big data and cognitive computing

Article

MOBDA: Microservice-Oriented Big Data Architecture for Smart City Transport Systems

Suriya Priya R. Asaithambi [1], Ramanathan Venkatraman [1] and Sitalakshmi Venkatraman [2,*]

1 Institute of Systems Science, National University of Singapore, Singapore 119077, Singapore; suria@nus.edu.sg (S.P.R.A.); rvenkat@nus.edu.sg (R.V.)
2 Department of Information Technology, Melbourne Polytechnic, Prahran, VIC 3181, Australia
* Correspondence: sitavenkat@melbournepolytechnic.edu.au; Tel.: +61-3-9269-1711

Received: 27 May 2020; Accepted: 6 July 2020; Published: 9 July 2020

Abstract: Highly populated cities depend highly on intelligent transportation systems (ITSs) for reliable and efficient resource utilization and traffic management. Current transportation systems struggle to meet different stakeholder expectations while trying their best to optimize resources in providing various transport services. This paper proposes a Microservice-Oriented Big Data Architecture (MOBDA) incorporating data processing techniques, such as predictive modelling for achieving smart transportation and analytics microservices required towards smart cities of the future. We postulate key transportation metrics applied on various sources of transportation data to serve this objective. A novel hybrid architecture is proposed to combine stream processing and batch processing of big data for a smart computation of microservice-oriented transportation metrics that can serve the different needs of stakeholders. Development of such an architecture for smart transportation and analytics will improve the predictability of transport supply for transport providers and transport authority as well as enhance consumer satisfaction during peak periods.

Keywords: smart technologies; intelligent transportation systems; big data; microservice-oriented big data architecture; data analytics

1. Introduction

In the past decade, steady population growth and digital technology advancement have formed the key influencing factors for moving towards intelligent transportation systems (ITSs) [1–3]. With the increase in population, the growing traffic congestion in the existing transportation infrastructure poses challenges in meeting the required service performance for realizing any ITS of the future [4–6]. In densely populated cities, efficiency in public transportation systems such as buses and taxis play a major role in traffic management on public roads. Hence, population growth in urban cities worldwide contribute to the ever-changing public transportation needs of diverse user groups. Enhancing the service performance is a key component for achieving stakeholder satisfaction in ITSs. With digital technology developments, huge data is being generated from vehicles, commuters, and various other sources such as electronic sensors, passenger travelling logs, traffic data, global positioning systems, and even images and videos from surveillance systems. The revolution of Industry 4.0 with IoT and big data technologies has significantly influenced the way of public life [7,8]. Advancements in wireless communications of sensors and edge devices have introduced many loosely coupled task-specific cloud and mobile applications (Apps) offered as microservices. The dire need for transport-related information has increased with the capabilities of data analytics and the culture of easy access to "must-have" information has become the norm over time [1,9,10]. Transportation providers and regulators expect to have all the information with regard to the existing transportation network, environmental situation such as weather condition that correlate to the demand of transportation services, movement of commuters and vehicles, and their insights with the aim to manage a cost-efficient service [5,11,12].

Technological advancements in the capture and storage of data have increased the potential to monitor both user behavior and the physical world, with provisions to track and triangulate diverse data sets using loosely coupled and independent services or microservices [13,14]. However, vast challenges remain for storing, analyzing and visualizing such volumes of heterogeneous data at the speed they are being generated [15,16]. This becomes more challenging when transportation stakeholders are required to monitor certain metrics on-the-fly so that real-time data-driven decision making can be made. The major technical challenges of big data applications are: (i) Varying data structure of the input format, (ii) dynamically changing data definition of the output to allow for easy exchange across different systems (velocity and variety), (iii) selecting optimal algorithm for processing huge amount of data with an acceptable computational burden (volume), (iv) post-processing of data to make them easily understandable despite the informational complexity, and (v) analyzing data deeply to gain potential insights (value) [16,17]. Hence, there is a need for a reliable and scalable big data architecture with a microservice-oriented objective to support real-time processing of massive data in providing intelligent and meaningful data analytics for all the transportation stakeholders. This would mark the first steps in achieving a practically successful ITS of the future.

A typical ITS is expected to have a wide range of service-driven provisions such as dynamic alternate route recommendations for vehicles, on-demand public bus service, vehicle navigation mobile Apps, traffic signal control systems, automatic number plate recognition, automatic incident-detection, real-time parking lot availability, and dynamic schedule updates of public transport for users [18,19]. In densely populated cities such as Singapore and Hongkong, such an ITS and the associated microservices have been pivotal in enhancing their public transportation systems, as well as in optimizing the limited land space for roads in providing a convenient, safe, and comfortable travelling experience for its users [1,5,20]. However, recent population growth and Industry 4.0 evolution warrant an ITS of the future to address various security and privacy challenges as well as to meet the dynamically changing service level expectations of its wide range of stakeholders [21,22]. Therefore, a big data architecture that allows for dynamic processing is warranted with the objective to deliver relevant and high-quality transport information in real-time, and to ensure data security and privacy for its stakeholders who directly or indirectly contribute to the big data collection.

Overall this paper proposes a service-driven approach for building an effective big data Architecture for a smart land transportation and analytics platform. To achieve this objective, we formulate a novel Microservice-Oriented Big Data Architecture (MOBDA) describing a hybrid model's architectural principles and its components to suit a dynamically adaptable ITS.

The rest of the paper is structured as follows. Section 2 on related work provides a brief survey of the research landscape for similar architectural contextualization and proposals. In Section 3, we explain the service-driven modeling for an ITS including the functional and non-functional microservice performance requirements of the proposed architecture. This section also discusses an assorted set of metrics as performance indicators of the analytics platform that are useful for evaluating the typical transport services in meeting the requirements of various stakeholders. In Section 4, we explain the components of our hybrid proposal of MOBDA, with the prime focus of achieving desired performance and service level efficiency in an ITS. The component selection, configuration, and implementation aspects of the framework are discussed in detail. Section 5 provides a use case of MOBDA for a smart transportation and analytics system. It establishes the practical application of MOBDA by describing the implementation and testing of the model conducted using public data sets. Finally, in Section 6, we conclude, highlighting our key contributions of our proposed architecture, design guidelines, and observations, including future work. We believe this study will be useful to both academia and industry in their big data architectural pursuits.

2. Related Work

Current progress in ITSs is based on the innovation in IoT and software industry that can provide a safe and reliable travel for commuters towards smart cities of the future improved services with performance metrics such as less waiting time for public transport, optimized transport capacity, automatic vehicle rerouting, on-time vehicle arrivals, traffic congestion free commutes, and reduced road accidents [6,9,18]. Modern transportation vehicles are powered by smart sensors, intelligent automation, and data analytics that can provide various information services for vehicle drivers, transport operators, and commuters for making various autonomous decisions on the road [23]. While microservice-oriented metrics from an ITS can provide data insights for enhancing transportation services, the digital transformation for smart systems catering to dynamically changing environment requires real-time adaptation [13,19]. It can be challenging to correlate and analyze data across various modes of transportation networks, vehicle operators, commuters, and other stakeholders to provide real-time visibility of data and its impact with a collaborative service-driven architecture [23,24]. Although developments in ITSs have progressed significantly in the last decade, most of the systems deployed so far rely on rigid, bespoke architectures with pre-defined goals to address only a limited number of services related issues.

Recently, data streaming-based architectures were studied using a simulator for scalability and response time for a smart city that can support real-time recommendations in improving driver efficiency and road safety [15]. A distributed streaming framework using Apache Kafka was adopted to test and benchmark the model configurations. Another study proposed a comprehensive and flexible architecture for real-time traffic control based on big data analytics [18]. Apache Kafka was not only used for data pipelines and stream-processing but also as a communication broker platform to connect the sensors (probe vehicles, loop detectors, etc.) and actuators (traffic lights, variable message sign, etc.) associated with the traffic system in collaboration with Hadoop Distributed File System (HDFS) data warehouse and an analytics engine for the ITS to enact change in traffic controls.

Data processing architectures that combine big data, IoT, machine learning (ML) models, and various other modern technologies form the backbone of smart systems, including ITSs. Currently, Lambda and Kappa are two popular data processing architectures being studied by researchers [25]. The Lambda architecture, developed before Kappa, consists of a batch processing layer, a transient real-time processing layer (stream layer), and a serving layer that mergers these layers. On the other hand, Kappa architecture considers data as a stream for real-time processing of distinct events and does not include batch processing. Several studies have compared Lambda and Kappa architectures with detailed discussion on their performance, advantages, and tradeoffs [17,25]. Key factors, such as the data columns being analyzed, the size of the cluster of nodes realizing the architectures and their characteristics, the deployment costs, as well as the quality of the output, were evaluated.

Several studies have considered areas that require processing of voluminous data in-motion and in real-time including stock market, e-commerce, fraud detection, social media, defense force, and healthcare apart from transportation [26,27]. One such study considered Lambda architecture to process real-time data streams from social media services (Github and Twitter) using Apache Storm to build a predictive model-based trends [28]. The stream layer included the data analytics with ML having "incremental" adaptive capabilities (using Apache Mahout) to keep the model up to date. Another study used stream/real-time computing to improve the quality of health services [26]. Big data analytics was employed to transform data collected from a wide variety of sources, such as healthcare records, biomedical imaging, clinical text reports, and sensor data (e.g., EEG), to predict epidemics, make clinical decisions, and diagnose and cure diseases.

Regarding the transportation domain, a real-life case study from Netherlands and Sweden had investigated the major challenges in their public transport industry [23]. The study focused on increasing efficiency, passenger ridership and consumer satisfaction by allowing data flow between service providers and consumers. Several real-time control strategies were designed and implemented based on historical data, actual and downstream conditions such as time of arrivals, travel times, headway distributions, traffic, fleet, and public demand. Another study discussed improvements in China's transportation through the integration of big data with the main goal to improve operational efficiency, reduce congestion and predicting demand [9,29]. Data from CCTV, sensors, and GPS were used to identify traffic streams and weather conditions and moving vehicle data to forecast road traffic and to provide an assessment of traffic conditions. In such situations, we find that a huge collection of independent and loosely coupled microservices are being developed. Beijing has been using a roadside electronic system to display traffic conditions of nearby road sections. Such real-time data stream processing and services are expected to assist the driver to divert vehicles from congested areas with the limitation that it covers only small areas of congested traffic. Similar works in this domain show that big data has been implemented for traffic scheduling, routing, and congestion management [10,19]. However, the highly heterogeneous transportation service ecosystem is complex with a huge variety in the modes of transport, types of vehicle manufacturers, commuters, management authorities, and regulators in the supply chains [30–32]. Data sharing and collaboration across multiple stakeholders, platforms, data formats, microservices, communications protocols, and sources of intelligence can be challenging and difficult. There is a need for a microservice-oriented big data architecture that can provide new and adaptable service offerings in an ITS to enable high service levels of safety, performance, and availability.

3. Service-Driven Modelling for ITSs

An ITS is designed with the main purpose of improving the services of a transportation system by applying state-of-the-art information communications technologies (ICT) smartly and effectively [33,34]. However, with the evolution of IoT and big data, ITS technologies are more and more characterized by heterogeneous and tightly coupled, ad hoc solutions that are catering to specific transportation services. It becomes difficult to integrate, interoperate, extend, and reuse these disparate heterogenous transportation services in an agile and seamless manner. Service-driven modelling paradigms, namely service-oriented architecture (SOA) a decade ago, and microservice-oriented architecture (MOA) of recent days, have the underlying principle to enable disparate service composition, and are considered appropriate for automation solutions in industries that require provision of dynamic and heterogenous services [32,35,36]. A microservice is a highly cohesive, single-purpose, and decentralized service running in its own process and communicating with lightweight mechanisms. Microservices can be developed to meet business stakeholder requirements and independently implemented using automated deployment approaches. The advantage of adopting such microservices in an ITS is that a set of distributed microservices can be linked independent of hardware and software platforms via external interfaces of heterogeneous systems for implementing value-added transportation services. Hence, microservice-oriented modelling would meet the key requirements of ITSs of the future, such as integration, interoperability, adaptability, and scalability. MOA provides a microservice-oriented modelling approach for designing and implementing the interaction and communication among software service layers to support information sharing among different stakeholders for providing smart services in an ITS. We identify the key software service requirements of our proposed microservice-oriented modelling for smart transportation and analytics [37,38].

