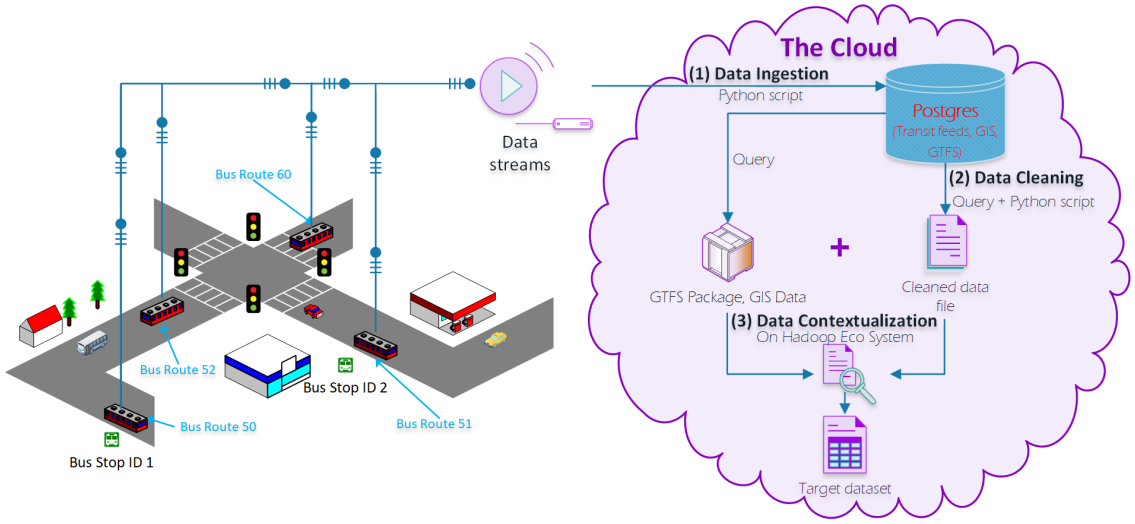


Figure 4.4: Overview of our IoT-GIS platform developed for Codiac Transit.

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 4.5.

4.5.1 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

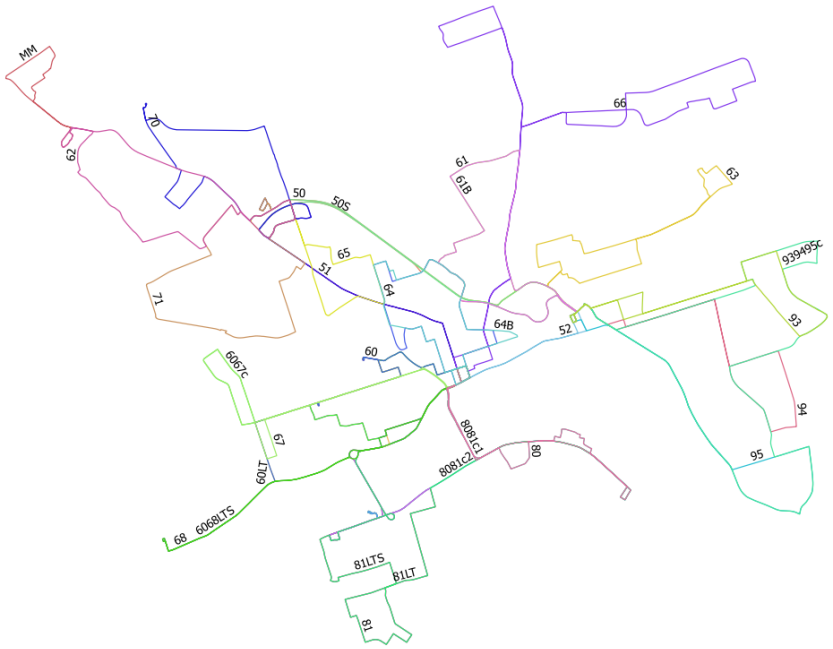


Figure 4.5: The Codiac Transit Network.

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 6 provides the pseudo code for the execution of the data ingestion task.

4.5.2 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 4.2.

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

Algorithm 6: 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.

```
1  Initializing database;
2  while True do
       // run the loop forever
3      establish connection;
4      read(G);
5      forall T_i ⊆ G do
6          if T_i valid then
7              insert(T_i) to the database;
8              print "Successful" ;
9          else
10             print "Failed" ;
11             pass;
12         end
13     end
14     delay(5); // ingest data streams every 5 seconds
15 end
```

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 4.2: SQL Statement implemented for the cleaning task.

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 7 provides the pseudo code developed to perform the data cleaning task.

Algorithm 7: 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 ⊆ G do
        extract Trip K_i from G (K_i= Set of different T_i);
        foreach K_i ⊆ 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;
```

4.5.3 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 4.3). 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 8 provides the pseudo-code developed to perform the data cleaning task.

Figure 4.6 illustrates the contextualized steps for the bus trip 51-12 of the

Algorithm 8: 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 ⊆ 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 ⊆ 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;
```

route 51 on the date 15-06-2016. This particular trip was randomly selected for explaining the outcomes of the 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

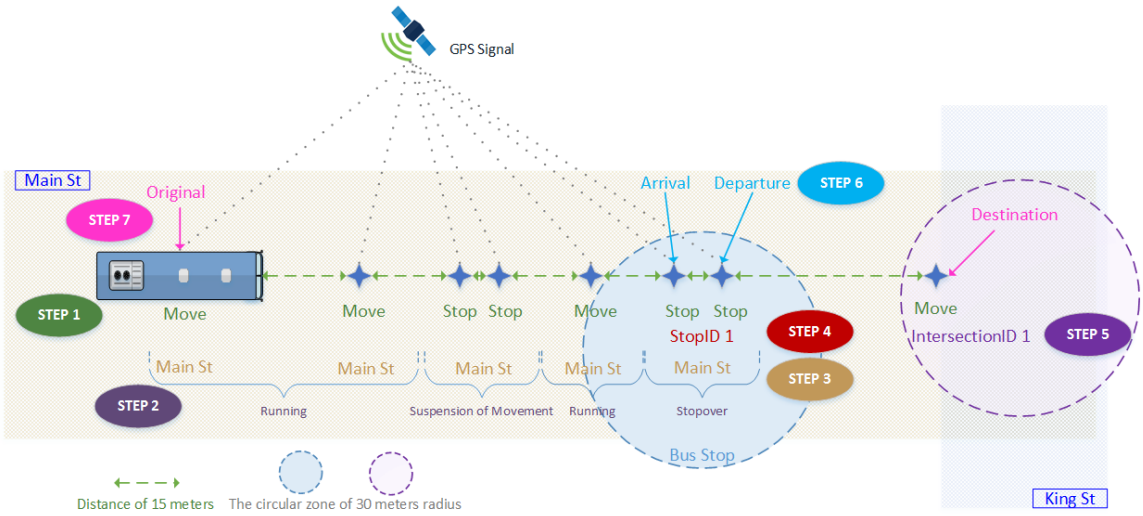


Figure 4.6: Overview of the automated steps designed for the data contextualization task.

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 4.7a 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 4.3). This information was available from the GTFS data previously stored in the PostgreSQL database.

