POLITECNICO
MILANO 1863

BIGSEA: A Big Data analytics platform for public transportation information

Andy S. Alic, Jussara Almeida, Giovanni Aloisio, Nazareno Andrade, Nuno Antunes, Danilo Ardagna, Rosa M. Badia, Tania Basso, Ignacio Blanquer, Tarciso Braz, Andrey Brito, Donatello Elia, Sandro Fiore, Dorgival Guedes, Marco Lattuada, Daniele Lezzi, Matheus Maciel, Wagner Meira, Demetrio Mestre, Regina Moraes, Fabio Morais, Carlos Eduardo Pires, Nadia P. Kozievitch, Walter dos Santos, Paulo Silva, Marco Vieira

Andy S. Alic, Jussara Almeida, Giovanni Aloisio, Nazareno Andrade, Nuno Antunes, Danilo Ardagna, Rosa M. Badia, Tania Basso, Ignacio Blanquer, Tarciso Braz, Andrey Brito, Donatello Elia, Sandro Fiore, Dorgival Guedes, Marco Lattuada, Daniele Lezzi, Matheus Maciel, Wagner Meira, Demetrio Mestre, Regina Moraes, Fabio Morais, Carlos Eduardo Pires, Nadia P. Kozievitch, Walter dos Santos, Paulo Silva, and Marco Vieira. Bigsea: A big data analytics platform for public transportation information. *Future Generation Computer Systems*, 96:243 – 269, 2019

The final publication is available via http://dx.doi.org/https://doi.org/10.1016/j.future.2019.02.011

BIGSEA: A Big Data analytics platform for public transportation information

Andy S Alic[1], Jussara Almeida[1], Giovanni Aloisio[1], Nazareno Andrade[1], Nuno Antunes[1], Danilo Ardagna[1], Rosa M. Badia[1,1], Tania Basso[1], Ignacio Blanquer[1,*], Tarciso Braz[1], Andrey Brito[1], Donatello Elia[1], Sandro Fiore[1], Dorgival Guedes[1], Marco Lattuada[1], Daniele Lezzi[1], Matheus Maciel[1], Wagner Meira Jr.[1], Demetrio Mestre[1], Regina Moraes[1], Fabio Morais[1], Carlos Eduardo Pires[1], Nadia Puchalski Kozievitch[1], Walter dos Santos[1], Paulo Silva[1], Marco Vieira[1]

[a] *Institute of Instrumentation for Molecular Imaging (I3M), Universitat Politècnica de València - CSIC*
[b] *Universidade Federal de Minas Gerais (UFMG), Brazil*
[c] *Fondazione Centro Euro-Mediterraneo sui Cambiamenti Climatici (CMCC), Italy*
[d] *Universidade Federal de Campina Grande (UFCG), Brazil*
[e] *CISUC, Department of Informatics Engineering, University of Coimbra, Portugal*
[f] *Politecnico di Milano, Milan, Italy*
[g] *Barcelona Supercomputing Center (BSC)*
[h] *University of Campinas (UNICAMP), Brazil*
[i] *Universidade Tecnológica Federal do Paraná (UTFPR), Brazil*
[j] *Artificial Intelligence Research Institute - Spanish National Research Council (IIIA-CSIC)*

Abstract

Data analysis of public transportation data in large cities is a challenging problem. Managing data ingestion, data storage, data quality enhancement, modelling and analysis requires intensive computing and a non-trivial amount of resources. In *EUBra-BIGSEA* (Europe-Brazil Collaboration of Big Data Scientific Research Through Cloud-Centric Applications), we address such problems in a comprehensive and integrated way. *EUBra-BIGSEA* provides a platform for building up data analytic workflows on top of elastic cloud services without requiring skills related to either programming or cloud services. The approach combines cloud orchestration, Quality of Service and automatic parallelisation on a platform that includes a toolbox for implementing privacy guarantees and data quality enhancement as well as advanced services for sentiment analysis, traffic jam estimation and trip recommendation based on estimated crowdedness. All developments are available under Open Source licenses (http://github.org/eubr-bigsea, https://hub.docker.com/u/eubrabigsea/).