We classify them into two broad categories: (i) Functional microservice performance requirements and (ii) non-functional microservice performance requirements. The functional microservice performance requirements consist of core microservices for modelling a unified data processing platform for an ITS to integrate different subsystems and share information among its stakeholders. SOA technology combined with big data are applied to integrate heterogeneous information systems from different transportation departments including internal and external stakeholder domains. On the other hand, the non-functional microservice performance requirements consist of microservice features that can support ITSs to enable high service levels of safety, performance, and availability of systems in enhancing stakeholder satisfaction. We provide a detailed overview of these functional and non-functional microservice performance requirements of our microservice-oriented modelling approach for a smart transportation and analytics system.

3.1. Functional Microservice Performance Requirements of ITSs

We identify four main categories of the functional microservice performance requirements of a typical ITS as summarized below:

3.1.1. Big Data Lake for Data Pooling

Data collected from disparate sources as silos need to be securely unified under one platform for useful collaboration [39,40]. Data Lake refers to a pool of storage which can be used for storage of data under multiple stages of processing in an ITS. It stores raw data (structured, semi-structured and unstructured), pre-processed data after Extract–Transform–Load (ETL), Just-In-Time (JIT) data processed for specific use cases, results of experiments conducted, and other support data. Due to the different data sources, the data displays multidimensional heterogeneity as it involves different data representations, data uncertainties and various data formats. Therefore, the data extraction, data formatting, and data filtering are important for data pooling to form the Big Data Lake. The Data Lake must be governed by a set of principles and design patterns during the stages of ingestion, processing, storage, and consumption so that it does not end up as a Data Swamp.

3.1.2. Real-Time Reporting Tools of Useful Metrics and Visualization for Data Insights

The big data system should not only enable real-time reporting of the current condition of various channels of various transport systems, but also allow complex processing of historical data to generate value from it [41,42]. Therefore, in this regard, the functional microservice performance requirements can be divided into two types based on their transport service time. The first type of service metrics/use cases are those that need to be reported in the lowest possible latency. Congestion on roads and bus delays are examples of metrics of this type. The other type of service metrics/use cases are those whose values do not diminish if not reported immediately. For example, daily or weekly public bus transport's timeliness report is a use case of this type. In this paper, we discuss how some of these metrics can be implemented through a hybrid architecture of both streaming and batch processing to deliver the required smart transportation services. Visualization of both types of service use cases is another key requirement. Such visualizations would assist authorities for making decisions effectively.

3.1.3. Predictive Modelling of Transportation Services

Past and current data accumulated over time are useful in descriptive (what) and diagnostic (why) analytics of transportation services. The predictive analytics help further in providing intelligence using ML and deep learning (DL) algorithms and captures patterns from the current data distribution to predict future values so that transport related decisions can be made in real-time [43,44]. Demand forecast of a public transport is an example of predictive modelling of transportation services which can help public transport providers to effectively make use of vehicle and manpower resources to meet the user needs.

3.1.4. Data Integration and Insights using Application Programming Interface (API)

Data insights gained from various reporting features that serve as use cases must also reach relevant end-users of an ITS. The system must expose transportation service-related data insights as endpoints which can then be consumed by other third party APIs from the cloud environment. Exposing data insights to the public consumers and stakeholders of an ITS would help to make smarter decisions. This requires the software components and interfaces to integrate applications using a mix of data sources, thereby accelerating the speed of developing services for smart transportation and analytics [37,39].

In general, microservices offer benefit such as faster change speed, better selective scalability, and cleaner, evolvable architecture.

3.2. Non-Functional Microservice Performance Requirements of ITSs

Typically, ITSs should be capable of scaling horizontally and handling large amount of traffic coming from several heterogeneous data sources such as IoT devices, smartphones, cameras, machine logs, and social media [38,45]. There are ten non-functional microservice performance requirements in the context of designing contemporary data architectures for intelligent systems: Batch data, stream data, late and out-of-order data, processing guarantees, integration and extensibility, distribution and scalability, cloud support and elasticity, fault-tolerance, flow control, and flexibility and technology agnosticism [38]. Conventional big data systems that rely on MapReduce jobs and batch ETL cannot be used because of high latency in ingesting new data [46]. Our proposed microservice-oriented architecture for ITSs should be capable of ingesting data continuously and support real-time processing of big data. The system should be robust, highly available and fault tolerant without losing incoming data. In addition, security measures should preserve user privacy while dealing with data which can potentially identify the user. For instance, data generated from a smartphone of a user must be anonymized to protect sensitive and private information. Further, the system should establish secure policies for granting access rights and roles to authorized users who are eligible to access data. We identify and summarize below seven key non-functional microservice performance requirements in providing useful metrics for smart transportation and value-added analytics in efficient use of the roads and public transport systems.

3.2.1. Public Transport Service Reliability

Many studies have emphasized that reliability of transport services is a key determinant of passenger ride satisfaction and strongly shapes people's choice in transportation modes. Reliability measures include not only absolute waiting time but also headway of the transport system. Reliability is more significant in the intra-city public transportation context where most bus services are of the high-frequency category where buses arrive at frequencies of less than 15 min. Headway is measured as an interval of time between bus arrivals and is the commonly used metric to specify the bus regularity or reliability [5,6]. In general, for such bus services, passengers do not rely on a timetable of scheduled arrival times and; therefore, bus arrival punctuality is not primarily a passenger concern. More importantly, expectation of passengers to "turn up and go" and any unexpected increase in waiting time are important in assessing the passenger satisfaction of the transportation service. To improve reliability, bus service controllers are required to assist in reducing bus bunching as it increases headway [47,48]. Based on reliability metrics, the transport service providers can arrive at transport policies and traffic governing measures, such as enforcing appropriate bus travel speed, providing advisories on bus stop dwell time, inserting stand-by buses in mid-route, or replacing a turnaround bus with a new one.

We provide an example measure for public transport service reliability such as excess wait time (EWT), which is the difference between actual wait time (AWT) and scheduled wait time (SWT). These reliability metrics can be calculated using the accumulated data at different instances n related to actual headway (A_n) and scheduled headway (S_n). For instance, bus drivers and service operators could be provided with incentives based on achieving certain performance metrics. The formula for the calculation of EWT, AWT and SWT are given below:

$$\text{EWT} = \text{AWT} - \text{SWT}$$
$$\text{AWT} = \frac{\sum_n A_n^2}{2 \times \sum_n A_n} \qquad \text{SWT} = \frac{\sum_n S_n^2}{2 \times \sum_n S_n}$$

Assuming a uniform arrival rate of passengers, EWT increases when there is bus bunching. EWT is zero if a bus service arrives exactly according to the scheduled headway. These metrics can be defined to bring out the complexity of the traffic environment and gain deep insights. In other words, a measure such as EWT can be measured in multiple ways from different data inputs such as peak period, non-peak period, across all trips, at critical bus stops, average across calendar month, and trip direction.

3.2.2. Seating Availability in Public Transport

In a public transport system, the available seating capacity is a concern for commuters who depend on such mode of transport every day. During peak hours, commuters may prefer to use taxis rather than overcrowded buses/subways [21,39]. A smarter decision of whether to use public transport can be made if the traveler has access to information on the available seating capacity at a time and the chance of getting a seat for bus/subway for taking the journey. Crowdedness of buses and subways can also help elderly and people in need of more space like parents travelling with infants/kids to make a smarter decision when to opt for a more comfortable travelling alternative than public transport. For instance, the total occupancy (TO) and crowdedness metric (C) of a bus or subway can be calculated as follows:

$$TO = O + B - E \text{ and } C = \frac{TO}{M}$$

where O denotes number of occupied seats, B denotes number of passengers boarded into the bus, E denotes number of passengers exiting the bus, and M denotes maximum capacity of the bus.

3.2.3. Public Road Congestion

Reduction in traffic congestion has economic, environmental, and health benefits. Firstly, reducing peak-time congestion saves both travel time as well as capital investment on building additional roads or highways. Further, reduced vehicle wait times in traffic jams reduces vehicle exhaust, thus reducing carbon emissions and air pollution. In the US alone, 25 million tons of carbon dioxide per year was emitted from vehicles stuck on congested roads [12]. With big data providing high-resolution GPS information, such data could be used to build public transport demand maps to enable better allocation of public transport resources. Various ridership trends and passenger routes can be processed to identify areas of high demand so that public transport resources could be redeployed accordingly. Such public transport resource optimization aim to minimize wait-times, increase transport efficiency, and encourage uptake of public transport, thereby reduce the volume of private vehicles in improving public road congestion [42,49]. We provide two metrics for determining the public road congestion such as travel time index (TTI) and duration of traffic congestion (DTC) below:

Travel Time Index

Travel Time Index (TTI) is a measure which compares peak period travel and traffic-free or free flow travel, while accounting for both recurring and incident conditions. The index has the advantage of expressing traffic congestion in terms of both space and time as follows:

$$TTI = \frac{T_p}{T_f} = \frac{S_f}{S_p}$$

where T_p denotes travel time during peak period, T_f denotes travel time free flow of traffic, S_f denotes travel speed of the bus during peak period, and S_p travel speed of the bus during free flow of traffic.

Duration of Traffic Congestion

Another measure that provides useful information on public road congestion is the duration of traffic congestion [9]. To determine this metric, the first step involves smoothing the instantaneous fluctuating vehicle velocity using the technique that is commonly used in the Transport Control Protocol (TCP) for smoothing out the average flow throughput on the Internet. This technique is called weighted exponential average velocity that can be adopted for vehicle traffic on public roads and can be expressed mathematically using data collected at different times. In this context, we adopt the following mathematical equation:

$$V_t = \alpha \cdot S_t + (1 - \alpha) \cdot V_{t-1}$$

where V_t denotes the average vehicle velocity at time t, S_t is the α percentage of instantaneous velocity at time t and V_{t-1} denotes $(1-\alpha)$ percentage of the average velocity at previous time $(t-1)$.

In our experimental study, $\alpha = 0.125$ is used, which is a standard value used by TCP. After the average velocity is obtained, the second step classifies it into three levels—red, yellow, and green—and two classification thresholds, β and γ, are determined as follows:

- Red level, if $V_t < \beta$ km/hr.
- Yellow level, if $\beta < V_t < \gamma$ km/hr.
- Green level, if $V_t > \gamma$ km/hr.

After the initial congestion level is determined, the congestion duration is calculated according to the time in which each congestion event remains in the same state. The third step determines the final congestion levels. A reported congestion event is assumed to last for at least ±180 s, which is determined based on mining of historical data. Experimental study should be conducted until the final congestion level obtained is compatible with the current system and ready to be reported to the public.

3.2.4. Public Transport Queuing Information

In densely populated cities, both situations where passengers queuing for taxis and taxis queuing for passengers frequently occur due to an imbalance of taxi supply and demand. Timely and accurate detection of such waiting queues and the queue properties would immensely benefit both public commuters and taxi drivers [40]. There are four types of queue contexts (C_1, C_2, C_3, and C_4) given in the following table that play a role in the context-aware data processing:

In Table 1, Taxi Queue refers to one or more available taxi waiting for taking new passengers, and Passenger Queue refers to one or more passenger waiting for taxis at a queuing area or taxi stand during a given period. Big data mining of various parameters associated with this queue information would help in providing appropriate advice for commuters and transport operators for an improved service.

Table 1. Passenger and transportation queues.

Queue Type	Passenger Queue	No Passenger Queue
Taxi Queue	C_1	C_3
No Taxi Queue	C_2	C_4

3.2.5. Monitoring Public Transport Incentives

One of the non-functional microservice performance requirements for continuously improving transport services is by providing context-aware rewards and incentives and monitoring their effectiveness [10,31]. However, there could be missing real-time headway metrics arising from missed real-time location data and messages from vehicles [40]. Such data may need to be imputed for determining the incentives for service reliability. For instance, the real-time headway metrics of a public transportation service could be imputed as an average over different groups of data available under certain criteria such as time (e.g., weekly, monthly, etc.), peak or non-peak period, type of bus services, single or both directions, and specific critical bus stops to create the overall reliability indicators of a bus service.

3.2.6. Predictive Demand Forecast

Predictive analytics could be employed to forecast the demand for public transportation services. A widely used statistical method for time series forecasting is autoregressive integrated moving average (ARIMA) that can be developed to predict the demand for public bus services from previously boarded passenger count [43]. The ARIMA model could be built in the batch layer, and the most recently boarded passenger count for the bus service will be used in the real-time stream layer to predict future demand of the bus service. An ARIMA model is represented by the equation given below:

$$\hat{y}_t = \mu + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q} + \mu_t$$

where y_t is the difference in boarded passenger count at time t, the moving average parameters are represented by $\theta_1,\ \theta_2, \ldots, \theta_q$ and $\phi_1, \phi_2 \ldots\ , \phi_p$ are the autoregressive parameters determined by the fitted model. Other model based on neural networks such as the recurrent neural network (RNN) and the long–short term memory model (LSTM) could also be used to for predictive analytics on time series data.