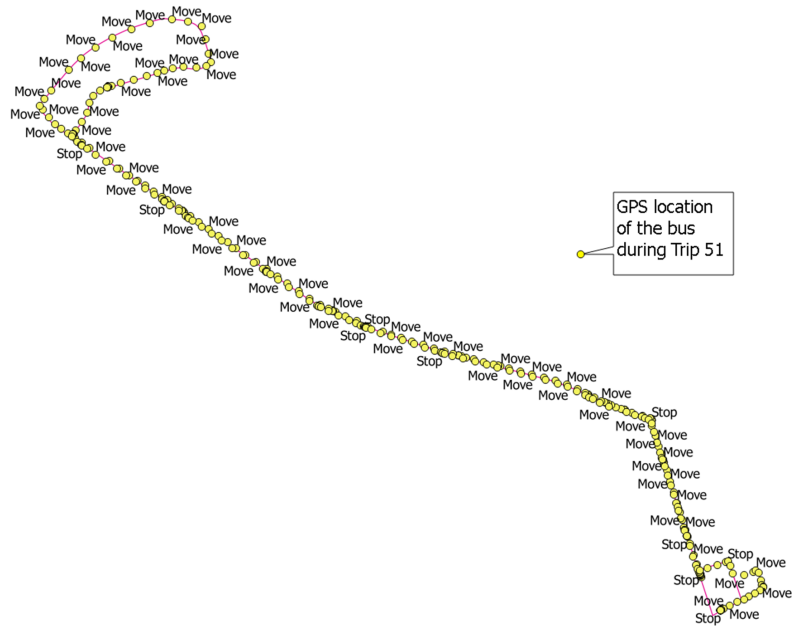
Description	SQL Statement
Get a list with all bus stations	SELECT trip_id, stop_id, stop_sequence, depart_return, change_direction, stop_lat, stop_lng FROM moncton_gtfs_dim WHERE trip_id = 'trip' ORDER BY stop_sequence ASC

Table 4.3: SQL Continuous query for the Stop/Move Classification Step.

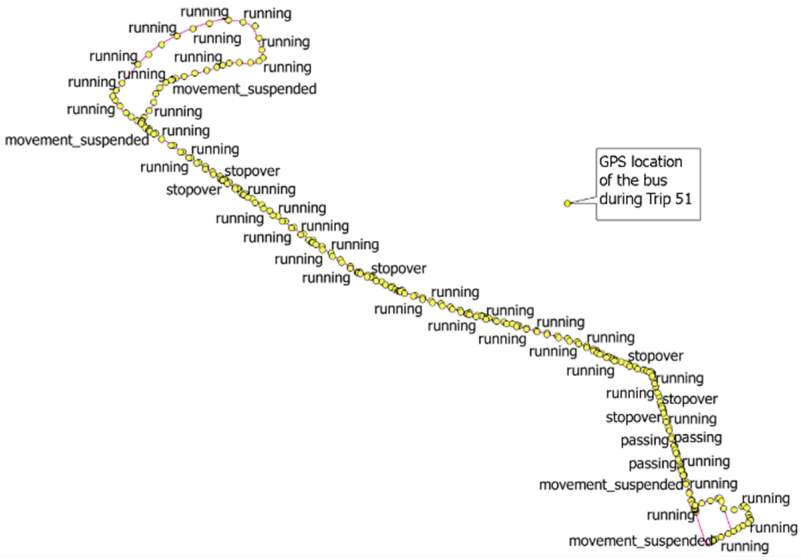
Afterwards, the algorithm created a circular zone with a radius of 30m for each bus station. The stops which are located inside the buffer are classified as *"stopovers"*; otherwise, they were classified as *"suspension of movement"*. Moreover, the moves which were located inside the buffer were classified as *"passing"*; otherwise they were classified as *"running"* 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 4.7b.

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 4.4). 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



(a) Stop/Move Detection.



(b) Stop/Move Classification.

Figure 4.7: Results for one trip of the bus route 51.

Codiac transit network was created. Table 4.5 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

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 4.4: Continuous query for the Street Name Annotation Step.

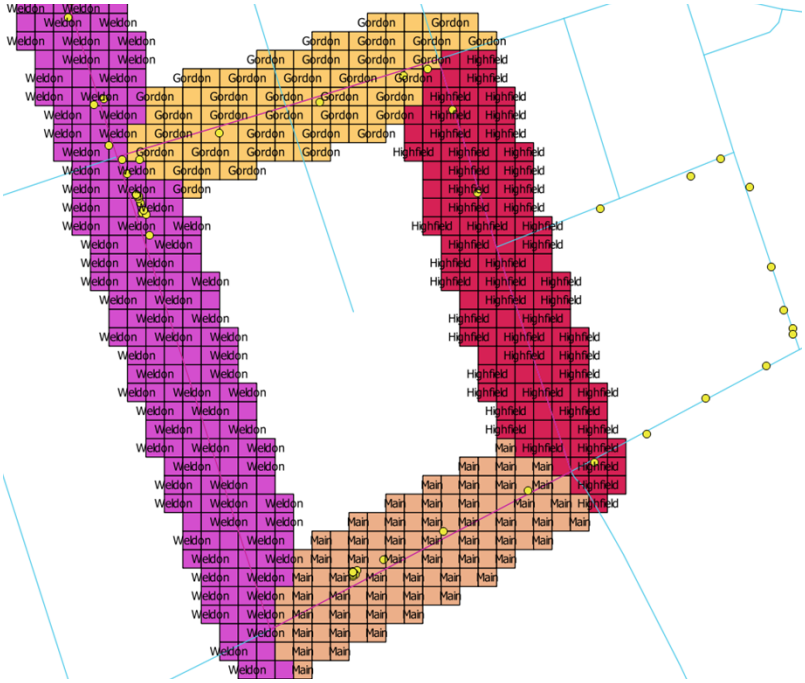
route. The results of the query are stored in a new geo-spatial table in our database. Finally, we query the GTFS table to get the list street names of the bus route and assign the street name for each square cell (See Figure 4.8a).

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 4.5: Continuous queries used for creating the grid cells.

Figure 4.8 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 4.8). 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 4.8: 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 4.9a). 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 4.9b). 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 4.9).

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 4.6).

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

Table 4.6: 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 4.10).

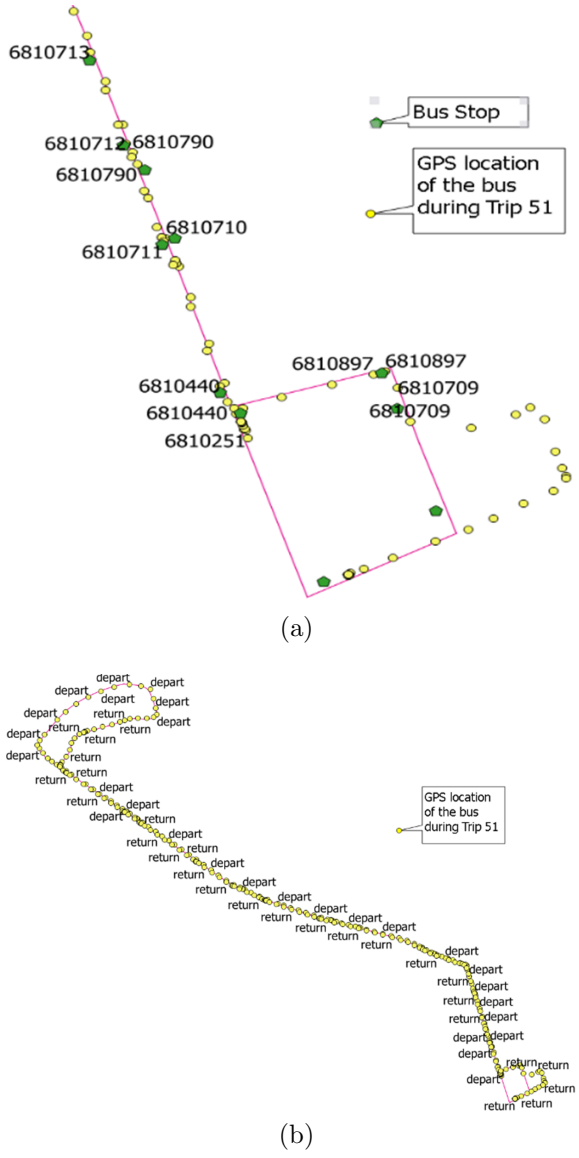


Figure 4.9: Results of the bus stop identification step for one trip of bus route 51.