1. Introduction

Public transportation in large cities is a major source of high-valuable data to understand and improve the citizens' lifestyle and to dynamically react to unplanned events. Multiple heterogeneous datasources are available, and different data analytics tools do exist. However, processing such data requires downloading the data, installing processing tools, managing the resources and developing processing software.

EUBra-BIGSEA[1] (Europe - Brazil Collaboration of Big Data Scientific Research Through Cloud-Centric Applications) is a collaboration aimed at developing convenient data analytic services based on the cloud mainly tailored for public transportation data, able to process data under several restrictions, such as Quality of Service constraints and Privacy-awareness, by means of convenient and auto-parallelisable programming models. *EUBra-BIGSEA* has developed and implemented a software architecture that addresses a major number of software requirements for three main use cases on Public transportation data analysis.

*Corresponding author with email: iblanque@dsic.upv.es

[1] *EUBra-BIGSEA* is a project funded by the European Commission under the Cooperation Programme, Horizon 2020 grant agreement No 690116. Este projeto é resultante da 3a Chamada Coordenada BR-UE em Tecnologias da Informação e Comunicação (TIC), anunciada pelo Ministério de Ciência, Tecnologia e Inovação (MCTI)

1.1. Requirements

Three main global use cases have been identified in public transportation data management. These three use cases refer to main issues: 1) Data Acquisition - ingesting heterogeneous and medium-quality time-varying data; 2) Creation and execution of Descriptive models - models that derive additional information and knowledge from raw data; and 3) Creation and execution of Predictive Models - to anticipate future events on a variety and diversity of scenarios.

Each use case imposes a set of requirements:

- Data Acquisition:
  - Integration of GIS, public transportation and meteorological/climate data sources, supporting CSV, XLS, JSON, Shapefile and NetCDF at least.
  - Integrating and dealing with Metadata from those sources.
  - Data quality improvement by cross correlation.
  - Supporting different Access Control level for the data and Metadata.
- Descriptive Models. A fundamental abstraction of the descriptive models are trajectories, that is, the path traversed by each end user while using public transportation:
  - Developing models to extract and characterize trajectories from vehicle movement data. Trajectories comprise not only dynamic spatial data, but also the other types of data that enrich the trajectory information.
  - Determining correlations and cluster trajectories to improve quality by integrating multiple sources.
  - Defining areas of interest to limit the processing data.
- Predictive Models, involving the whole life-cycle and focusing on trip analysis and trip selection.
  - Training, validating and building Predictive Models based on geographic (static and dynamic), social, and meteorological data.
  - Recomputing Predictive Models periodically and with a predictable performance.
  - Project Predictive Models as a service.
  - Specifying data sources and regions of interest for any of the above-mentioned operations.

One of the goals of *EUBra-BIGSEA* is to provide a simple interface through which data scientists can describe their processing tasks. This is achieved through the integration of a visual data flow tool with the programming models and the underlying architecture. Many data mining tools support the creation of data flows visually by using building blocks on Graphical User Interface (*e.g.*, RapidMiner [**?**], Orange [**?**] and KNIME [**?**]). Most of these platforms do not include distributed execution features. Others, such as Microsoft Azure Machine Learning (ML) Studio [**?**] and ClowdFlows [**?**] allow tasks to be executed in distributed fashion, but do not include functionalities to exploit parallelism, manage a coherent authentication and authorization model or provide data privacy annotation and enhancement tools. Moreover, they are bound to specific cloud deployments and providers.

1.2. Platform architecture

According to the requirement enactment and the analysis of the state of the art, *EUBra-BIGSEA* has proposed an infrastructure (see Figure **??**) that addresses the following needs:

- Efficient and convenient development of data analytic applications addressing different access levels (application building based on graphical interfaces, use of general purpose programming languages, use of data-analytic specific APIs and use of data-analytic specific languages such as Spark).