3.2.7. Information for Traffic Enforcement

In many countries, traffic enforcement in an ITS makes use of images and video-based intelligent systems that are instrumental in traffic analysis and traffic management [50,51]. Typically, traffic analysis includes information on vehicle count, and other image and video-based parameters for traffic state recognition and automatic incident detection. Traffic management involves placing closed loop sensor systems to track vehicles and their usage of different locations of the road such as temporary hard shoulders to control their access using traffic signals. All these rich traffic data from various images and videos collected from speed cameras will help the traffic enforcement for public roads and transportation systems. Such data can be used by ITSs from traffic analysis to traffic management and then to traffic enforcement of average speed controls, traffic light system, and toll collection at various locations of the road. An integrated ITS with service-driven design is required to increase the reliance and accuracy from the images and video sources for different transport modes. For instance, public bus service could be enhanced using the images and videos of buses taken at different bus stops and locations of the road. An accurate and intelligent data computation plays a major role in an ITS such as bus headways for regulatory reporting purposes and prediction of future headways for better bus fleet management. Using machine vision of spatial-temporal traffic data, insightful information, and predictions can determine traffic speed bands of road segments, road traffic congestion, future traffic status, and alerts.

4. Proposed Microservice-Oriented Big Data Architecture (MOBDA) for a Smart Transportation and Analytics System

Driven by the user-centric requirements in the design of an ITS, we propose Microservice-Oriented Big Data Architecture (MOBDA) as a hybrid model framework that leverages both stream processing and batch processing for achieving a smart transportation and analytics system. By dividing the metrics and use cases into two popular categories of data processing architectures, as discussed in Section 3, the system can exploit low latency proficiencies of Kappa and high throughput of Lambda, as shown in Figure 1.

4.1. Overview of Proposed MOBDA

The proposed MODBA relies on a distributed message broker backend to ingest data, communicate between microservice-oriented components, and exchange messages to share data processing models, data analytics, and insights. The processing framework will also provide querying, pattern recognition, and text analytics facilities made available to the serving layer. The architecture derives its computations from both online and offline computation models depicted as the real-time processing layer and batch layer, respectively, in Figure 1. The real-time processing layer can produce output in the lowest possible latency to compute metrics that can be calculated incrementally and does not rely on time-consuming complex CPU/GPU based processing. Real-time processing covers dynamic dashboards and quick analytics. On the other hand, batch processing architecture is used to process data in batches to gain higher throughput. Training of complex ML models, feature engineering, and data transformation for reporting from analytics apps will be performed in this layer.

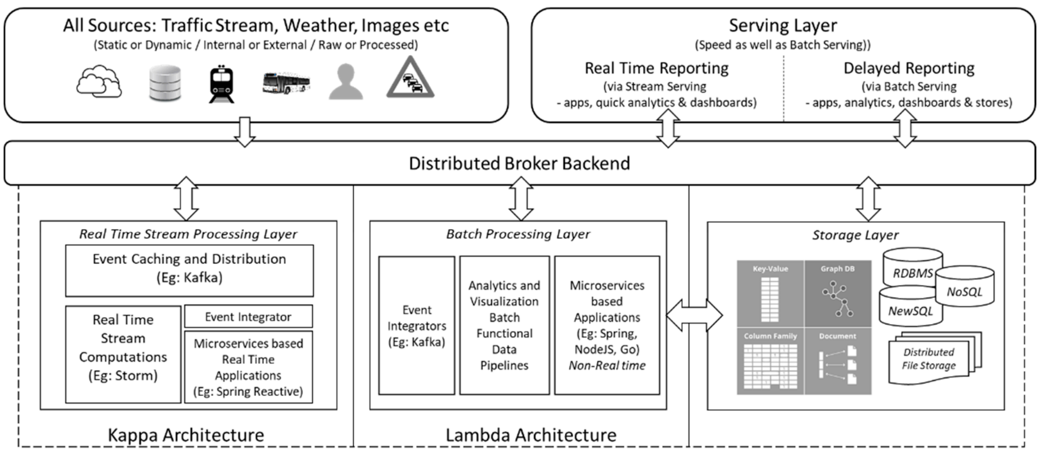

All Sources: Traffic Stream, Weather, Images etc
(Static or Dynamic / Internal or External / Raw or Processed)
Serving Layer
(Speed as well as Batch Serving))
Real Time Reporting
(via Stream Serving
- apps, quick analytics & dashboards)
Delayed Reporting
(via Batch Serving
- apps, analytics, dashboards & stores)
Distributed Broker Backend
Real Time Stream Processing Layer
Event Caching and Distribution
(Eg: Kafka)
Real Time Stream Computations (Eg: Storm)
Event Integrator
Microservices based Real Time Applications (Eg: Spring Reactive)
Batch Processing Layer
Event Integrators (Eg: Kafka)
Analytics and Visualization Batch Functional Data Pipelines
Microservices based Applications (Eg: Spring, NodeJS, Go) Non-Real time
Storage Layer
Key-Value
Graph DB
Column Family
Document
RDBMS
NoSQL
NewSQL
Distributed File Storage
Kappa Architecture
Lambda Architecture

Figure 1. Overview of proposed Microservice-Oriented Big Data Architecture (MOBDA).

The main motive behind having a hybrid between stream and batch processing is to achieve performance and service level efficiency in an ITS. Performing computation intensive big data operations such as joins, grouping, sorting will increase the reporting latency of the streaming layer for arriving at transport service level metrics. Additionally, we show that a clear majority of metrics related to an ITS can be calculated by handling the incoming data as a continuous stream and applying intelligent window functions. The core technologies used in each of the layers is discussed in further sections.

Today, an ecosystem of open source tools and cloud-based services make it possible to develop decomposed and distributed microservices in a cost-effective way. In the proposed MOBDA, we off-load the cost of dealing with some of the repeated batch and real-time processing pipelines to effective microservices. While this results in several advantages, the most significant advantage in big data context is the ability to run selected stateless processes as microservices thereby making it easier to scale across distributed clusters. MOBDA aims at identifying components from the existing ITS functional pipelines that can shift towards self-contained, independently deployable high performing microservices that can bind to a port. The goal is not to build microservices that are as small as possible like in traditional microservice based solutions. Instead the primary aim in our proposal is to achieve process optimization through bounded right-sized microservices. Consequently, in the proposed architecture, our reactive microservice layer works in-tandem with the event integrator to provide real-time interactive interfaces for the hybrid architecture. It does not replace the real-time stream processing pipelines that were built using frameworks such as Apache Storm. Similarly, microservice-based applications will also supplement traditional batch processing pipelines by offering domain driven decomposition of intelligent transport services. Hence, through the use of our microservices layer over the data pipelines, we are able to achieve non-real-time transport services that are loosely coupled, query-based, and independently deployable, and these form critical considerations for the adaptability and scalability of big data solutions.

The input sources of big data in its ingestion phase can be broadly categorized into two major types as follows:

- Internal sources—This refers to data originating from each transport organization and its services. A collection of these data should have started with the associated service, and hence will have a historical data such as GPS locations, passenger entry and exit of bus, commuter usage on mobile apps, camera feeds from bus, seat availability, bus routes, etc.
- External sources—This refers to data obtained from third-party sources. This data does not originate from the transport organization but are useful as complementary sources in the data analysis. External data sources could be from weather bureau, public events, bus and taxi stand cameras, real-time traffic information of Google Maps, etc.

Figure 2 presents a high-level flow of data from the big data input sources to the operational and edge servers of MOBDA. The arrows from various input sources indicate the different datasets ingested from both external and internal sources to various publishers. The edge/gateway node is an independent server that acts as a connection between the external sources and the actual big data cluster in offering many microservices. In Figure 2, we consider Weather as an example of a microservice originally exposed via API endpoints from the government or a private company. The edge node periodically calls the API (e.g., via CRON), does the required pre-processing (e.g., converting XML/JSON to flat file), and produces a message in the Kafka topic. The difference with the internal sources is that they are handled by the Operations Server. The dotted box in Figure 2 clearly shows the components of this Operations Server, wherein the distributed data services and endpoints already in place for the typical create–read–update–delete (CRUD) operations as well as related to data streaming topics. An additional service called KafkaMsgToTopic converts the data into a message and passes it along the specific Kafka topic, and will be useful in creating microservices meeting the requirements of different users for a scalable system. Each Kafka topic is a category or feed name to which data records belonging to that topic are organized, stored, and published. Figure 2 shows along the arrows the partition of topics indicating the data under the specific topics communicated from KafkaMsgToTopic Service in parallel to Kafka broker.

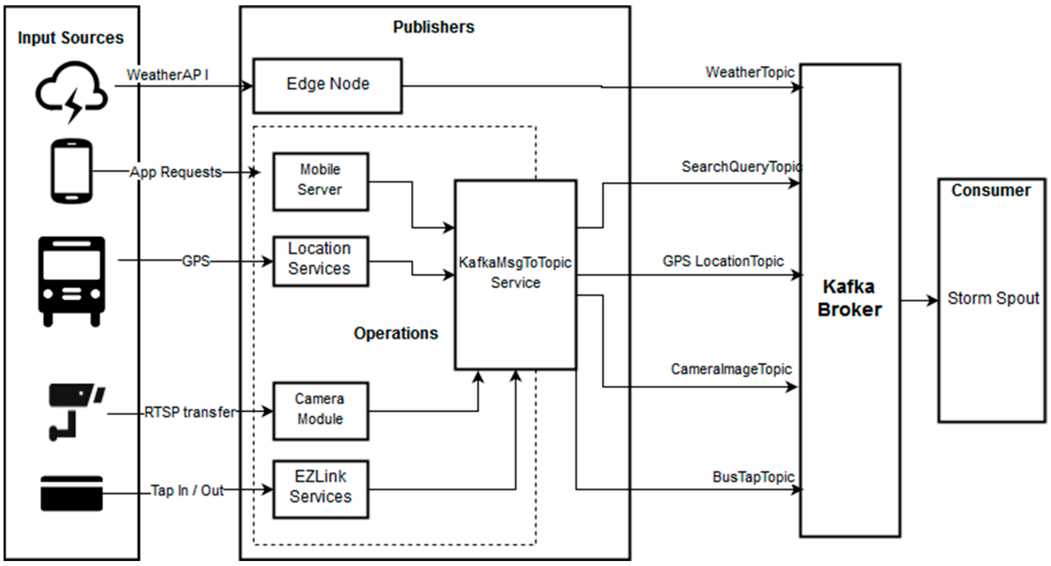


Figure 2. Data flow model of MOBDA from various big data input sources.

4.2. Real-Time Stream Processing Layer

The foundation of our approach to real-time stream processing layer is the stream processing speed layer using Kappa architecture. An ITS generated data being dominantly time-series data with rapidly changing contexts, Kappa stream processing frameworks such as Flink, Storm, and Spark provide processing pipelines using data extraction libraries, task/pipeline parallelism, functional scalability, high computational power, finite data-retention rates, data compression support, and relevant specific time-series analytics.

In the real-time stream processing layer of MOBDA, a Storm cluster is used with an architectural perspective shown in Figure 3. It is composed set of nodes called Worker node running the daemon called Supervisor. The cluster also hosts one master node, which runs the daemon called Nimbus. The Nimbus is responsible for assigning work and managing resources in the cluster [15]. Figure 3 provides a pictorial illustration of a high-level stream processing of MOBDA by adopting Apache Storm cluster to complete the assigned tasks with real-time data in the best possible way. The Nimbus daemon distributes code to get the tasks done within the cluster and monitors for failures using the Apache ZooKeeper. Using Figure 3, we demonstrate high-level topology as an example of Zookeeper employing three Worker nodes (Supervisor) to execute the Worker processes. The double arrows show the mutual reliance of these components. The single arrows show the communication among the Worker processes that collaboratively complete the assigned tasks and update their status in the Apache ZooKeeper service. This topology is useful to illustrate how the microservice tools can be used to store Nimbus states in Apache ZooKeeper and monitor the state to restart Nimbus from where it left in the case of any failure. Hence, from the perspective of stream processing, Storm is scalable and fault tolerant, and can be configured to deliver a different level of guarantee in processing tuples and the three semantics employed are at-most-once, at-least-once, and exactly-once to enhance reliability.