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 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 4.11a). This step can be improved if automatic passenger counters (APCs) are used in a transit network because they

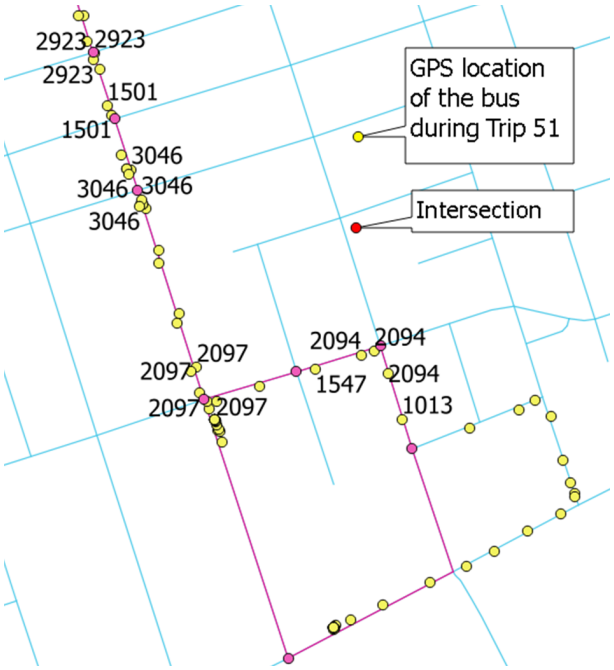


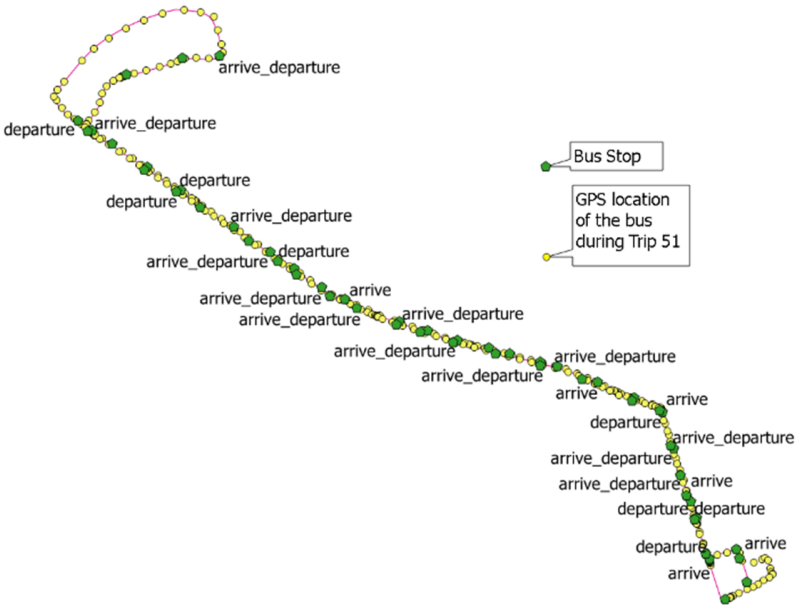
Figure 4.10: Results of the intersections identification step for one trip of bus route 51.

provide information about passenger activity on bus trip time.

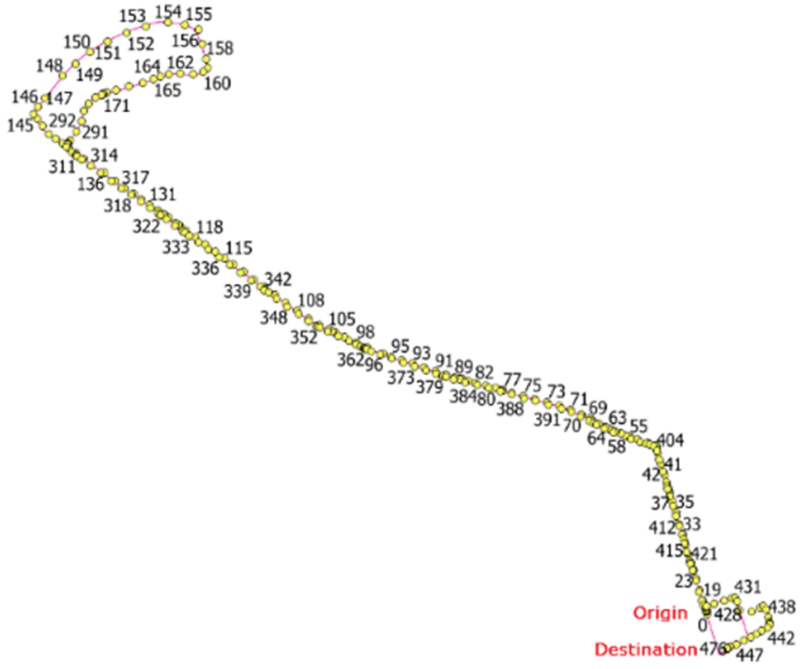
Step 7 - Trip Annotation: Finally, the last step was to tag each first tuple of a bus trip as origin, and each last tuple of a bus trip as destination (Figure 4.11b).

4.6 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.



(a)



(b)

Figure 4.11: Results from steps 6 and 7 of the contextualization task.

4.6.1 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 4.12 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.

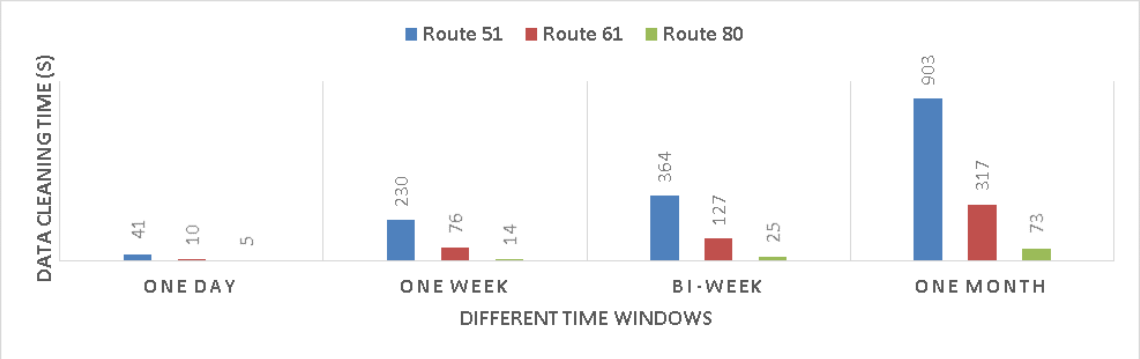


Figure 4.12: The measured cleaning times of 3 sample bus routes (51, 61, 80) operating in different areas over different time windows.

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 4.13 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.

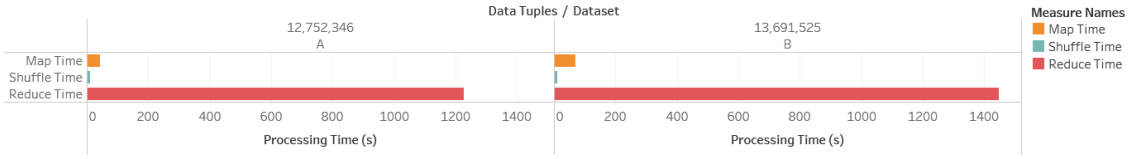


Figure 4.13: The measured processing times obtained from the MapReduce framework.