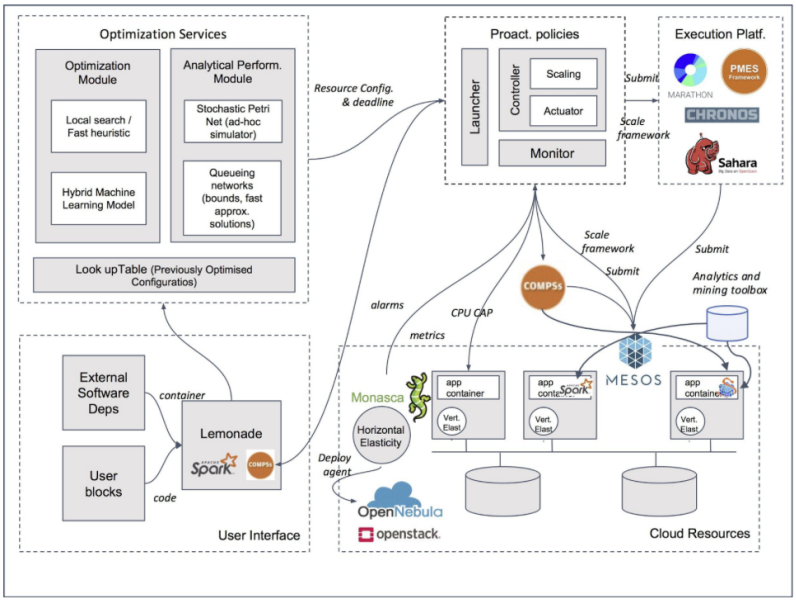


Figure 1: EUBra-BIGSEA software architecture.

- Application characterization and performance prediction under different scenarios (larger or smaller number of resources) in parallel data analytic applications. This is achieved by means of a log analyzer, a performance prediction service and an optimizer module, which learns, projects and backsolves the problem of finding the resource requirements for reaching a given deadline.
- Horizontal and vertical elasticity at the level of the cloud resources that run the data analytic applications by means of a fine-grain monitoring, which triggers the deployment, power-on, contextualization of computing resources and the dynamic allocation of resources to the jobs executed.
- Being able to characterize sensitive data thanks to policies at the level of fields in a dataset, by means of a framework that annotates parts of the dataset and implements privacy enhancement policies.

Each component is described in the following sections.

2. Programming Models

The programming abstraction layer offers developers the functionalities needed to satisfy the requirements for the implementation of the applications scenarios on top of the Big Data layer of the BISGEA platform. In the next sections, we focus on the description of the components that enable the development of modules and libraries (building blocks) which abstract the data layer intricacies to the applications. *EUBra-BIGSEA* provides two programming models by means of two frameworks (COMPSs and Apache Spark) that provide the developers with complementary functionalities. The implementation can be done from scratch by adopting the most appropriate model, or through the Lemonade (Live Exploration and Mining Of Non-trivial Amount of Data from Everywhere) platform, by composing existing building blocks to generate the code.

2.1. Lemonade

Visual workflows tools provide a higher level of abstraction than general-purpose programming languages,even those specifically created for data processing, such as the "R" language. Currently, the increased capacity and reduced price of existing processing infrastructures, as well as the availability of large amounts of data, has democratized the development of new applications, previously restricted to very large companies and organizations. However, to fully exploit such opportunity, a team should deal with different expertise, such as business domain, programming skills and infrastructure maintenance. Sometimes, researchers just want to test a hypothesis about the data. If they require a complex learning process to use a specific technical solution, this will not be used.

Lemonade is a visual platform for distributed computing, aimed at enabling the implementation, experimentation, testing and deployment of data processing and machine learning applications. It provides high-level abstractions, called operations, for developers to build processing workflows using a graphical web interface. Lemonade uses high-performance and scalable technologies for discovering inherent concurrency (such as COMPSs for automatic parallelisation of workflows and Ophidia, providing parallel data analytic functions) to enhance Spark code. Lemonade can process very large amounts of data, hiding all back-end complexity to the users and allowing them to focus mainly on the construction of the solution.

The Lemonade architecture is composed of seven components, built as micro-services, which handle tasks including the user interface, the data management, the security, and the execution of processing jobs. Data sources meta-data (location, permissions, formats) are stored in *Limonero*, while meta-data of the available processing operations are kept in *Tahiti*. Operations are the smallest processing units and include Extract, Transform and Load (ETL) operations, data mining and machine learning algorithms, input/output, and visualization abstractions. The information stored in Tahiti includes configuration parameters for the algorithms, privacy and security constraints, visualization and QoS requirements. New operations can be easily created by adding the appropriate meta-data to Tahiti. The users web interface is managed by *Citron*, where flows can be built, instantiated and inspected. The actual execution of flows is controlled by *Juicer*, which generates COMPSs [**?**] or Spark code for the operations selected by the user and instantiates it in the cloud execution environment observing the user configuration parameters. The communication between Citron and Juicer is controlled by *Stand*, which ensures their independence and facilitates the use of different programming frameworks. The visualization of results through different metaphors is provided by *Caipirinha*. Finally, all the security, privacy and access control solutions developed in *EUBra-BIGSEA* are the responsibility of *Thorn*. The interaction of the components is illustrated in Figure **??**.