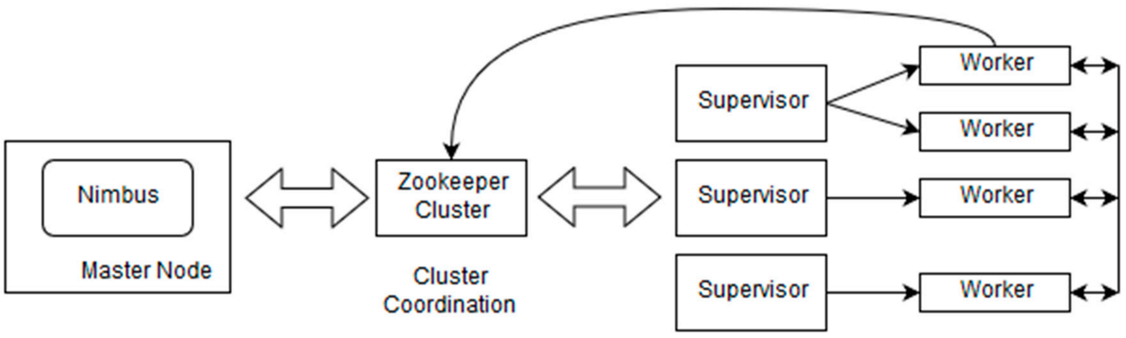


Figure 3. Stream Processing of MOBDA using Storm Architecture.

For ITSs, many data sources, like IoT sensors, smartphones, cameras, etc., emit data continuously, making Apache Storm a natural choice for processing the incoming stream [28]. In addition, as discussed in Section 3, most microservices offering reporting of traffic-related services require certain metrics either calculated instantaneously or based on a windowed time frame. These metrics do not rely on historical data, thus eliminating the need to maintain a central cache of historical data for Storm bolts to access. Additionally, seamless Kafka integration using Kafka spouts enables easy integration of microservices with the proposed architecture.

We propose a Storm topology for MOBDA, as shown in Figure 4, to compute the transport service metrics discussed in Section 3. Kafka spouts, which emit the incoming Kafka messages (curly arrows) as a stream of tuples, act as the starting point for calculating the metrics (shown in processing bolts as circles). Then each of the spouts release data to a corresponding bolt which transforms the stream using a valid stream operator. In a few cases, these bolts rely on the saved state to access historic or factual values (denoted by dotted links). The bolts can also utilize pre-trained models, which can be applied to transform the stream data. For instance, the Object Detection bolt can apply previously trained DL model to the incoming stream of images to count the number of people/taxis waiting in the taxi stand. These pre-trained models can be either loaded to memory or accessed via an API call. Finally, the final bolt in each of the computation emits the desired metric which is published to Kafka queue. The published messages are consumed by another set of spouts and bolts which are responsible for persistence committed to the storage layer. Figure 4 serves as a design model that is useful for the implementation of ITS metrics in the Apache Storm topology of MOBDA.

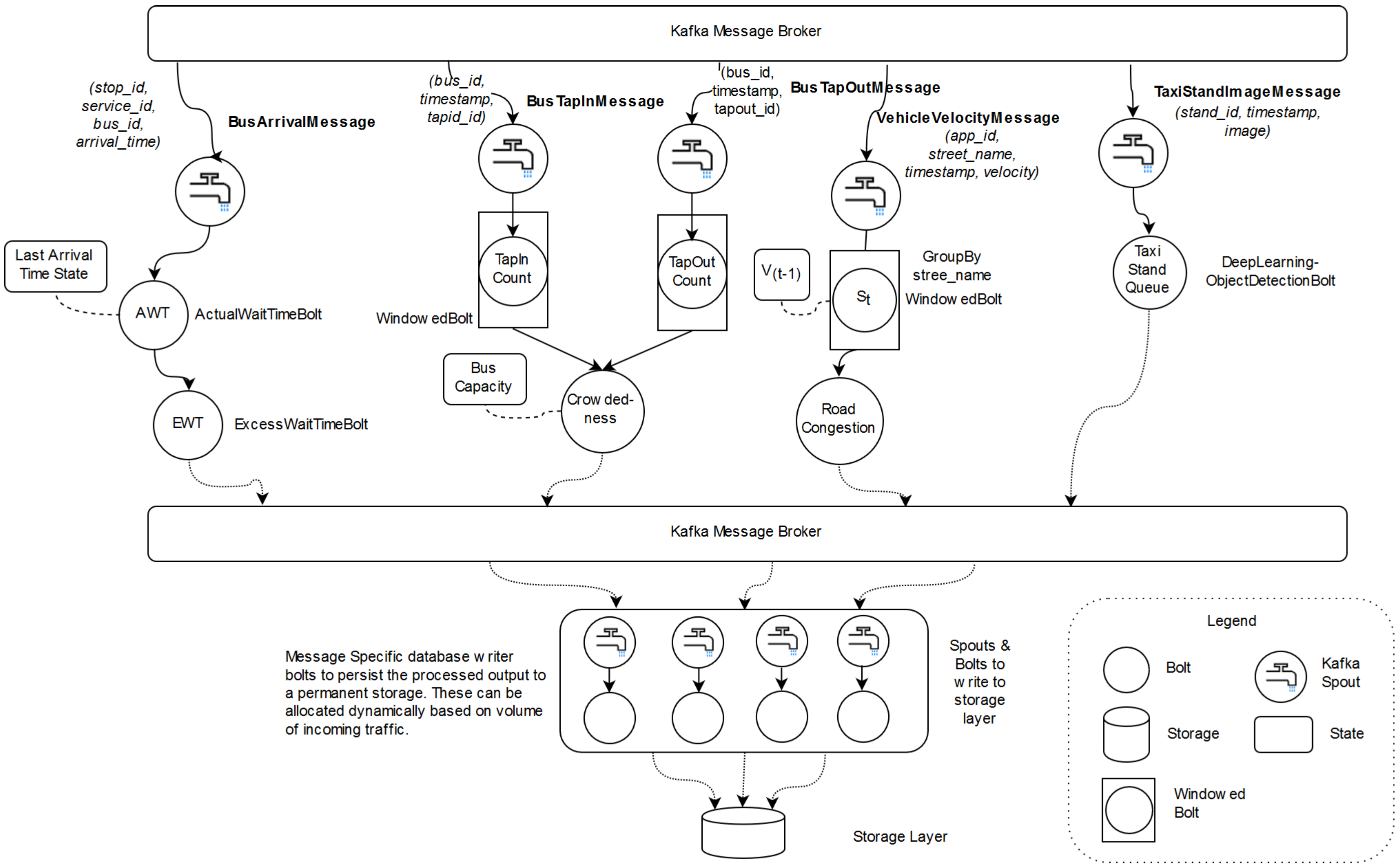

Kafka Message Broker
(stop_id, service_id, bus_id, arrival_time)
BusArrivalMessage
(bus_id, timestamp, tapid_id)
BusTapInMessage
(bus_id, timestamp, tapout_id)
BusTapOutMessage
VehicleVelocityMessage
(app_id, street_name, timestamp, velocity)
TaxiStandImageMessage
(stand_id, timestamp, image)
Last Arrival Time State
AWT
ActualWaitTimeBolt
EWT
ExcessWaitTimeBolt
TapIn Count
Window edBolt
TapOut Count
Bus Capacity
Crow ded-ness
V(t-1)
St
GroupBy stree_name Window edBolt
Road Congestion
Taxi Stand Queue
DeepLearning-ObjectDetectionBolt
Kafka Message Broker
Message Specific database w riter bolts to persist the processed output to a permanent storage. These can be allocated dynamically based on volume of incoming traffic.
Spouts & Bolts to w rite to storage layer
Storage Layer
Legend
Bolt
Kafka Spout
Storage
State
Window ed Bolt

Figure 4. Overview of Storm topology of MOBDA for real-time transport service metrics.

4.3. Batch Processing Layer

The batch processing layer is aimed to provide the delayed reporting of transport service metrics and the visualizations of the data insights. For this layer, we adopt Apache Spark, an open-source distributed general-purpose cluster-computing framework, which is gaining popularity due its speed and adaptability with interactive queries and iterative algorithms. Spark is faster than Hadoop MapReduce as it uses in-memory processing, which is about 100 times faster with data in RAM and about 10 times faster with data in storage [46]. Spark can be adopted with the Akka framework for creating reactive, distributed, parallel and resilient concurrent applications.

The Spark Core's version of MapReduce allows in-memory computing, caching, and chaining multiple tasks together. It has the advantage that offline descriptive and diagnostic analytics (such as weekly or monthly reporting microservices) are simple to implement. Spark SQL is a module in Apache Spark that integrates relational processing with Spark's functional programming API. Further, Spark SQL Engine allows combining batch and stream processing using the new structured streaming approach. Spark MLlib is a collection of distributed ML algorithms that utilizes large-scale historical data for batch processing and is useful in the development of models for predictive analytics in an ITL.

4.4. Storage Layer

The storage layer needs to meet all the functional and non-functional microservice performance requirements which differ particularly in scalability and real-time accessibility of big data. Real-time metrics which are used for transport operational control purposes require a storage solution which supports quick database reads and writes. For management reporting, the storage layer requires more complex query support with less need for fast-transactional response or real-time analytics. A polyglot persistence architecture is best able to meet these differing storage requirements, albeit the cost of increased implementation complexity [16]. More recently, multi-model datastores have appeared in the NoSQL landscape and they hold the promise of a homogeneous single-system implementation of polyglot persistence, which significantly reduces the implementation complexity [52]. Such software components can provide scalability to meet a wide range of storage layer requirements in our proposed MOBDA for an ITS.

In the polyglot storage layer, raw data from the data sources are stored directly into a scalable HDFS. Processed data from the real-time stream processing layer and the batch layer is stored in Cassandra, which is used as a distribute column-based database to allow for different consistency levels and to support a large volume of concurrent reads and write with a low latency. In addition, MOBDA makes use of Cassandra query language (CQL) for the formulation of efficient and powerful queries into the data storage layer to support management reporting microservices. Cassandra, which is fault tolerant and highly available, can be connected to Apache Spark and Apache Storm easily through connectors. Cassandra's data model is an excellent fit for handling data in sequence, making it a preferred choice for storing the calculated transport service level metrics which are time dependent. Incoming image stream which is raw files is simply stored into the HDFS. A meta-data table to store the details about the traffic images and videos like the place, timestamp, and location in HDFS is maintained using Cassandra.

4.5. Broker Layer

The broker layer of MOBDA can be considered as the meta-data layer for data communication, security, and governance by enforcing the required business rules of an ITS. Frameworks such as Apache Atlas and HCatalogue are helpful in a successful implementation if this layer. It plays an integral part of MOBDA as many decisions are made based on the various transport data that needs to be validated and protected from unintended parties. In addition, big data security and governance are required to ensure privacy of individuals and safety of commuters. This poses restrictions on data that can be captured, and every data collected needs to be accounted for and tracked across various organizations in achieving the required transport service levels. Data lineage of MOBDA includes a data cycle that captures data visibility across all stages of the pipeline and every action is traced back to its source. This helps in troubleshooting and recovery of middle stage data by running only from the affected parts of the pipeline. Figure 5 lists four essential components of the security broker layer of MOBDA for managing sensitive data [36]. By separating these security broker components, the business rules for each component could be independently adopted using best practice methods. Data classification or tagging allows one to identify the source where data was obtained, thereby custom security checks can be applied to the source depending on vulnerability. Fine-grained access control policies are to keep a check on the user access to authorize only the right user(s) to have limited access to the right data. Data anonymization, encryption, and differential privacy mechanisms protect public information. Finally, monitoring options are included for audit purposes.

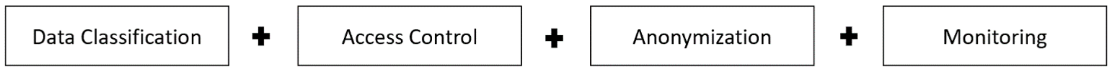


Figure 5. Security broker components of MOBDA.