4.6.2 Experiment Evaluation

Table 4.7 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

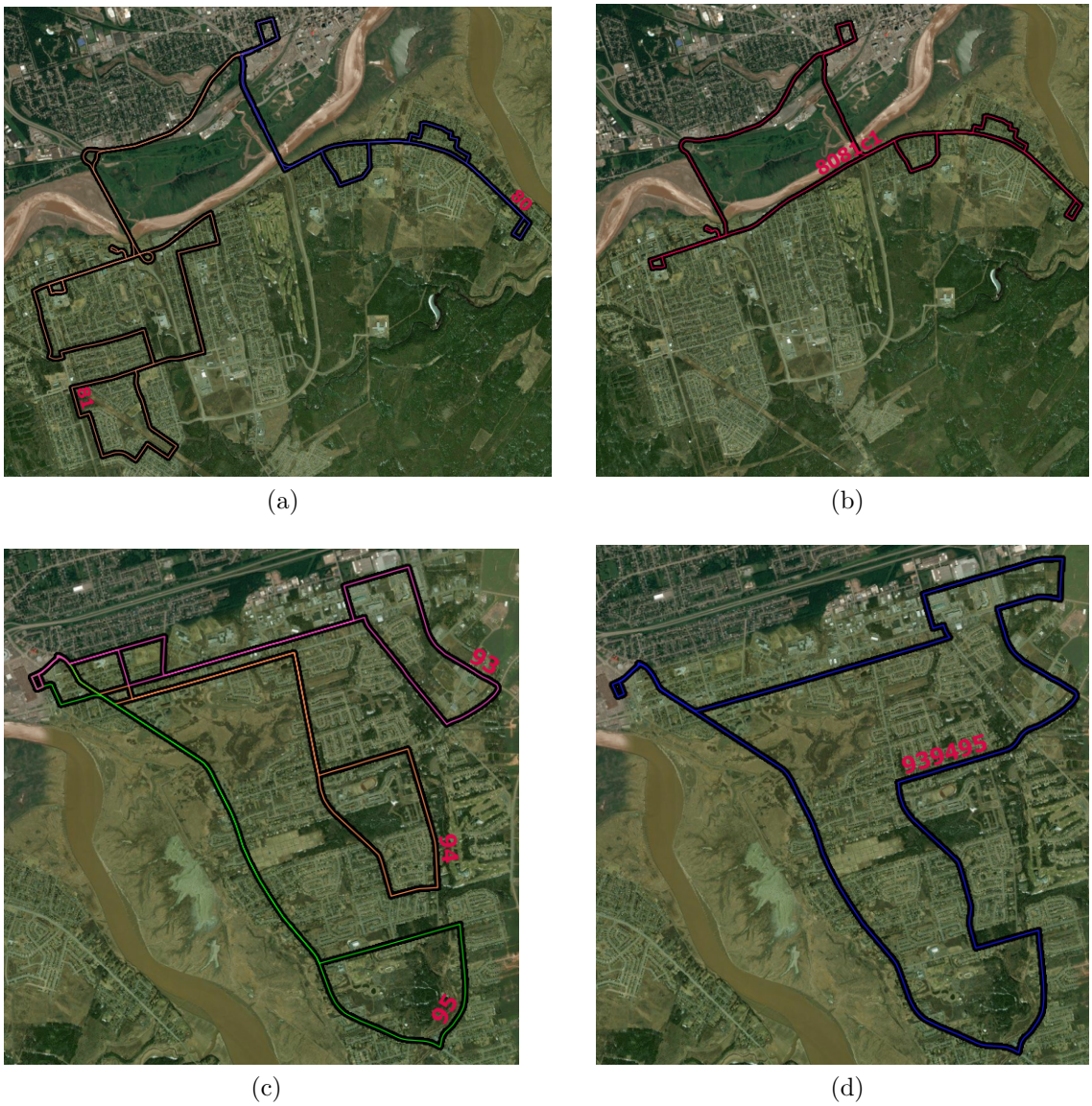
BUS ROUTE	MOBILITY CONTEXT						
	Moves	Stops	Running	Passing	Stopover	Movement suspended	Trips
50	661,373	552,356	643,592	17,781	268,294	284,062	4,270
50S	45,809	35,820	44,755	1,054	13,113	22,707	301
51	2,049,041	2,135,951	1,779,599	269,442	888,435	1,247,516	10,263
52	1,359,354	2,079,425	1,288,311	71,043	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 4.7: Contextual statistics for the Codiac transit network.

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 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 4.7 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 4.14 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 4.14: 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 4.8 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.

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 4.9 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 4.15 shows that the highest number 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"*.

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

Furthermore, the most congested intersections were found by looking at the the total number of the suspension of movement for bus route 51. Figure 4.16 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

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

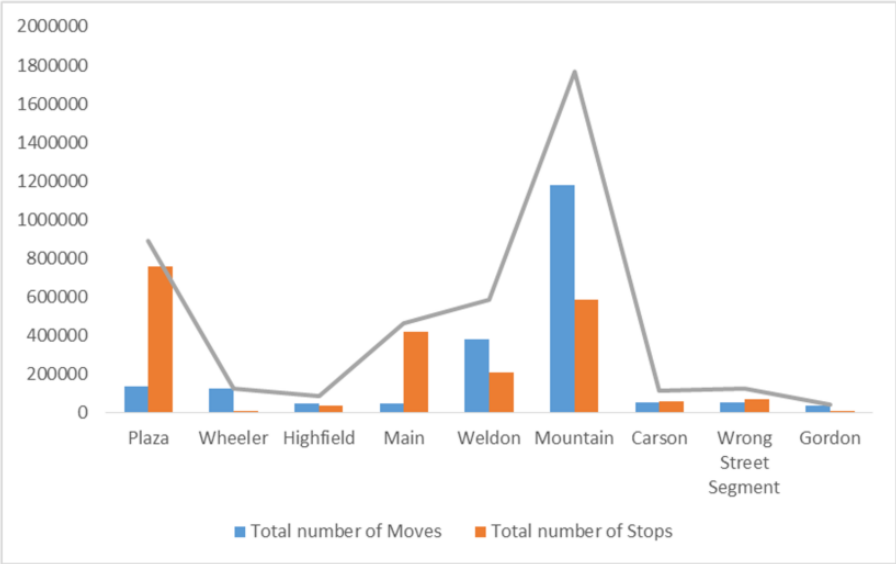


Figure 4.15: Overview of the total number of stops and moves of all trips of bus route 51.

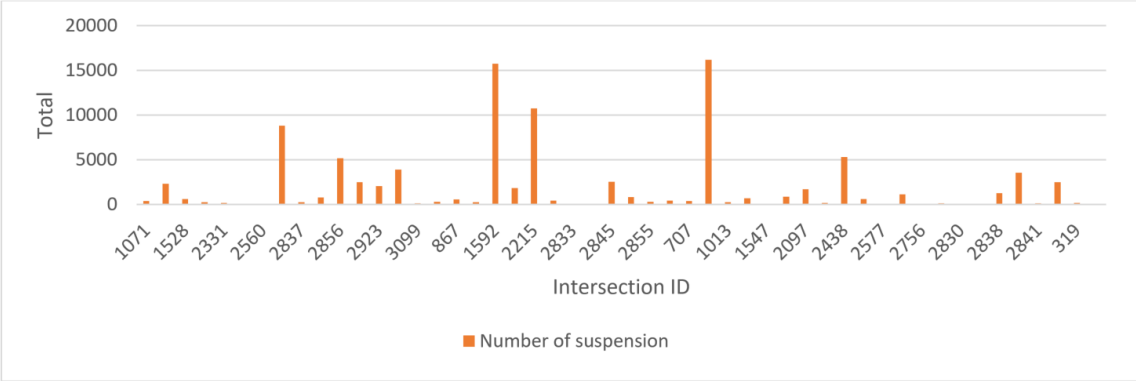


Figure 4.16: Total number of stops (suspension of movement) per intersection for all trips of the bus route 51.

their suspension of movement at the intersections might cause delays. Figure 4.17 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 4.18 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

Figure 4.17: The movement suspended pattern along the bus route 51.

a period of one year. According to this analytical result, the allocated resource for the bus stops need to be optimized to eliminate the redundant bus stops along this bus route.

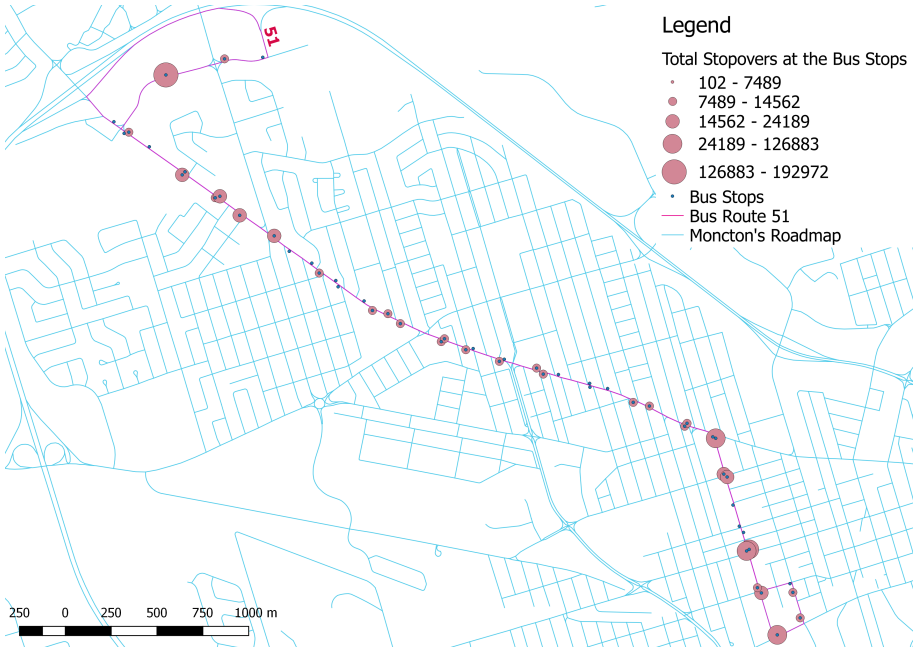


Figure 4.18: Total number of stopovers at each bus stop for all trips of the bus route 51.