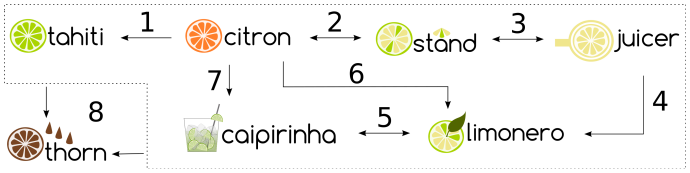


Figure 2: Communication between Lemonade components.

During operation, *Citron* determines the available operations, their parameters and the already created flows by accessing *Tahiti* (1). It also accesses *Stand* to deliver the description of the flows to be executed, and to receive feedback information about their execution (2). *Stand*, in turn, feeds the flow descriptions to *Juicer*, which is responsible for generating the actual COMPSs/Spark code that will be executed and starting it by means of the available cloud services (3), as well as returning the execution information that will be displayed by Stand (2). *Juicer* uses information from *Limonero* to locate and access the available data sources, as well as to register new data sets that are created as a result of a flow execution (4). *Limonero* also interacts with *Caipirinha* to enable data visualization (5) and back with *Citron* to make the new data sets available to the user. Visualizations can also be requested by the user directly from *Citron* (7). Finally, *Thorn* encapsulates all security control, regulating user access, providing API security keys, etc (8).

2.2. COMPSs

COMPSs [**?**][**?**] is a framework composed of a programming model and a runtime system, which aims to ease the development and deployment of distributed applications and web services. The core of the framework is its programming model, which allows the programmer to write applications in a sequential way and execute them on top of heterogeneous infrastructures by exploiting the inherent parallelism of the applications. The COMPSs programming model is task-based, allowing the programmer to select the methods of the sequential application to be executed remotely. This selection is done by means of an annotated interface where all the methods to be considered as tasks are defined with annotations describing their data accesses and constraints on the execution of resources. At execution time this information is used by the runtime system to build a dependency graph and orchestrate the tasks on the available resources, which can be nodes in a cluster, cloud or containers in a Mesos cluster.

One important feature of the COMPSs runtime is the ability to elastically adapt the amount of resources to the current workload. When the number of tasks is higher than the available cores, the runtime turns to the cloud looking for a provider which offers the type of resources that better meet the requirements of the application and are most cost-effective. Similarly, when the runtime detects an excess of resources for the actual workload, it will power off unused instances in a cost-efficient way. Such decisions are based on the information about the type of resources, that contains the details of the software images and instance templates available for every cloud provider. In the *EUBra-BIGSEA* project, this elasticity has been extended to support Mesos clusters. As depicted in Figure **??**, the implementation includes a scheduler for the COMPSs Runtime that receives the offers from the Mesos master and an Executor that runs on the slave nodes to execute the COMPSs tasks. Both components are executed in Docker containers and automatically deployed by Mesos. The black lines in the figure represent the communication amongst COMPSs and the Mesos Master to receive the updates on the offers, and also among the Mesos Master and the slaves to deploy the containers. Once the resources are offered to COMPSs, the runtime deploys its workers and establishes a direct connection (blue arrows) to send the tasks.