4.6. Serving Layer

The serving layer includes both batch and real-time data processing. The former involves querying the stored data at rest, while the later stores queries that works on moving data. The serving layer is responsible to meet the functional microservice performance requirements as discussed in Section 3. Typically, it should satisfy such functional microservice performance requirements originating from two different end-user groups. As shown in Figure 6, the first group of users are the administrative staff, decision makers and data analysts who would rely on computed metrics and generated reports for big data insights and to drive important decisions. This group is mostly interested in real-time visualization of the metrics, easy access of the generated reports and capability to perform analytical processing either by querying the data or through business intelligence tools like Tableau, Power BI, etc. To monitor the metrics in real-time, the messages published by Storm topology of MOBDA can be directly subscribed with appropriate secure access rights and authorization privileges. The metric can be displayed in a dashboard (using microservice built using popular languages like Python or Java). To extract big data insights directly from moving data or perform complex querying like joining message stream, Kafka provides a familiar SQL like API called KSQL. The other group of users which comprises of common customers or third-party app developers will not have direct access to the system and would rely on the exposed API endpoints to consume the data. The microservice application responsible for exposing the desired service would host its own data caching and persisting mechanisms.

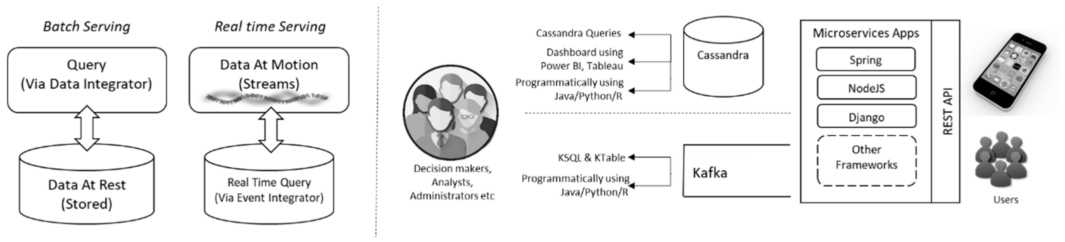

Batch Serving
Real time Serving
Query (Via Data Integrator)
Data At Motion (Streams)
Data At Rest (Stored)
Real Time Query (Via Event Integrator)
Decision makers, Analysts, Administrators etc
Cassandra Queries
Dashboard using Power BI, Tableau
Programmatically using Java/Python/R
Cassandra
KSQL & KTable
Programmatically using Java/Python/R
Kafka
Microservices Apps
Spring
NodeJS
Django
Other Frameworks
REST API
Users

Figure 6. Serving layer of MOBDA using Apache Cassandra and Kafka.

Overall, Figure 6 illustrates integrating systems connecting different BI tools, NoSQL stores like Cassandra and Kafka via representational state transfer (REST) architecture, REST API within the Serving Layer of MOBDA. In many ITS scenarios, the flow of data from various IoT devices is continuous but has limited capacity to buffer data. The use of Kafka microservices allows additional events to be added to the system from a simple application to buffer data in Cassandra to advanced intelligent applications that performs real-time stream processing for business decisions in ITSs.

5. Use Case of MOBDA for a Smart Transportation and Analytics System

5.1. Case Scenario and Method

In many densely populated countries, ITSs have been pivotal in enhancing their public transport systems, as well as in managing and optimizing the limited road space to deliver a convenient, safe, and comfortable travelling experience for its commuters. Behind these systems, there exists a need for big data processing capabilities. Therefore, in this section, we explore use case of our proposed MOBDA for a smart transportation and analytics system to process a large-scale transportation data in the context of a country such as Singapore [1]. We consider one use case of MOBDA for the scenario of determining bus service headway, which is a key service level metric used for determining many of the above smart and timely information on transportation.

The data for exploring the use case of our proposed hybrid architecture was obtained from the Singapore Land Transport Authority (LTA) DataMall. The LTA provides API access for various types of transport related data and microservices in Singapore [5,20]. Figure 7 provides a pictorial overview of a combination of actual real-time streaming of traffic data downloaded from public dataset of LTA and simulated data related to public bus transport system. Hence, it gives an illustration of the main types of transport related data that was collected as input data stream that we processed using our proposed MOBDA model. We mapped some of the key metrics/use cases for transport service level requirements discussed in Section 3, with the big data source and the architecture type in Table 2. The following gives a list of key components and microservices of transportation subsystems:

- Alternative dynamic route recommendations
- On-demand public bus service
- Vehicle navigation service
- Traffic signal control systems
- Variable traffic message signs
- Automatic number plate recognition
- Automatic incident-detection
- Real-time parking lots availability
- Regular and timely updates for users

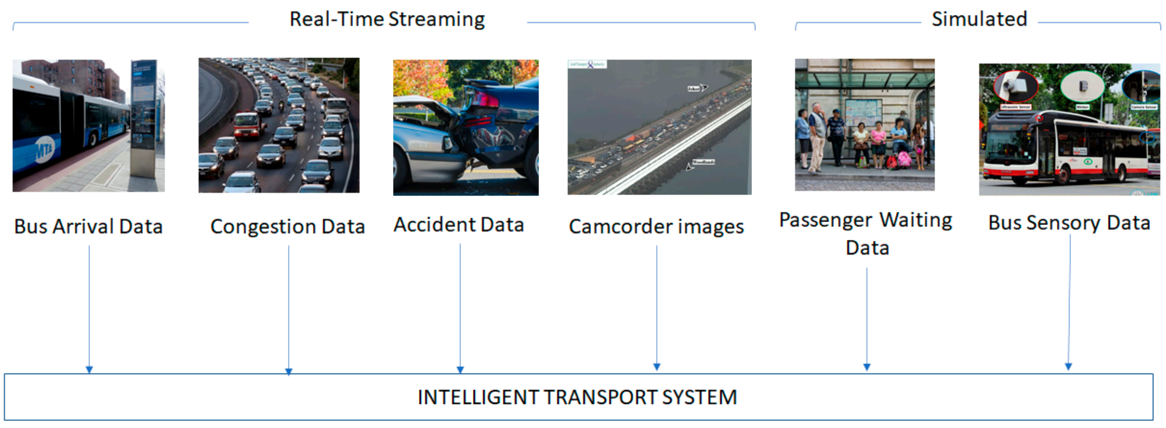


Figure 7. Input data stream for use case of MOBDA.

Table 2. Mapping of key transport service requirements and big data processing architecture.

Transport Service Level Metric/Use Case	Big Data Source	Big Data Processing Architecture
EWT and AWT	Bus arrival	Stream
Available Seat Capacity and Crowdedness	Bus passenger count, image stream from cameras	Stream
TTI (Travel Time Index)	Driver's GPS using smartphone apps	Stream
Velocity Measurement	Driver's GPS using smartphone apps	Stream
Bus Stand Queue	Image stream from surveillance cameras	Stream
Bus Timeliness Report Generation	Historical EWT data	Batch
Predictive Analytics to Forecast the Demand	Historical data	Batch
Intelligent Systems using Deep Learning	Historical data stored image/video stream and records	Batch

For exploring the use case of MOBDA, we determined the bus service headway as an illustration, with real-time bus arrival information for bus services at a specific bus stop. To achieve this, a sample of 50 bus stops were selected among more than 5000 existing bus stops to obtain the bus arrival data. In this section, we provide the implementation details of various layers of MOBDA such as the data ingestion at the storage layer, real-time data, and batch processing layer and the serving layer with results. Table 3 provides a summary of the technologies used for big data processing in different layers of MOBDA.

Table 3. Technologies used for big data processing in different layers of MOBDA.

Big Data Processing/Service Level Reporting	Technology Used
Raw sensor data	Apache Kafka/Storm/Cassandra
Data stream from smartphone apps	Apache Storm/Cassandra/HBase/Hive
Real-time operational metrics	Apache Cassandra/HDFS
Management daily/weekly reports	Apache Spark (with SparkSQL)/HDFS
Captured images and videos	Apache Cassandra/HDFS

The proposed MOBDA was applied using Apache Storm for processing the incoming stream of data to calculate the metrics required by microservices in real-time. The topology of Apache Storm consists of spouts that represent the source of the data stream and bolts that consume the spouts. Together they define the stream processing pipeline, which eventually transforms the stream by applying filtering, aggregations, joining with another stream. Unlike a MapReduce pipeline, which processes the data in batches, the Storm topology processes the data as it enters the system, thereby effectively providing the lowest possible latency. For the data integration from various big data sources, the primary data ingestion tool we employed was Apache Kafka. Using a distributed streaming platform such as Kafka, we could apply MOBDA to handle trillions of transport-related events per day. At the heart of Kafka lies three main components: (i) Pub-Sub (publish and subscribe), (ii) WAL (write ahead log) to ensure persistent commit that can recover from failures, and (iii) processes which are small on-the-fly transformations that can be performed between data transmits.

For the storage layer, different data storage options are chosen such as HDFS and HBase. For the big data querying engine, we chose Hive as a suitable option. The implementation of data ingestion and microservice consumption from Kafka to storage is illustrated in Figures 8 and 9. Ingestion of big data to storage was completed with HDFS for storing unstructured data such as checkpoint images, HBase for storing data of bus arrivals, and Hive for data utilization in the downstream processes. Ingestion of traffic speed bands and bus arrivals are shown in Figure 8. Ingestion of footage from surveillance cameras at bus stands and other checkpoints is shown in Figure 9. The data in HBase was exposed as a Hive table by efficiently making Hive to be in sync with the incoming data inserts without having to explicitly schedule an ETL job to transfer the data.

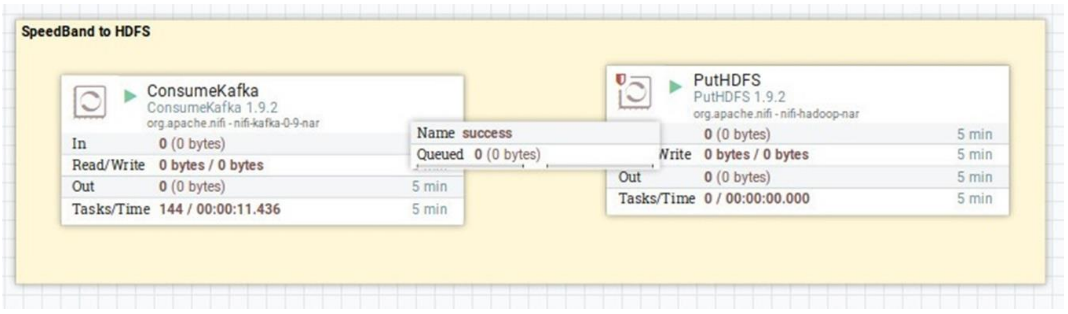

SpeedBand to HDFS
ConsumeKafka
ConsumeKafka 1.9.2
org.apache.nifi - nifi-kafka-0-9-nar
In 0 (0 bytes)
Read/Write 0 bytes / 0 bytes
Out 0 (0 bytes) 5 min
Tasks/Time 144 / 00:00:11.436 5 min
Name success
Queued 0 (0 bytes)
PutHDFS
PutHDFS 1.9.2
org.apache.nifi - nifi-hadoop-nar
0 (0 bytes) 5 min
Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

Figure 8. Data ingestion of MOBDA: Consumption of speedband data from Kafka to storage.