4.7 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.

References

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and 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., and 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*, pages 1239–1246. ACM.

Banos, O., Amin, M. B., Khan, W. A., Afzal, M., Hussain, M., Kang, B. H., and 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., and 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., and 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., and 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., and Grace, P. (2017). A Reference Architecture for federating IoT infrastructures supporting semantic interoperability. In *2017 European Conference on Networks and Communications (EuCNC)*, pages 1–6. IEEE.

Cavalcante, E., Pereira, J., Alves, M. P., Maia, P., Moura, R., Batista, T., Delicato, F. C., and 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. and Collet, C. (2016). A scalability comparison study of data management approaches for smart metering systems. In *Parallel Processing (ICPP), 2016 45th International Conference on*, pages 474–483. IEEE.

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

Codiac (2018). Codiac transpo sevices. `http://www.codiactranspo.ca/Information/About_Codiac_Transpo.htm`. [Online; accessed on 2018-09-30].

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

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

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

Doulkeridis, C. and 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., and Savaglio, C. (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)*, pages 493–498. IEEE.

Gama, J. and Rodrigues, P. P. (2007). Data stream processing. In *Learning from Data Streams*, pages 25–39. Springer.

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., and Lee, U. (2014). Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 241–246. IEEE.

Giannella, C., Han, J., Pei, J., Yan, X., and 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., and Willinger, W. (2016). Network monitoring as a streaming analytics problem. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pages 106–112. ACM.

Isukapati, I. K., Rudová, H., Barlow, G. J., and 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. and Mouftah, H. T. (2014). Mobility-aware trustworthy crowdsourcing in cloud-centric internet of things. In *Computers and Communication (ISCC), 2014 IEEE Symposium on*, pages 1–6. IEEE.

Krco, S., Pokric, B., and Carrez, F. (2014). Designing IoT architecture(s): A European perspective. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 79–84. IEEE.

Lee, G., Yun, U., and 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., and 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*, pages 174–196. Springer.

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

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

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

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

Lv, Z., Song, H., Basanta-Val, P., Steed, A., and 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., and 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*, pages 764–769. IEEE.

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

Nelson, A., Toth, G., Hoffman, D., Nguyen, C., and 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*, pages 63–68, New York, New York, USA. ACM Press.

Ngu, A. H., Gutierrez, M., Metsis, V., Nepal, S., and 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., and 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., and 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*, pages 1218–1225. IEEE.

Rajeshwari, U. and 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*, pages 439–444. IEEE.

Ranasinghe, Y. S. and Walpola, M. J. (2016). Integrating context-awareness with

reminder tools. In *Advances in ICT for Emerging Regions (ICTer), 2016 Sixteenth International Conference on*, pages 216–221. IEEE.

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

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

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

Shibata, Y. and Sato, G. (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)*, pages 645–648. IEEE.

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

Song, H., Srinivasan, R., Sookoor, T., and 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*, pages 101–106. IEEE.

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

Vaquero, L. M. and 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., and 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*, pages 2091–2105. ACM.

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

Wang, L. and Ranjan, R. (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., and McCann, J. A. (2015). UbiFlow: Mobility management in urban-scale software defined IoT. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 208–216. IEEE.

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

Chapter 5

Conclusions and Future Research Work

5.1 Summary of the Research

In the past few years, the IoT research paradigm has been shifting from building physical infrastructures to developing analytical capabilities according to the requirements of IoT applications. However, the mobility and co-location of IoT devices, coupled with the high velocity, variety, and volume of IoT data streams created by these geo-distributed devices, have presented significant challenges for handling the data streams in a timely way. This dissertation proposed a conceptual framework that has addressed these challenges and provided both low latency and low complexity.

Three published articles have been selected to describe the core of the proposed *"Analytics Everywhere"* framework in *Chapters 2, 3, 4*. The summary of each chapter is presented as follows:

- *Chapter 2* proposed my *"Analytics Everywhere"* framework, which has demonstrated the possibility of developing a conceptual prototype that is capable of integrating a network of distributed compute nodes (i.e. edge, fog, and cloud nodes) into a unique continuum. This contiuum provided a variety of computing resources to perform multiple analytical capabilities, including descriptive, diagnostics, and predictive analytics. In this framework, a data life-cycle, which consists of accumulated IoT data streams, plays an important role in *a priori* mapping between analytical capabilities with the appropriate computation resources to ensure that each automated analytical task had the right data at the right time. The proposed framework has also been validated through a smart transit scenario to support three groups of users (e.g. transit operators, bus drivers, and passengers) with different user applications (e.g. schedule adherence, abnormalities detection, and trip behaviors prediction).

- *Chapter 3* presented the design and implementation for an edge-fog-cloud architecture to support a seamless execution of automated analytical tasks of my *"Analytics Everywhere"* framework. As an evolution of the last chapter, this architecture has been designed with several main modules including Admin/Control, Stream Processing & Analytics, Run Time, Provision & Orchestration, and Security & Governance that were integrated into a unique fabric working on top of an edge-fog-cloud continuum. Two data life-cycles have been implemented to carry on various analytical tasks (descriptive, diagnostics, and predictive analytics) in both continuous and accumulated IoT data streams. The proposed framework has been validated using a smart parking scenario in the City of Saint John, NB, Canada.

- *Chapter 4* described the development of an IoT-GIS platform for supporting automated tasks to retrieve, integrate, and contextualize IoT data streams with the purpose of adding value and insights to the provision of transit services.

Our proposed platform was designed with the ability of couple data streams with automated tasks with the purpose of assessing existing transit services. Since mobility plays an important role in the explanation of the phenomena, it reinforces different perspectives and provides a true understanding of the background of the IoT problems. This IoT-GIS platform was designed to perform three automated analytical tasks: data ingestion, data cleaning, and data contextualization. The platform also took into account the mobility contexts given by a transit agency of a small urban area. It has shown the potential to provide operational information to improve the average speed of service.

5.2 Research Contributions

The research questions in this dissertation revolve around:

- How to combine various resource capabilities for performing multiple analytical capabilities based on different IoT data life-cycles using continuous and accumulated streams to form an unique integrated framework?
- How to integrate IoT and GIS into the edge-fog-cloud continuum without compromising resource capabilities?
- What the advantages and disadvantages of the proposed approach in smart cities are.

This dissertation has demonstrated that it is possible to combine various computational resources, such as compute nodes at the edge, fog, and cloud computing environments to support IoT applications. Moreover, it has also proved the feasibility of implementing various automated analytical tasks for supporting streaming descriptive, streaming diagnostic, and streaming predictive analytics according to

different IoT data life-cycles. For example, two data life-cycles have been developed and executed to perform different analytical capabilities based on both continuous and accumulated streams in *Chapter 3*, while *Chapters 2* and *4* described automated analytical tasks using accumulated streams for supporting various streaming descriptive, streaming diagnostic, and streaming predictive analytics.