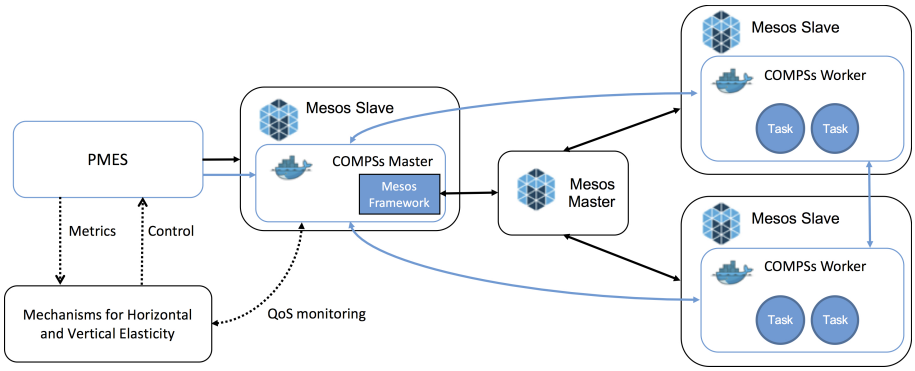


Figure 3: Integration of COMPSs in the *EUBra-BIGSEA* Platform

The deployment of COMPSs in *EUBra-BIGSEA* is complemented by the PMES service, which is a tool that eases the management of COMPSs applications. PMES is a service that takes care of the deployment, configuration and execution of COMPSs applications on distributed infrastructures. PMES has also been extended with a Mesos connector and acts as bridge between COMPSs and the rest of the QoS platform. To this aim, a set of specific methods has been added to the PMES REST interface to monitor the execution of the applications in Mesos. In the figure, the interactions of PMES with COMPSs and Mesos are depicted as black and blue lines, respectively; the interactions of PMES and COMPSs with the QoS services are depicted as dotted lines. When the monitoring system detects a need for additional resources in Mesos, it adds new nodes that are eventually offered to COMPSs, leaving to runtime the decision to profit from this change depending on the parallelism and the actual number of tasks (horizontal elasticity). On the other hand, vertical elasticity is completely transparent to the COMPSs execution because a change in the size (be it speed or capacity) of the virtual resource does not affect the scheduling policies but improves the performance of the execution of the single task on a given node.

2.3. Ophidia

Ophidia [**?**][**?**][**?**] is one of the main Big Data technologies involved in the *EUBra-BIGSEA* project to address the issues related to the data processing applications (e.g. descriptive models for routes from the raw data of public transportation) built on top of the project platform. It represents a framework that provides a complete environment for the execution of scientific data-intensive analysis, exploiting parallel computing techniques, data distribution methods, jointly with a native in-memory engine to perform parallel I/O operations. Ophidia provides an array-based storage model designed to handle multi-dimensional scientific datasets, implementing the data cube abstraction typical of On-Line Analytical Processing (OLAP) systems. From an architectural point of view, an Ophidia instance consists of the following components:

- some client modules, like the *CLI Ophidia Terminal* and *PyOphidia*, the Ophidia Python bindings [**?**];

- the *Ophidia Server*, a front-end server to submit the execution of analytics tasks or workflows. It also manages jobs scheduling and monitoring, as well as user authentication and authorization, integrating several AuthN/AuthZ methods as the one developed in the project, the token-based *AAA as a Service* (AAAaaS);
- the *Analytics framework,* providing a wide set of parallel MPI-based operators (both for data and metadata) executed over the computational resources (i.e., multiple compute nodes);
- a set of *I/O servers* performing operations over the data partitioned on the storage layer.

More details regarding the overall architecture and the single components are provided in [**?**]. In the context of the *EUBra-BIGSEA* project, an Ophidia cluster is composed of: (i) a single server node dedicated to the front-end and (ii) multiple I/O & compute nodes used to host both the framework and the I/O servers (the so called *super-node* configuration). The storage resources are shared among the various I/O nodes. This deployment schema is shown in Figure **??** and guarantees a good balance between the scalability of the cluster and deployment simplicity.