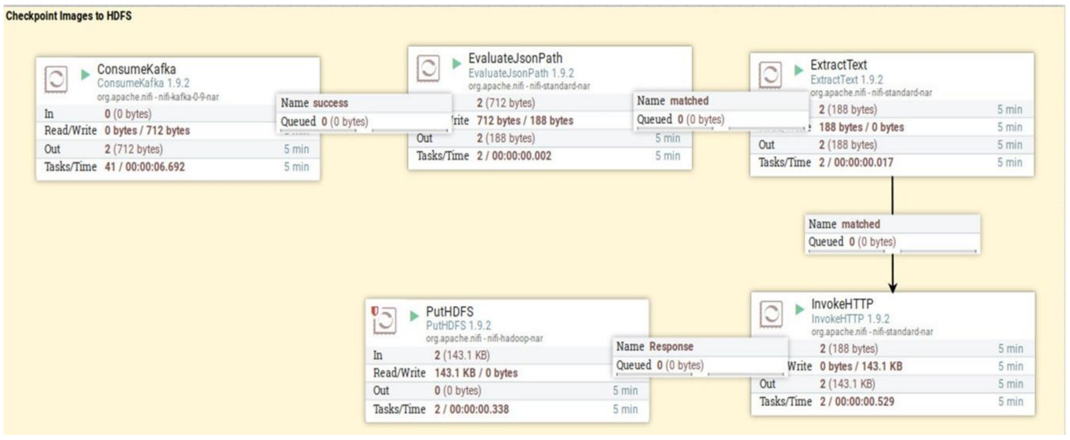
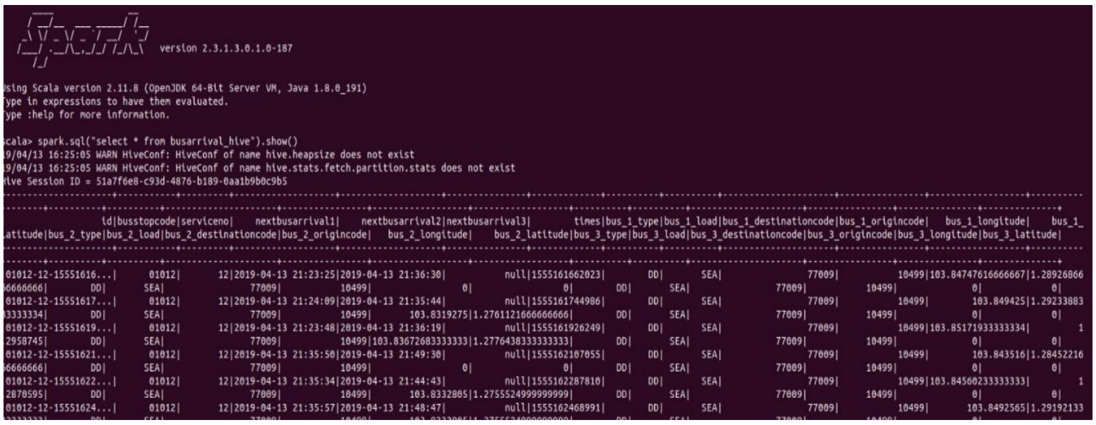

Checkpoint Images to HDFS
ConsumeKafka
ConsumeKafka 1.9.2
org.apache.nifi - nifi-kafka-0-9-nar
In 0 (0 bytes)
Read/Write 0 bytes / 712 bytes
Out 2 (712 bytes) 5 min
Tasks/Time 41 / 00:00:06.692 5 min
Name success
Queued 0 (0 bytes)
EvaluateJsonPath
EvaluateJsonPath 1.9.2
org.apache.nifi - nifi-standard-nar
2 (712 bytes)
Write 712 bytes / 188 bytes
Out 2 (188 bytes) 5 min
Tasks/Time 2 / 00:00:00.002 5 min
Name matched
Queued 0 (0 bytes)
ExtractText
ExtractText 1.9.2
org.apache.nifi - nifi-standard-nar
2 (188 bytes) 5 min
188 bytes / 0 bytes 5 min
Out 2 (188 bytes) 5 min
Tasks/Time 2 / 00:00:00.017 5 min
Name matched
Queued 0 (0 bytes)
InvokeHTTP
InvokeHTTP 1.9.2
org.apache.nifi - nifi-standard-nar
2 (188 bytes) 5 min
Write 0 bytes / 143.1 KB 5 min
Out 2 (143.1 KB) 5 min
Tasks/Time 2 / 00:00:00.529 5 min
Name Response
Queued 0 (0 bytes)
PutHDFS
PutHDFS 1.9.2
org.apache.nifi - nifi-hadoop-nar
In 2 (143.1 KB)
Read/Write 143.1 KB / 0 bytes
Out 0 (0 bytes) 5 min
Tasks/Time 2 / 00:00:00.338 5 min

Figure 9. Data ingestion of MOBDA: Consumption of checkpoint images from Kafka to storage.

5.2. Results

Computation of bus headways are required for regular reporting purposes in different traffic related microservices. Determining traffic speed bands, analysis of road traffic congestion, and prediction of future headways for better bus fleet management make use of various unstructured traffic data, including footage from surveillance cameras at bus stands and other checkpoints using the real-time data and batch processing layers of MOBDA. Machine vision is used for calculating service level metrics and other measures for road traffic congestion. DL and other ML models are trained using the meta-data table by employing the Cassandra data store in the storage layer of MOBDA. Table 3 provides the choice of technologies used in this study for processing transport related big data and the service level reporting with typical performance metrics required for the ITS. We adopted Spark SQL to make queries on the bus arrival data from the storage layer and the output of a sample query is provided in Figure 10. We illustrate, in Figure 10, the testing of the data pipeline implementation using the Spark SQL system, conducted by outputting the data values related to each bus and bus stop such as bus code, bus arrival time, departure time, bus stop code, and location.



```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.3.1.3.0.1.0-187
      /_/

sing Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_191)
ype in expressions to have them evaluated.
ype :help for more information.

cala> spark.sql("select * from busarrival_hive").show()
9/04/13 16:25:05 WARN HiveConf: HiveConf of name hive.heapsize does not exist
9/04/13 16:25:05 WARN HiveConf: HiveConf of name hive.stats.fetch.partition.stats does not exist
ive Session ID = 51a7f6e8-c93d-4876-b189-0aa1b9b0c9b5
--------------------+----------+---------+-------------------+-------------------+--------------+-------------+----------+----------+--------------------+---------------+------------------+---------
-------+----------+----------+--------------------+---------------+------------------+------------------+----------+----------+--------------------+---------------+------------------+-----------------+
                  id|busstopcode|serviceno|    nextbusarrival1|    nextbusarrival2|nextbusarrival3|        times|bus_1_type|bus_1_load|bus_1_destinationcode|bus_1_origincode|   bus_1_longitude|    bus_1_
atitude|bus_2_type|bus_2_load|bus_2_destinationcode|bus_2_origincode|   bus_2_longitude|    bus_2_latitude|bus_3_type|bus_3_load|bus_3_destinationcode|bus_3_origincode|bus_3_longitude|bus_3_latitude|
--------------------+----------+---------+-------------------+-------------------+--------------+-------------+----------+----------+--------------------+---------------+------------------+---------
-------+----------+----------+--------------------+---------------+------------------+------------------+----------+----------+--------------------+---------------+------------------+-----------------+
01012-12-15551616...|     01012|       12|2019-04-13 21:23:25|2019-04-13 21:36:30|          null|1555161662023|        DD|       SEA|               77009|          10499|103.84747616666667|1.28926866
6666666|        DD|       SEA|               77009|          10499|                 0|                 0|        DD|       SEA|               77009|          10499|             0|             0|
01012-12-15551617...|     01012|       12|2019-04-13 21:24:09|2019-04-13 21:35:44|          null|1555161744986|        DD|       SEA|               77009|          10499|        103.849425|1.29233883
3333334|        DD|       SEA|               77009|          10499|       103.8319275|1.2761121666666666|        DD|       SEA|               77009|          10499|             0|             0|
01012-12-15551619...|     01012|       12|2019-04-13 21:23:48|2019-04-13 21:36:19|          null|1555161926249|        DD|       SEA|               77009|          10499|103.85171933333334|         1
2958745|        DD|       SEA|               77009|          10499|103.83672683333333|1.2776438333333333|        DD|       SEA|               77009|          10499|             0|             0|
01012-12-15551621...|     01012|       12|2019-04-13 21:35:50|2019-04-13 21:49:30|          null|1555162107055|        DD|       SEA|               77009|          10499|        103.843516|1.28452216
6666666|        DD|       SEA|               77009|          10499|                 0|                 0|        DD|       SEA|               77009|          10499|             0|             0|
01012-12-15551622...|     01012|       12|2019-04-13 21:35:34|2019-04-13 21:44:43|          null|1555162287810|        DD|       SEA|               77009|          10499|103.84560233333333|         1
2870595|        DD|       SEA|               77009|          10499|       103.8332805|1.2755524999999999|        DD|       SEA|               77009|          10499|             0|             0|
01012-12-15551624...|     01012|       12|2019-04-13 21:35:57|2019-04-13 21:48:47|          null|1555162468991|        DD|       SEA|               77009|          10499|       103.8492565|1.29192133
```

Figure 10. Results of a sample query of bus arrival data using Spark SQL.

The quality and frequency of bus service, reliability, waiting time, and other metrics are key factors of passenger satisfaction [53]. The time interval between bus arrivals denoted by bus service headway is a key quality of service (QoS) metric in a microservice for management reporting. It is also used for the calculation of incentives/penalties for bus service operators. We model the bus arrival data obtained at each time instance, such as t0 and t1 at each bus stop, as shown in Figure 11. The pictorial representation of three different buses—Bus1, Bus2 and Bus3—at a specific bus stop and its queueing at different timestamps illustrates the importance of metrics, such as average headways, in making decisions that are further constrained by the queue size and bus bay length at the bus stop. For this purpose, average headways are calculated using the following formula:

$$Average\ Headways = \sum_{i=1}^{n} \frac{t_{Bus\ i+1} - t_{Bus\ i}}{n-1}$$

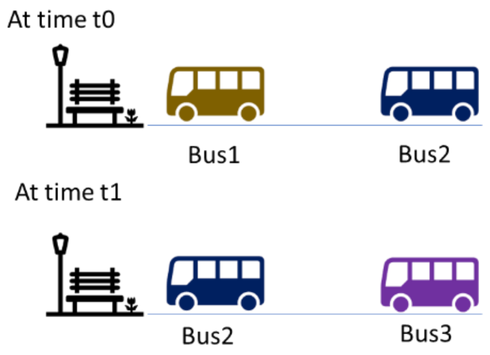


Figure 11. Bus arrival at a bus stop for different time instances.

The average bus service headway was calculated using bus arrival data gathered from 5021 bus stops in total. Table 4 shows the output for a sample of 40 bus services. In order to establish an improved bus fleet management, bus service headway was predicted using ML models such as linear regression and random forest regression [54]. Spark ML was used to fit regression models to predict the headway for the next time frame and the output is given in Table 5. The Spark processing pipeline using Spark ML was comprised of two steps:

(i). Assembly of various features. The sole feature of this simple regression model was based on the timestamp.

(ii). Fitting the ML model. Two different machine learning models were fitted—linear regression and random forest regression model. The evaluation metric used was root mean square error.

Table 4. Average bus service headway output generated using Spark SQL.

ServiceNo	Average_Headway	ServiceNo	Average_Headway
51	821.0	174	716.0
7	648.0	502	1054.0
124	584.0	61	784.0
700	655.0	75	689.0
851	608.0	166	889.0
11	1124.0	131	699.0
961	751.0	130	813.0
64	835.0	147	634.0
133	604.0	851e	1619.0
162	706.0	77	685.0
139	779.0	111	738.0
7A	1676.0	197	859.0
16	671.0	167	846.0
171	594.0	190	404.0
980	832.0	857	430.0
195	631.0	33	781.0
107	858.0	48	906.0
167e	1189.0	97	698.0
100	692.0	106	625.0
70	612.0	158	949.0

Table 5. Regression model output of bus service headways using Spark ML.

Prediction	Headway	Features
618.8091607030974	792	[$1.554951864 \times 10^{-9}$]
739.4689226078593	705	[$1.554952656 \times 10^{-9}$]
711.9043973618633	593	[$1.554953361 \times 10^{-9}$]
736.4733593312665	858	[$1.554953954 \times 10^{-9}$]
642.3014599489723	478	[$1.554954812 \times 10^{-9}$]

For the road checkpoint image classification in Spark DL, we used various road checkpoint images taken from traffic cameras and transformed the images to obtain the features. The following steps were adopted in the processing, training and prediction:

(i). Loading images. We made use of utility functions such as ImageSchema.readImages() in Pyspark to load millions of images in Spark Dataframe. It was split to obtain train dataframe and test dataframe.
(ii). Model definition: The DeepImageFeaturizer provided by SparkDL uses a pre-trained DL model, such as InceptionV3, which was used to transform the images into numeric features. Logistic Regression was used to perform binary classification in addition to the features provided by InceptionV3. We employed pyspark.ml pipeline to define the model.
(iii). Model training: The model pipeline was trained using the training dataframe.
(iv). Model prediction: The model was employed to predict on the test dataframe. To evaluate the accuracy of the model, a MultiClassClassificationEvaluator() utility function was used.
(v). Model testing: A new set of images was loaded in a Spark dataframe and the trained model was used to obtain the labels (high or low) for the given images.

The root mean square error (RMSE), a standard deviation that provides a measure of the prediction errors, was calculated. Our prediction of bus service headway using linear regression model resulted with an RMSE of 291.46, while the random forest regression model resulted with an RSME of only 206.42.

Overall, the use case of our proposed MOBDA for the scenario of determining bus service headway demonstrates the computation and reporting of important metrics used in microservices of a modern

ITS. These performance metrics and data insights report the current condition and effectiveness of the system in providing the desired service level of a microservice or an ITS subcomponent. The application of MOBDA effectively leverages two different computation archetypes: stream processing and batch processing. The stream processing layer enables real-time reporting of the metrics while batch processing enables building complex processing pipelines and models. The batch processing layer has successfully used SparkSQL and SparkML to compute the metrics.