Aiming to find a solution to integrate IoT and GIS into the edge-fog-cloud continuum without compromising resource capabilities, *Chapter 4* proposed an IoT-GIS platform to retrieve, integrate, and contextualize accumulated data streams in the cloud. This platform is not fully integrated into an edge-fog-cloud continuum. However, it is a first step towards designing an IoT-GIS platform that is able to provide GIS functionalities and interoperability between IoT devices and big data analytics softwares to analyze massive amounts of data streams without human intervention. In addition, this platform indicates that there is a need to move and implement some analytical tasks from the cloud computing environment to the edge and fog computing environments in order to bring analytical capability closer to the IoT devices and to fully integrate this platform into an edge-fog-cloud continuum.

Finally, two IoT applications (smart transit and smart parking applications) have been developed and implemented that aims to validate the advantages and disadvantages of the proposed *"Analytics Everywhere"* in smart cities. The proposed *"Analytics Everywhere"* framework was developed based on the research premise that IoT applications are convenient for pushing the computation toward the edge network while trying to keep most of the data as close as possible to where it is generated. Some immediate advantages can be obtained from this approach. For example, data privacy can be retained to the user to a certain extent. Also, the cost to transfer enormous amounts of data to the remote data centers can be reduced, while the analytical results can be delivered quickly to a variety of users. Furthermore, the

high availability of the provided IoT applications and services can be preserved to a certain extent, since the analytical tasks are able to perform in close proximity to the IoT devices. However, it is worth noting that the proposed *"Analytical Everywhere"* framework does not need to be modified to support dynamic task sharing, since the analytical tasks are assumed to be *a priori* allocated, exploiting the different resources regardless of workload balancing.

By proposing a novel *"Analytics Everywhere"* framework, several major scientific contributions have been emerged during this research which can be summarized as follows:

1. I proposed an *"Analytical Everywhere"* framework that integrates computational resources needed for a seamless execution of a network of analytical tasks having automated analytical capabilities, generating useful and high level information in a timely way.

2. I demonstrated that a single computational resource (e.g. cloud) is not sufficient to support all analytical capabilities that are needed for IoT applications, considering computing power, data stream management, storage, and networking capabilities.

3. I discussed the challenges and how an *"Analytics Everywhere"* framework can be designed to perform descriptive, diagnostic, and predictive analytical tasks. Moreover, the proposed *"Analytics Everywhere"* framework has been validated using the smart transit and the smart parking scenarios by highlighting the pitfalls and discussing the experiences.

4. Most of the IoT architectures rely on a cloud environment in which n-tiers of horizontal layers are designed to perform analytical tasks. My approach proposed a new architecture based on an integrated fabric of compute nodes that

are designed to work together to perform many analytical tasks, which are triggered by IoT data streams transported through an edge-fog-cloud continuum.

5. Automated analytics for IoT data streams is still in its infancy, and applications usually require a diverse number of outputs having different temporal granularities. There have been very little research reported on the impact of analytical tasks in the IoT architectures. The scientific contribution of this research is therefore to ascertain this impact using a smart parking and smart transit scenario.

6. An IoT-GIS platform has been designed and developed to handle the continuous incoming data streams from IoT devices through a variety of analytical tasks performing on this platform.

7. IoT data streams have been explored via the IoT-GIS platform not only temporally and spatially, but the notion of a mobility context is also integrated into this platform so that it can help us to explain the phenomena, reinforcing different perspectives, and providing a greater understanding of the background of the problems.

8. Automated analytical tasks have been implemented on the IoT-GIS platform without human intervention and can simultaneously cope with the incoming unbounded IoT data stream at a high data rate.

5.3 Future Work

The proposed *"Analytics Everywhere"* framework can also be applied in many other smart city applications and can provide some benefits as aforementioned. However, *a priori* mapping of streaming analytical tasks and of computational resources in a

static manner is a limitation that needs to be addressed in the near future. Hence, this study opens up many new research directions. Specifically, streaming analytical tasks and computational resources could be dynamically mapped together in the next development step. The support of continuous and accumulated data life-cycle orchestration will play a paramount role in mapping the analytical capability and computational resources in a dynamic sense. To achieve this goal, further research towards mitigating the issue with the *concept drift* to be addressed in the future. Since the underlying distribution of the incoming IoT data stream may change over time. As a result, the algorithms/models, which are built based on the older data tuples, will be obsoleted and no longer accurately reflected the distribution of the new data tuples.

One important lesson drawn from this study is that if any aspect of the edge-fog-cloud resources is considered in isolation, it would not be able to manage the data life-cycles of IoT applications without compromising functionality or performance. Many threats to the validity of the proposed architecture using the edge-fog-cloud computing might arise in other IoT applications. Therefore, the proposed framework should be extended by considering the security, latency, fault tolerance, and privacy requirements of IoT applications. Moreover, it is challenging to find the optimal balance of all analytics tasks leading to the highest value chain discovered from IoT data streams. This is because there are currently no metrics to compute the correlation between the complexity and the obtained new insights. Thus, a new set of metrics is needed for identifying the balance point and for providing feedback about the optimal performance of the framework. Other IoT applications in smart cities should be implemented for further validation of the proposed framework.

In an attempt to integrate IoT and GIS in the proposed *"Analytics Everywhere"* framework, an IoT-GIS platform has been developed in the cloud computing

environment to provide the automated streaming analytical tasks in a mobility context. However, the results from the data cleaning task proved that it is not reasonable to send all the data streams to the cloud. Thus, extending streaming analytical tasks and GIS functionalities to the edge and fog computing environment is definitely the next research step.

The proposed *"Analytics Everywhere"* framework is capable of producing knowledge/insights to create value within each streaming analytical task. However, the next research question is: How can a streaming analytical task in the network of tasks reuse and exploit the gained knowledge from the other tasks to resolve its own problems? A potential approach that could answer this research question is the transfer learning methods. Transfer learning techniques involve the concept of improving learning in a specific domain (target domain) by training the learning model with the datasets from other domains (multiple source domains), using similar features and constraints. Therefore, our future research work will also focus on developing a transfer learning process for the proposed *"Analytics Everywhere"* framework. A transfer learning process should be considered as a new pillar to be developed and integrated into the current proposed framework. The learning process could be described as the outcomes of the analytical tasks which are capable of being transferred within the distributed compute nodes that are involved in a data life-cycle to provide the highest overall performance. Transfer learning tasks will be *a priori* defined or dynamically allocated depending on future IoT research scenarios. Some suggested approaches for implementing transfer learning settings in this framework includes inductive transfer learning, transductive transfer learning, and unsupervised transfer learning.

Appendices

Appendix A

Results from the Descriptive Analytics Task at the Edge

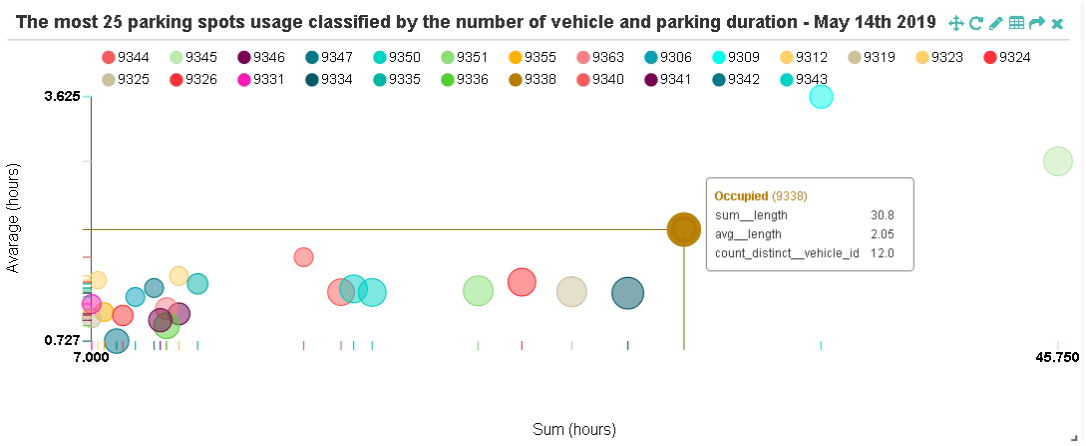
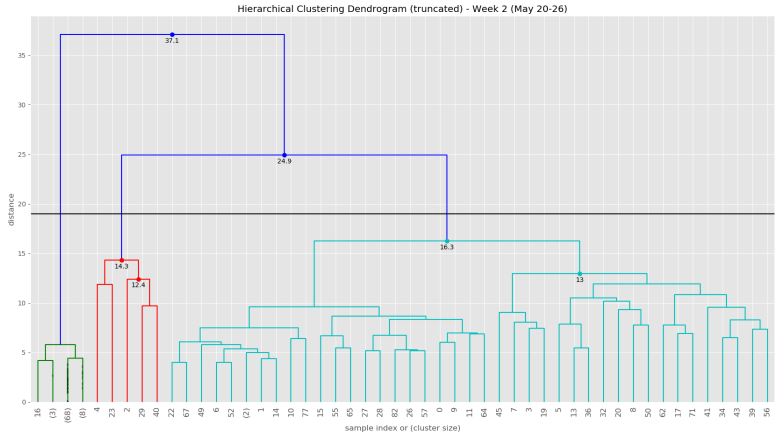


Figure A.1: Parking usage pattern at each spot on 14 May 2019.