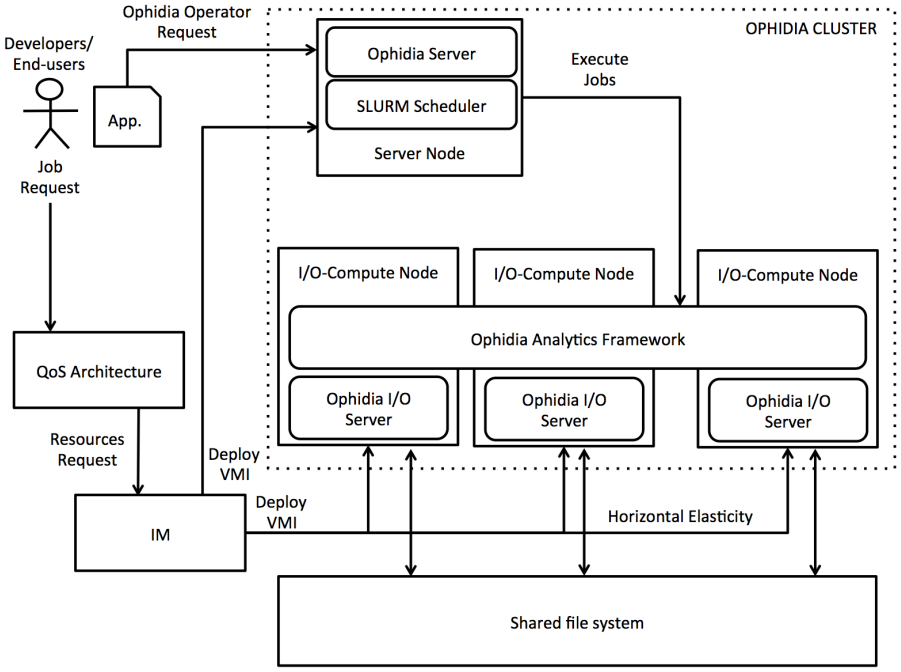


Figure 4: Ophidia deployment schema

In the context of the project, Ophidia is mainly used to extract statistical aggregate information for public transportation data and to provide a Dashboard for a better planning (the City Administration Dashboard). Moreover, it has been strongly integrated with the other services and technologies provided by the *EUBra-BIGSEA* platform to tackle data processing in QoS-based scenarios on cloud IaaS infrastructure, as well as security and data privacy. Additionally, Ophidia has been integrated with the COMPSs programming framework into Python-based applications through the combination of PyOphidia and PyCOMPSs (COMPSs Python bindings) modules. This integration allows exploiting the features provided by Ophidia and COMPSs for concurrent data processing, offering, at the same time, an increased programmability to the end-users.

To address QoS-based scenarios, the framework has been extended from several points of views, i.e. to support the elastic deployment of the cluster, monitor the job and instance status, ease the scaling of the resources through a better decoupling of the I/O from the storage layer and provide a more balanced scheduling of the jobs at the level of the resource manager. Additionally, Ophidia has been integrated into the cloud services provided by the project to fully support QoS guarantees. To this end, an *Ansible role* has been developed [**?**] to automate the dynamic deployment of an Ophidia cluster. Such role is managed by the *EC3* and the *IM* (see section **??**). The initial deployment of the cluster starts from a request issued to the *pro-active policies* services (i.e., the *broker API*), while the *optimization service* provides the initial configuration size.

3. Security and Privacy Model

Static and dynamic data analytics require complex infrastructures, which are always a challenge in terms of security. The type of applications to be supported in this work is also a challenge in itself, as they deal with large amounts of heterogeneous and complex data produced very quickly by a high number of diverse sources. Traditional treatment of data, from security to transformation, may be inefficient and inadequate. Thus, the system requires efficient mechanisms to ensure privacy and security, in a scalable fashion.

It is well known that the security concerns of a large and complex system should not be addressed individually or in an ad-hoc manner, as this may result in insufficient solutions.

The defined solution is based on three key pillars: (1) An *Authentication, Authorization and Accounting* (AAA) solution, flexible enough to provide functionalities both at the infrastructure management level and to serve the end users of hosted applications; 2) a *security assessment* of key infrastructure components, leading to the development of solutions for the uncovered issues, and the characterization of the trustworthiness of the system; and (3) two distinct **privacy control barriers**, which are responsible for protecting the anonymity of both the raw data to be used and the data resulting from the predictive and descriptive models built.

The **trustworthiness characterization** supports security measurement and includes the assessment and improvement of infrastructure components [**?**], the benchmarking and improvement of intrusion detection systems, and the proposal of metrics to characterize the trustworthiness of the system. The techniques of *field measurement*, *robustness and security testing*, *vulnerability and attack injection* were applied in the assessment of the components of the architecture shown in Figure **??**, that are most exposed to attacks and faults, namely: *COMPSs*, *OpenStack*, *Docker*, *virtualization layer*, *Intrusion Detection Systems (IDSs)*, and *NoSQL databases*.