6. Discussion

The successful prototype development of our proposed MOBDA leveraged Apache Kafka as the entry point to the Data Lake. Once the API responses from LTA data mall were pushed to the Data Lake as messages to Kafka broker using producers, the processes from Nifi, a software project of Apache, were adopted to deliver these messages to storage sinks. Nifi provided an effective data flow management, making it easier to move data between different kinds of sources and sinks. Once the data was stored in HBase, a few of the required tables were exposed as Hive tables, which made it easier to connect to data processing engines like Apache Spark.

The design of our hybrid model in MOBDA facilitated in having each component of the architecture to be easily scaled based on application requirements. Our proposed model is quite different from the recently proposed microservice-oriented platform reported as a proof of concept in literature [55]. It was called IoBDA platform as it was proposed to facilitate big data analytics (BDA) in IoT environments. The study concluded that scalability of components is a limitation in IoBDA's processing abilities. While IoBDA focuses on Monte Carlo and convergence analytics in IoT-based systems, our proposed MOBDA is more generic and scalable. Further, we demonstrated the successful application of MOBDA in a real-world case scenario of an ITS. Another similar research proposes the use of Apache Spark for ITSs by employing a single data source and a prototype processing pipeline for descriptive analytics [56], but lacks generalized architecture and scalability.

Further, in this discussion, we compare the merits of our work in this paper with a review on various architectural models and their stereotypical use cases that were profiled recently [57]. We observe from such reviews that the hybrid architecture is preferred in situations where there is a need for both real-time analytics and periodic batch analytics on the entire data collection. The kappa architecture is particularly suited for real-time analytics such as user behavior, network analytics, and social media trends. The zeta architecture is particularly suitable for complex data-centric applications that enables broker-based communications. For instance, zeta architecture is useful for building machine learning systems of an analytics solution. The recently proposed iota architecture is an extension of zeta architecture but focuses only on stream-based event triggered systems (covering both batch and real-time processing). On the other hand, the microservices architecture is useful to achieve modularity, query-based interactivity and service-based interactive responses. They are generally found to be deployed as apps for stakeholder, device, and sensor interactions. Similarly, in another comparative study, Lambda, Kappa, and IoT Architecture were profiled with a focus on the four V's (volume, variety, velocity, and veracity) of big data, as well as latency, storage, adaptability, and processing aspects [58]. However, the study did not provide implementation details to support the stated claims. Another survey [59] on various big data architectural models briefly reviewed the ecosystem and conceptual terminologies as defined by research, academia and industry.

A modular ITS for automobile accident detection system (U.S. Patent No. 7,983,835) [60] used both audio and visual inputs to identify special situations. It was highly useful in incident monitoring in situations where it is important to directly or inferentially determine whether an accident had occurred. This patent used a proprietary hybrid algorithm thereby limiting its application. In another IoV (Internet of Vehicles)-based real-time analytics proposal [61], IoT devices, fog computing, and cloud computing were suggested for solving traffic congestion by incorporating alternate route calculations. The proposal used three main layers (vehicles, intelligent road, and communication messages) and was tested using simulated data by employing cloud/fog services. The work supported hybrid models; however, lacked in adaptability and scalability due to the inherent limitations of the velocity and veracity aspects of big data in their architecture and the restricted nature of the simulated data used.

In comparison to the above-mentioned studies and proposals, our proposed MOBDA uses a hybrid architecture leveraging the advantages of lambda, kappa, and microservices appropriately. We have adopted an inclusive approach in building a solution model that positions well to embrace Industry 4.0. The use case of MOBDA has been demonstrated with a real-world open data, as well as real-time data streams and messages of an ITS with a futuristic purpose of improving various service metrics such as traffic prediction accuracies and entropy of parameters.

A major challenge in any big data processing architecture implementation is in real-time processing of IoT data, as most of the sensors will be generating big data every second resulting in high data ingestion rates. In MOBDA, by seamless addition of more worker nodes in the cluster, Nifi and Kafka's throughput can be increased to handle data at a rate of a few gigabytes per second, which is a typical requirement of large-scale big data IoT architectures. By using HBase we have ensured that the data is persisted quickly. Additionally, given that incoming data from IoT are small messages, HBase was a preferred choice over Hive with higher overhead (due to starting a Map-Reduce job for each small batch of messages) in persisting data. Finally, Spark was used to process the data as it is the industry standard for large scale data processing. The stream processing layer enabled real-time reporting of the metrics while batch processing enabled in building complex processing pipelines and models. The batch processing layer was successfully implemented using SparkSQL and SparkML to compute the metrics.

The use case of MOBDA was tested with the transportation data obtained from the Singapore Land Transport Authority (LTA) DataMall. This data is actively consumed by research community under Singapore's open data initiative for various purposes. The use of LTA's public datasets is gaining popularity due to the realization of various research outcomes and real-world implications that include developing (i) plans for charging infrastructure of electronic vehicle [62]; (ii) sustainable transportation policies, in particular for last mile metrics [63]; (iii) offline journey mappings as part of smart nation initiative [64]; and (iv) timetabling of trains in transit link line system [65].

The LTA provided API access for various types of transport related data in Singapore. Three types of data were obtained from the DataMall, namely real-time bus arrival information for various bus services, images of live traffic conditions, and traffic speed bands. Three Kafka producers were implemented in Java language. Each of the producers made a REST API call to LTA DataMall and pushed the response to the broker. The processed results from Spark were written back as Hive tables. The results stored in these tables were exposed through API services and were used for visualization and exploratory analysis using tools like Apache Zeppelin. Table 6 gives an overall efficiency in speed of API call frequency and LTA update frequency for bus arrival, traffic images, and traffic speed bands.

Table 6. Sources of real-time data and API call frequency.

Data	Response Type	LTA Update Frequency	API Call Frequency
Bus Arrival	JSON	1 min	3 min
Traffic Images	JSON and JPEG	1 to 5 min	4 min
Traffic Speed Bands	JSON	5 min	5 min

Use of Kafka, Nifi and HBase/Hive had guaranteed that the overall data flow of MOBDA was fault tolerant with no loss of messages. Kafka and Nifi guaranteed message delivery and HBase provided data replication to ensure that data was always available, and no data was lost.

The limitations of our proposed MOBDA are mainly attributed to its implementation with the technology options that we had to adopt while using LTA data. The ingested JSON messages stored as a form of Kafka logs were not suitable for visualization or exploratory analysis by tools like Zeppelin. Thus, there was a need to structure these messages in tabular format and write to suitable sinks like

HBase (for later processing by Spark) or HDFS (for historical record archival). In addition, we observed that the newer version of Kafka was packaged with numerous connectors under the umbrella called KafkaConnect, which could be used in future implementation and testing of MOBDA. KafkaConnect plugins could be employed to implement scalable pipelines using declarative syntax for transferring data from sources to sinks, and this would be more efficient as it eliminates the need for a separate integration layer. These observations of the limitations of this work motivates us to explore further in our future research in this direction.

7. Concluding Remarks and Future Work

With IoT-based transport services becoming the backbone of future ITSs, smart transportation and analytics leveraging big data is one of the major research challenges. In this paper, we proposed, developed, and applied a microservice-oriented big data architecture (MOBDA) for meeting the dynamically changing functional and non-functional service performance requirements of an ITS. To support smart transportation and analytics over large traffic data, the microservice and component-based hybrid technologies were adopted to integrate the big data streaming from heterogeneous and disparate platforms of various stakeholders of an ITS. MOBDA supports smart analytics on different data sources by combining both real-time stream processing and batch processing of big data to report several transport service use cases and performance metrics. A hybrid of these two data processing layers of MOBDA was designed to leverage different trade-offs between latency and throughput requirements. This aided in the adaptability and scalability of the big data architecture to efficiently handle the dynamically changing transportation environment. Further, with a use case of MOBDA in Singapore's busy bus transportation as a pilot study, we demonstrated the application of different layers of the architecture using Apache Storm topology, and successfully computed the bus service headway metrics as a use case for a smart transport reporting service within an ITS.

While we explored the various big data storage options such as HDFS and HBase as well as big data processing engines such as Hive, due to time limitations, the proposed MOBDA was not evaluated with the objective of establishing performance thresholds for each layer of the architecture individually. The serving layer was developed in Cassandra, whose query language does not support join, which might not be able to cater to all the analytical exploration use cases. In future, more sophisticated OLAP offerings, such as Apache Druid, would be integrated into the implementation. Guided by industry best practices, the architecture proposed in this paper scales horizontally with fault tolerance and can easily be adapted over any cloud/edge platform. However, future research direction would aim at extending the work to develop several other traffic-based performance metrics and microservice use cases. This pilot study provided a minimal viable product that would serve as a stepping stone to extend the smart transportation and analytics system to include a variety of metrics. By augmenting real-world transport data with other public data sets, machine learning and deep learning models for predicting various traffic conditions could be enhanced and more meaningful data insights could be derived.

Author Contributions: Conceptualization, S.P.R.A. and R.V.; Methodology, S.V.; Investigation, S.P.R.A.; Validation, R.V.; Writing—Original Draft Preparation, S.P.R.A., R.V. and S.V.; Writing—Review and Editing, S.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This work contains processed information from static and dynamic datasets from DataMall through the open data initiative of Land Transport Authority (LTA) of Singapore, which is made available under the terms of the Singapore Open Data License version 1.0 URL. The authors thank the students (Siddharth P, Pranshu R, Nyon Yan Zheng, Tan Kok Keng, Rajan D, Anjali S, Gopalakrishnan S) for taking part in the project that helped in the use case implementation. The authors also wish to acknowledge the reviewers and editor for their invaluable inputs and support rendered for this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chin, K.; Ong, G. Smart Mobility 2030–ITS Strategic Plan for Singapore. Available online: https://esci-ksp.org/wp/wp-content/uploads/2012/06/J15Nov_p04Chin_SmartMobility2030.pdf (accessed on 28 August 2019).
2. Eckhardt, J.; Aapaoja, A.; Nykänen, L.; Sochor, J.; Karlsson, M.; König, D. The European roadmap 2025 for mobility as a service. In Proceedings of the 7th Transport Research Arena TRA 2018, Vienna, Austria, 16–18 April 2018.
3. Goodall, W.; Fishman, T.D.; Bornstein, J.; Bontrhon, B. *The Rise of Mobility as a Service–Reshaping How Urbanites Get Around*; Deloitte Review; Deloitte University Press: New York, NY, USA, 2017.
4. Lau, S.K.; Zamani, R.; Susilo, W. A semantic web vision for an intelligent community transport service brokering system. In Proceedings of the 2016 IEEE International Conference on Intelligent Transportation Engineering (ICITE), Singapore, 20–22 August 2016; pp. 172–175.
5. Leong, W.; Goh, K.; Hess, S.; Murphy, P. Improving bus service reliability: The Singapore experience. *Res. Transp. Econ.* **2016**, *59*, 40–49. [CrossRef]
6. Trompet, M.; Liu, X.; Graham, D.J. Development of key performance indicator to compare regularity of service between urban bus operators. *Transp. Res. Rec.* **2011**, *2216*, 33–41. [CrossRef]
7. Herrera-Quintero, L.F.; Vega-Alfonso, J.C.; Banse, K.B.A.; Zambrano, E.C. Smart its sensor for the transportation planning based on iot approaches using serverless and microservices architecture. *IEEE Intell. Transp. Syst. Mag.* **2018**, *10*, 17–27. [CrossRef]
8. Verma, P.; Bhatia, J. Design and development of GPS-GSM based tracking system with Google map based monitoring. *Int. J. Comput. Sci. Eng. Appl.* **2013**, *3*, 33. [CrossRef]
9. Biuk-Aghai, R.P.; Kou, W.T.; Fong, S. Big data analytics for transportation: Problems and prospects for its application in China. In Proceedings of the 2016 IEEE Region 10 Symposium (TENSYMP), Bali, Indonesia, 9–11 May 2016; pp. 173–178.
10. Mullich, J. *Drivers Avoid Traffic Jams with Big Data and Analytics*; Bloomberg L.P.: New York, NY, USA, 2013.
11. Aoki, E.; Otsuka, S.; Ikenaga, T.; Yatsuzuka, M.; Tokiwa, K. Study on regional transportation linkage system that enables efficient and safe movement utilizing LPWA. In Proceedings of the Conference on Complex, Intelligent, and Software Intensive Systems, Matsue, Japan, 4–6 July 2018; pp. 968–977.
12. De Gennaro, M.; Paffumi, E.; Martini, G. Big data for supporting low-carbon road transport policies in Europe: Applications, challenges and opportunities. *Big Data Res.* **2016**, *6*, 11–25. [CrossRef]
13. Milne, D.; Watling, D. Big data and understanding change in the context of planning transport systems. *J. Transp. Geogr.* **2019**, *76*, 235–244. [CrossRef]
14. Tavasszy, L.; de Jong, G. Data availability and model form. In *Modelling Freight Transport*; Elsevier: Amsterdam, The Netherlands, 2014; pp. 229–244.
15. Fernández-Rodríguez, J.Y.; Álvarez-García, J.A.; Fisteus, J.A.; Luaces, M.R.; Magana, V.C. Benchmarking real-time vehicle data streaming models for a smart city. *Inf. Syst.* **2017**, *72*, 62–76. [CrossRef]
16. Wiese, L. Polyglot Database Architectures = Polyglot Challenges. In Proceedings of the LWA, Trier, Germany, 7–9 October 2015; pp. 422–426.
17. Persico, V.; Pescapé, A.; Picariello, A.; Sperlí, G. Benchmarking big data architectures for social networks data processing using public cloud platforms. *Future Gener. Comput. Syst.* **2018**, *89*, 98–109. [CrossRef]
18. Amini, S.; Gerostathopoulos, I.; Prehofer, C. Big data analytics architecture for real-time traffic control. In Proceedings of the 2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), Napoli, Italy, 26–28 June 2017; pp. 710–715.
19. Chen, L.-W.; Chung, J.-J. Mobility-aware and congestion-relieved dedicated path planning for group-based emergency guiding based on internet of things technologies. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 2453–2466. [CrossRef]
20. Menon, A.; Sinha, R.; Ediga, D.; Iyer, S. Implementation of internet of things in bus transport system of Singapore. *Asian J. Eng. Res.* **2013**, *1*, 8–17.
21. Sun, S.; Akhtar, N.; Song, H.; Zhang, C.; Li, J.; Mian, A. Benchmark data and method for real-time people counting in cluttered scenes using depth sensors. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 3599–3612. [CrossRef]