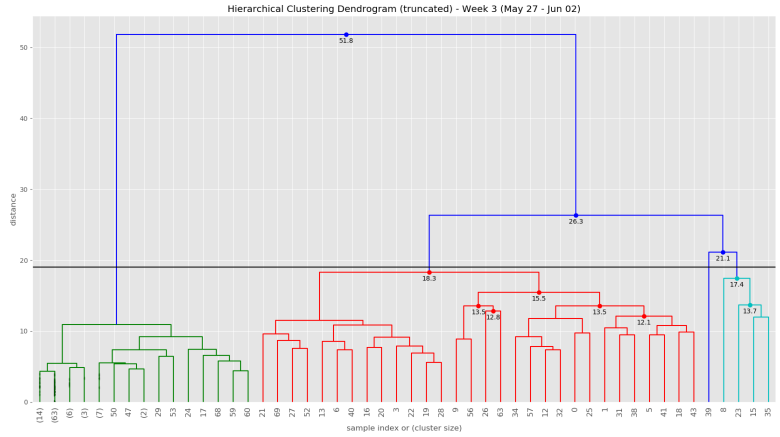
Appendix B

Clustering Results from the Diagnostics Analytics Task at the Fog

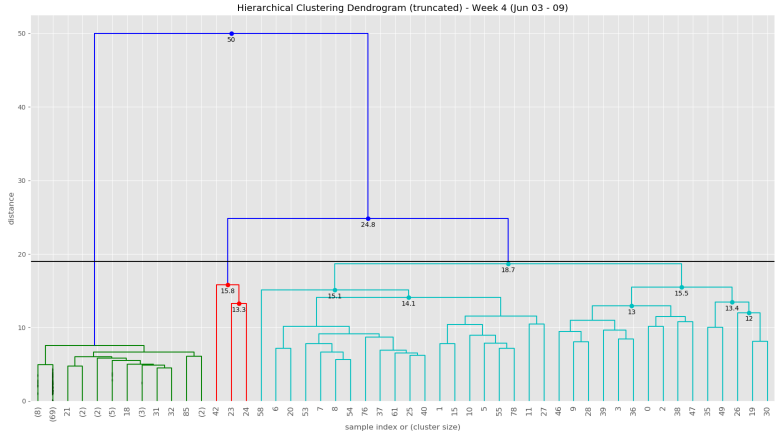
B.1 Dendrogram of Clusters for the Next 4 Weeks of Observation



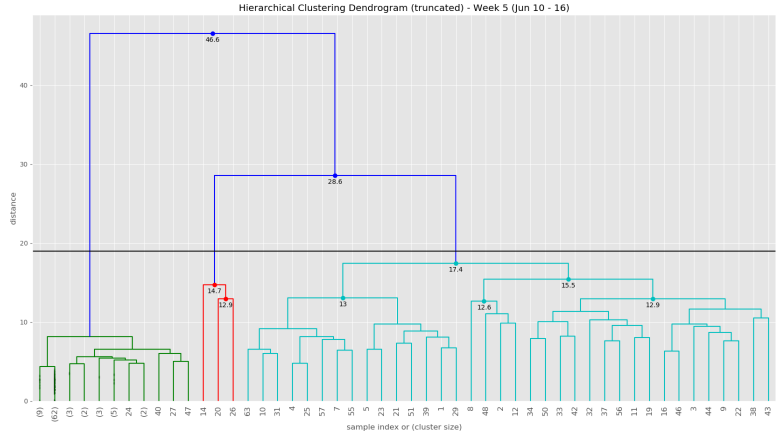
(a) Week 2



(b) Week 3



(c) Week 4



(d) Week 5

Figure B.1: Dendrogram of clusters.

B.2 Clustering Results for the Next 4 Weeks of Observation

Figure B.2: Clustering results for the 2nd week.

Figure B.3: Clustering results for the 3rd week.

Figure B.4: Clustering results for the 4th week.

Figure B.5: Clustering results for the 5th week.

B.3 Clustering Results Using PCA for the Next 4 Weeks of Observation

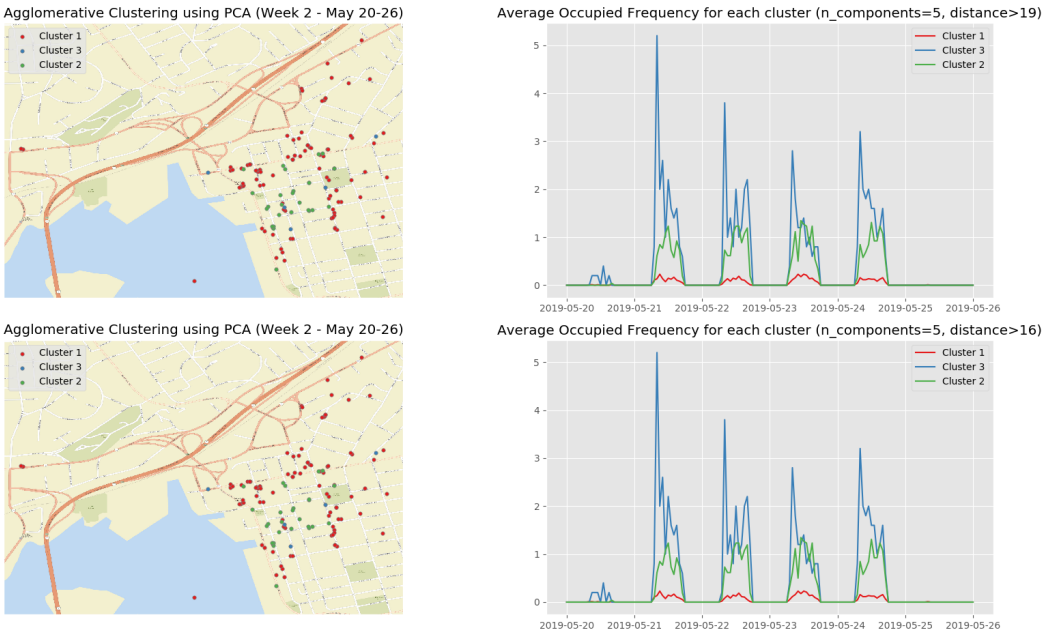


Figure B.6: Clustering results using PCA for the 2nd week.