The results showed that COMPSs mostly provides a robust interface, except for very rare situations; OpenStack has most of its concerns related to insider threats. Docker is still prone to issues of privilege escalation and bypass, while the virtualization layer is mature and secure nowadays, but some problems still arise when users have complete control over one machine. The analysis of IDSs showed the continuous need for evaluation, comparison and improvement of the adopted solutions. Most of the experiments NoSQL databases revealed integrity issues in the data. These results supported the identification of better configurations in terms of security, the potential mitigation of some of the identified vulnerabilities, and the estimation of the level of trustworthiness of the assessed components. The key risks behind the observed weaknesses have been identified, and the respective mitigation strategies have been defined and put in place. The obtained information also allows the adjustment of the quality of protection established from the provider point of view, thus obtaining a realistic measure of what level of security can be promised. The evaluation of the final solution showed an estimation of a high level of trustworthiness.

The techniques developed towards achieving the goals of AAA and privacy resulted in two specific tools, which have been named *AAA-as-a-Service* (presented in Section **??**) and *Privacy-as-a-Service* (presented in Section **??**) respectively.

3.1. Authentication, Authorization and Accounting as a Service (AAAaaS)

AAAaaS (Authentication, Authorization and Accounting as a Service) provides the general functionalities of traditional AAA and Identity and Access Management (IAM) services. Additionally, it is possible to include interfacing with external identity providers. The software is deployable and manageable according to three fundamental cloud principles: scalability, elasticity and resilience.

The solution is based on a RESTful service developed in Python and uses `MongoDB` database because it is open source, document oriented and provides fast performance. It does not use schema and therefore it provides more flexibility than relational databases. `CloudFlare SSL (CFSSL)` is the tool selected to generate and manage the certificates that can be used for communication between the RESTful service and the database. This way, all internal communication is encrypted.

Through the Application Programming Interface (API) or the web pages (for authentication) the solution provides the following functionalities:

- Authentication – sign in, token verification, read user information, sign up, sign out, update user information, delete user account, change password, reset password, resend account confirmation email.

- Authorization – create rules, update rule, show rule, delete rule, use resource.
- Accounting and other features – traditional accounting (i.e. read accounting of a user) and also other available actions such as creating email associations, reading email associations, deleting email associations.

Figure ?? presents an overview of the defined AAA architecture. It has been designed in such a way that is suitable to be maintained or further developed (due to its Open Source characteristics) in a DevOps fashion if necessary. This web application handles all the HTTP(S) requests made to the service. Every request is then validated using secure methodologies. For instance, passwords are encrypted with SALT functions. Passwords must fulfill three out of four conditions (e.g. minimum length, letters, numbers, capital), they cannot be the same as the user name, as well as other criteria. As mentioned before, the validation process which queries the database can also be secured with SSL certificates. The service is based on tokens, which are randomly issued at each sign-in session and have an expiry date that can be up to seven days (when the "stay signed in" option is checked). After the expiry date, tokens are no longer valid. Thus, a new sign in is required.

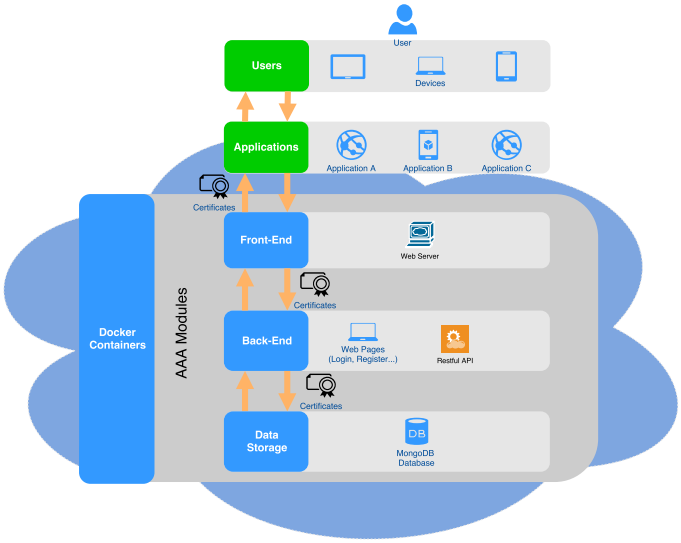


Figure 5: AAAaaS Architecture Overview

As we can observe, the front-end layer is based on an `Nginx` web server, acting as a reverse proxy redirecting all communications to the web application. With Nginx, we introduce an additional layer to the service, that provides load balancing and resilience capabilities. It is possible to load SSL certificates to ensure secure communications with all clients through HTTPS. By providing the location of the web application instances, the requests can be redirected according to the introduced settings (e.g. instance weights, least-connected or other settings).