22. Reyes García, J.R.; Lenz, G.; Haveman, S.P.; Bonnema, G.M. State of the Art of Mobility as a Service (MaaS) Ecosystems and Architectures—An Overview of, and a Definition, Ecosystem and System Architecture for Electric Mobility as a Service (eMaaS). *World Electr. Veh. J.* **2020**, *11*, 7. [CrossRef]
23. Van Oort, N.; Cats, O. Improving public transport decision making, planning and operations by using big data: Cases from Sweden and the Netherlands. In Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems, Las Palmas de Gran Canaria, Spain, 15–18 September 2015; pp. 19–24.
24. Authority, L.T. *DataMall @ MyTransport.sg an LTA Open Data Initiative*; API User Guide & Documentation; DataMall: Singapore, 2019.
25. Lin, J. The lambda and the kappa. *IEEE Internet Comput.* **2017**, *5*, 60–66. [CrossRef]
26. Ta, V.-D.; Liu, C.-M.; Nkabinde, G.W. Big data stream computing in healthcare real-time analytics. In Proceedings of the 2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), Chengdu, China, 5–7 July 2016; pp. 37–42.
27. Zhou, Z.; Gao, C.; Xu, C.; Zhang, Y.; Mumtaz, S.; Rodriguez, J. Social big-data-based content dissemination in internet of vehicles. *IEEE Trans. Ind. Inform.* **2017**, *14*, 768–777. [CrossRef]
28. Batyuk, A.; Voityshyn, V. Apache storm based on topology for real-time processing of streaming data from social networks. In Proceedings of the 2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 23–27 August 2016; pp. 345–349.
29. Tang, L.; Zhao, Y.; Cabrera, J.; Ma, J.; Tsui, K.L. Forecasting short-term passenger flow: An empirical study on shenzhen metro. *IEEE Trans. Intell. Transp. Syst.* **2018**, *20*, 3613–3622. [CrossRef]
30. Fernandez, S.; Hadfi, R.; Ito, T.; Marsa-Maestre, I.; Velasco, J.R. Ontology-based architecture for intelligent transportation systems using a traffic sensor network. *Sensors* **2016**, *16*, 1287. [CrossRef]
31. Goel, D.; Pahal, N.; Jain, P.; Chaudhury, S. An ontology-driven context aware framework for smart traffic monitoring. In Proceedings of the 2017 IEEE Region 10 Symposium (TENSYMP), Cochin, India, 14–16 July 2017; pp. 1–5.
32. Hachani, S.; Gzara, L.; Verjus, H. A service-oriented approach for flexible process support within enterprises: Application on PLM systems. *Enterp. Inf. Systems* **2013**, *7*, 79–99. [CrossRef]
33. Wang, J.; Jiang, C.; Han, Z.; Ren, Y.; Hanzo, L. Internet of vehicles: Sensing-aided transportation information collection and diffusion. *IEEE Trans. Veh. Technol.* **2018**, *67*, 3813–3825. [CrossRef]
34. Wang, Y.; Ram, S.; Currim, F.; Dantas, E.; Sabóia, L.A. A big data approach for smart transportation management on bus network. In Proceedings of the 2016 IEEE International Smart Cities Conference (ISC2), Trento, Italy, 12–15 September 2016; pp. 1–6.
35. Paulraj, D.; Swamynathan, S.; Madhaiyan, M. Process model-based atomic service discovery and composition of composite semantic web services using web ontology language for services (OWL-S). *Enterp. Inf. Syst.* **2012**, *6*, 445–471. [CrossRef]
36. Pflügler, C.; Schreieck, M.; Hernandez, G.; Wiesche, M.; Krcmar, H. A concept for the architecture of an open platform for modular mobility services in the smart city. *Transp. Res. Procedia* **2016**, *19*, 199–206. [CrossRef]
37. Beutel, M.C.; Gökay, S.; Kluth, W.; Krempels, K.-H.; Samsel, C.; Terwelp, C. Product oriented integration of heterogeneous mobility services. In Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qindao, China, 8–11 October 2014; pp. 1529–1534.
38. Vergilio, T.; Ramachandran, M. Non-functional requirements for real world big data systems: An investigation of big data architectures at Facebook, Twitter and Netflix. In Proceedings of ICSOFT 2018-Proceedings of the 13th International Conference on Software Technologies, Porto, Portugal, 26–28 July 2018; pp. 833–840.
39. Chen, C.; Jiao, S.; Zhang, S.; Liu, W.; Feng, L.; Wang, Y. TripImputor: Real-time imputing taxi trip purpose leveraging multi-sourced urban data. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 3292–3304. [CrossRef]
40. Pecheux, K.K.; Pecheux, B.B.; Carrick, G. *Leveraging Big Data to Improve Traffic Incident Management*; The National Academies of Sciences Engineering and Medicine: Washington, DC, USA, 2019.
41. Chien, S.I.-J.; Kuchipudi, C.M. Dynamic travel time prediction with real-time and historic data. *J. Transp. Eng.* **2003**, *129*, 608–616. [CrossRef]
42. Singla, L.; Bhatia, P. GPS based bus tracking system. In Proceedings of the 2015 International Conference on Computer, Communication and Control (IC4), Indore, India, 10–12 September 2015; pp. 1–6.
43. Vanajakshi, L.; Subramanian, S.C.; Sivanandan, R. Travel time prediction under heterogeneous traffic conditions using global positioning system data from buses. *IET Intell. Transp. Syst.* **2009**, *3*, 1–9. [CrossRef]

44. Williams, B.M.; Hoel, L.A. Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *J. Transp. Eng.* **2003**, *129*, 664–672. [CrossRef]
45. Pattar, S.; Kulkarni, D.S.; Vala, D.; Buyya, R.; Venugopal, K.; Iyengar, S.; Patnaik, L. Progressive Search Algorithm for Service Discovery in an IoT Ecosystem. In Proceedings of the 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Atlanta, GA, USA, 14–17 July 2019; pp. 1041–1048.
46. Singh, A.; Khamparia, A.; Luhach, A.K. Performance comparison of Apache Hadoop and Apache Spark. In Proceedings of the Third International Conference on Advanced Informatics for Computing Research, Shimla, India, 15–16 June 2019; pp. 1–5.
47. Kormáksson, M.; Barbosa, L.; Vieira, M.R.; Zadrozny, B. Bus travel time predictions using additive models. In Proceedings of the 2014 IEEE International Conference on Data Mining, Shenzhen, China, 14–17 December 2014; pp. 875–880.
48. Shalaby, A.; Farhan, A. Prediction model of bus arrival and departure times using AVL and APC data. *J. Public Transp.* **2004**, *7*, 3. [CrossRef]
49. Córdoba, A.; Astrain, J.J.; Villadangos, J.; Azpilicueta, L.; López-Iturri, P.; Aguirre, E.; Falcone, F. Sestocross: Semantic expert system to manage single-lane road crossing. *IEEE Trans. Intell. Transp. Syst.* **2016**, *18*, 1221–1233. [CrossRef]
50. Chen, R.; Tong, Y.; Yang, J.; Wu, M. Residual reconstruction algorithm based on sub-pixel multi-hypothesis prediction for distributed compressive video sensing. In Proceedings of the Conference on Complex, Intelligent, and Software Intensive Systems, Sydney, Australia, 3–5 July 2019; pp. 599–605.
51. Lin, Y.; Yang, X.; Zou, N.; Jia, L. Real-time bus arrival time prediction: Case study for Jinan, China. *J. Transp. Eng.* **2013**, *139*, 1133–1140. [CrossRef]
52. Oliveira, F.R.; del Val Cura, L. Performance evaluation of NoSQL multi-model data stores in polyglot persistence applications. In Proceedings of the 20th International Database Engineering & Applications Symposium, Montreal, QC, Canada, 11–13 July 2016; pp. 230–235.
53. Sun, D.; Luo, H.; Fu, L.; Liu, W.; Liao, X.; Zhao, M. Predicting bus arrival time on the basis of global positioning system data. *Transp. Res. Rec.* **2007**, *2034*, 62–72. [CrossRef]
54. Rajput, P.; Toshniwal, D.; Agggarwal, A. Improving infrastructure for transportation systems using clustering. In Proceedings of the International Conference on Big Data Analytics, Ahmedabad, India, 17–20 December 2019; pp. 129–143.
55. Li, Z.; Seco, D.; Sánchez Rodríguez, A.E. Microservice-oriented platform for internet of big data analytics: A proof of concept. *Sensors* **2019**, *19*, 1134. [CrossRef]
56. Zhu, L.; Yu, F.R.; Wang, Y.; Ning, B.; Tang, T. Big data analytics in intelligent transportation systems: A survey. *IEEE Trans. Intell. Transp. Syst.* **2018**, *20*, 383–398. [CrossRef]
57. Kalipe, G.K.; Behera, R.K. Big Data Architectures: A detailed and application oriented review. *Int. Journal Innov. Technol. Explor. Eng.* **2019**, *8*, 2182–2190.
58. Rassam, L.; Zellou, A.; Rachad, T. *Towards a Big Data Architecture for Heterogeneous Data Sources*; EasyChair: Manchester, UK, 2020.
59. Singh, K.N.; Behera, R.K.; Mantri, J.K. Big Data Ecosystem: Review on Architectural Evolution. In *Emerging Technologies in Data Mining and Information Security*; Springer: Berlin, Germany, 2019; pp. 335–345.
60. Lagassey, P.J. Modular Intelligent Transportation System. Google Patent US6810817B1, 2 February 2011.
61. Nahri, M.; Boulmakoul, A.; Karim, L.; Lbath, A. IoV distributed architecture for real-time traffic data analytics. *Procedia Comput. Sci.* **2018**, *130*, 480–487. [CrossRef]
62. Xue, F.; Gwee, E. Electric vehicle development in singapore and technical considerations for charging infrastructure. *Energy Procedia* **2017**, *143*, 3–14. [CrossRef]
63. Lee, D.-H.; Palliyani, S. Sustainable transport policy-An evaluation of Singapore's past, present and future. *J. Infrastruct. Policy Dev.* **2017**, *1*, 112–128. [CrossRef]

64. Keong, A.P.; Smitha, K.; Sinha, S. Smart Nation: Offline Public Transport Made Easy. In Proceedings of the 2019 4th International Conference on Intelligent Transportation Engineering (ICITE), Singapore, 5–7 September 2019; pp. 175–179.
65. Li, D.; Zhang, T.; Dong, X.; Yin, Y.; Cao, J. Trade-off between efficiency and fairness in timetabling on a single urban rail transit line under time-dependent demand condition. *Transp. B Transp. Dyn.* **2019**, *7*, 1203–1231. [CrossRef]