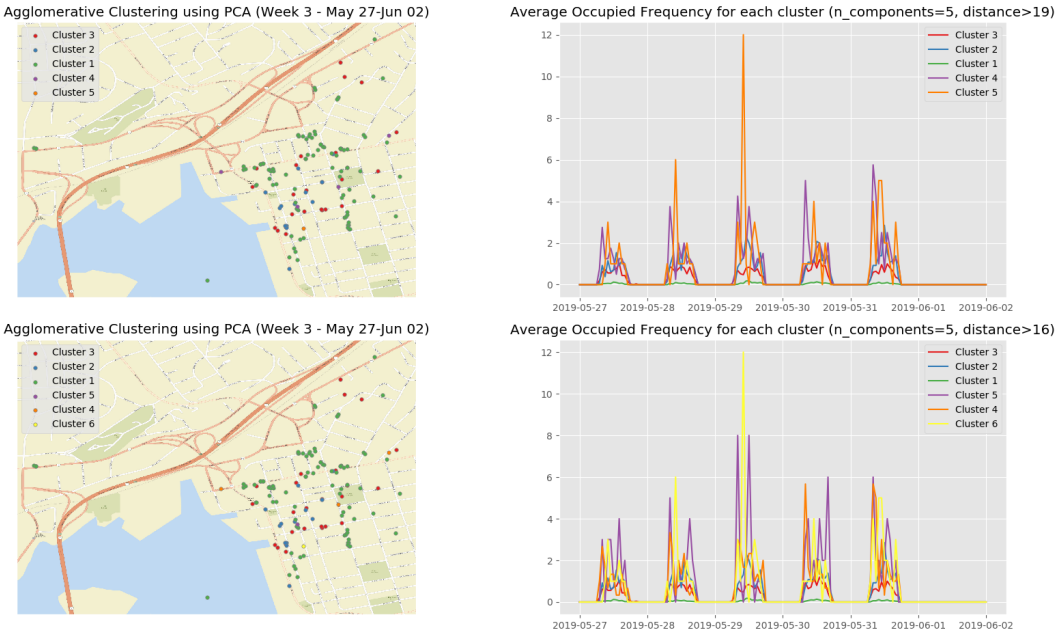


Figure B.7: Clustering results using PCA for the 3rd week.

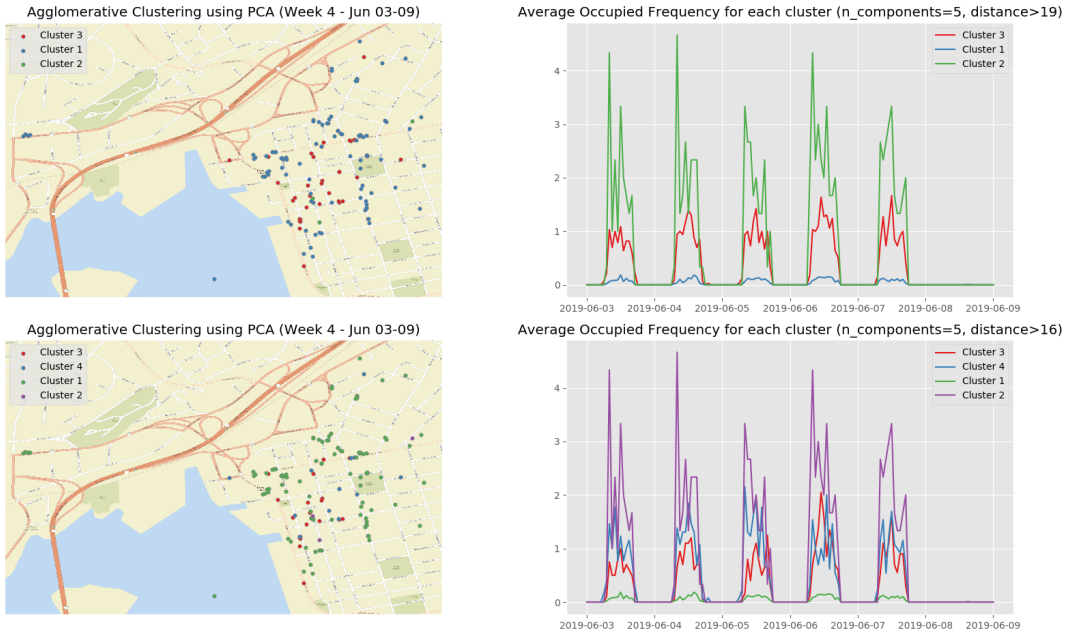


Figure B.8: Clustering results using PCA for the 4th week.

Figure B.9: Clustering results using PCA for the 5th week.

Curriculum Vitae

Candidate's full name: **HUNG VAN CAO**

Universities attended:

2016 – 2019	**Ph.D. Candidate in Geomatics Engineering**, *University of New Brunswick*, Canada
2017 – 2018	**Diploma in University Teaching**, *University of New Brunswick*, Canada
2014 – 2015	**M.Sc. in Computer Science**, *University College Dublin*, Ireland
2006 – 2011	**B.Eng. in Computer Engineering**, *University of Information Technology, Vietnam National University Ho Chi Minh City*, Vietnam

List of Publications

During my PhD research work, there are a number of publications have emerged which are listed as follows:

1. **Cao, H.**, & Wachowicz, M. (2019). An Edge-Fog-Cloud Architecture of Streaming Analytics for Internet of Things Applications. Special Issue Edge/-Fog/Cloud Computing in the Internet of Things. In *Sensors,* 19(16), 3594. (**Peer Reviewed** - Impact Factor: 3.031)

2. **Cao, H.**, Wachowicz, M., Renso, C., & Carlini, E. (2019). Analytics Everywhere: generating insights from the Internet of Things. In *IEEE Access,* 7, 71749-71769. (**Peer Reviewed** - Impact Factor: 4.098)

3. **Cao, H.**, & Wachowicz, M. (2019). The design of an IoT-GIS platform for performing automated analytical tasks. In *Computers, Environment and Urban Systems,* 74, 23-40. (**Peer Reviewed** - Impact Factor: 3.393)

4. **Cao, H.**, & Wachowicz, M. (2019). Analytics Everywhere for streaming IoT data. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security.* IEEE. Granada, Spain. (**Peer Reviewed**)

5. **Cao, H.**, Brown M., Chen L., Smith R., & Wachowicz, M. (2019). Lessons learned from integrating batch and stream processing using IoT data. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security.* IEEE. Granada, Spain. (**Peer Reviewed**)

6. Parise A., Callejo, M. A. M., **Cao, H.**, Mendonca M., Kohli H., & Wachowicz, M. (2019). Indoor Occupancy Prediction using an IoT Platform. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security.* IEEE. Granada, Spain. (**Peer Reviewed**)

7. **Cao, H.**, Wachowicz, M., Renso, C., & Carlini, E. (2018). An edge-fog-cloud platform for anticipatory learning process designed for internet of mobile things. In *arXiv: 1711.09745.*

8. **Cao, H.**, Wachowicz, M., & Cha, S. (2017, December). Developing an edge computing platform for real-time descriptive analytics. In *Big Data (Big Data), 2017 IEEE International Conference on (pp. 4546-4554).* IEEE. Boston, MA, USA. (**Peer Reviewed**)

9. Maduako, I., **Cao, H.**, Hernandez, L., & Wachowicz, M. (2017, October). Combining edge and cloud computing for mobility analytics. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing (p. 22).* ACM. San Jose, CA, USA. (**Peer Reviewed**)

10. Hernandez, L., **Cao, H.**, & Wachowicz, M. (2017, October). Implementing an Edge-Fog-Cloud architecture for stream data management. In *Fog World Congress (FWC), 2017 IEEE (pp. 1-6).* IEEE. Santa Clara, CA, USA. (**Peer Reviewed**)

11. **Cao, H.**, & Wachowicz, M. (2017, August). The design of a streaming analytical workflow for processing massive transit feeds. In *2nd International Symposium on Spatiotemporal Computing.* Harvard University, Cambridge, MA, USA. (**Peer Reviewed**)

12. **Cao, H.** (2017). What is the next innovation after the internet of things?. In *arXiv:1708.07160.*

13. Cha, S., Ruiz, M. P., Wachowicz, M., Tran, L. H., **Cao, H.**, & Maduako, I. (2016, December). The role of an IoT platform in the design of real-time recommender systems. In *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on (pp. 448-453).* IEEE. Reston, Virginia, USA. (**Peer Reviewed**)

14. **Cao, H.**, Maduako, I., Cavalheri, E., Brideau, E., & Wachowicz, M. (2016, September). The Role of Graph Databases in Geomatics. In *Geomatics Atlantics 2016.* Fredericton, NB, Canada.

15. **Cao, H.**, Maduako, I., Cavalheri, E., Brideau, E., & Wachowicz, M. (2016, September). How can graph databases improve transit systems? In *UNB Research Showcase.* University of New Brunswick, Fredericton, NB, Canada.