EUBra-BIGSEA provides all services as containerized solutions. In the case of the AAA, this implies the three main components: web server, web application container and database containers. The architecture of our service (Figure ??) represents the interaction between the containers in the Cloud. The Front-End block includes a Docker container with Nginx acting as a reverse proxy and redirecting all the traffic to the web application container Back-End. In turn, the web application container queries the database container represented by Data Storage. The use of containers allows several instances of our components (e.g. web application) to provide a scalable solution. Also, the possibility to perform health checks internally (through Nginx) or externally (e.g. Marathon) allows us to provide a resilient solution capable of monitoring the components.

AAAaaS also provides the iAA (infrastructure Authentication and Authorization). iAA deals with the authentication and authorization of infrastructure accesses instead of applications, services or end-users. It also provides a graphical user interface, as well as a RESTful API. The iAA module provides an end-point so Mesos agents or frameworks can authenticate themselves and gain clearance to access certain resources. The module is working as a middleware between Mesos agents or frameworks and the Mesos Master. It can be easily adapted to support different frameworks

executed from the Command Line Interface (CLI) and it allows changes (e.g. updates) to be made to the Mesos system without having a disruptive effect on the iAA process.

The iAA control is carried out in accordance with the authorization mechanisms (i.e. credentials and Access Control Lists (ACLs)) available on the Mesos Master. In order to achieve this scenario, a mapping between the credentials created by the user and a set of credentials previously loaded on the Mesos Master is provided. This means that each registered user is assigned a pair of Mesos credentials (*principal* and *secret* in the Mesos terminology). This action is completely transparent to the user.

3.2. Privacy as a Service (PRIVAaaS)

Privacy management is a key issue when dealing with citizens' data. Privacy preservation and knowledge extraction is typically a trade-off, so it is important to understand the risk of privacy leakage when processing data. In this context, *PRIVAaaS* (*PRIVAcy as a Service*) [**?**] is a software toolkit that allows controlling and reducing data leakage in the context of Big Data processing and, consequently, protecting sensitive information that is processed by data analytics algorithms. *PRIVAaaS* is based on anonymization policies, i.e. it performs data anonymization by enforcing the rules specified in the predefined policies. This allows improving, throughout the anonymization process, the privacy laws compliance as well as the compliance of the privacy requirements provided by the data source owners.

PRIVAaaS provides multiple anonymization phases and its integration in big data analytics platforms allows performing data anonymization at the several stages of data processing, properly targeting data privacy regulations and policies during the whole data life-cycle and smoothing the trade-off between data privacy and data utility. Figure **??** shows the *PRIVAaaS* general architecture.

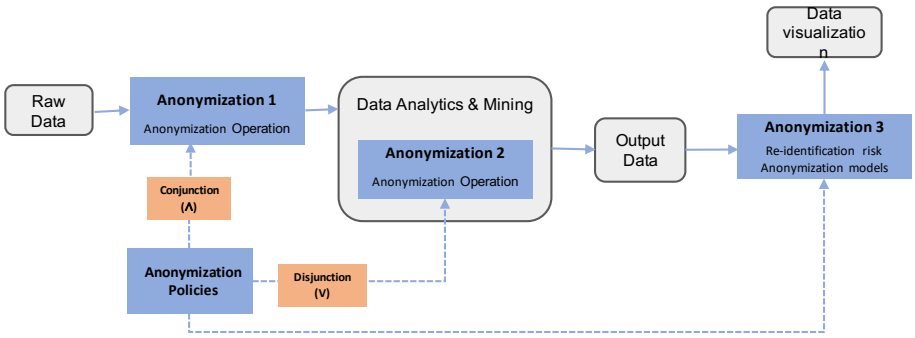


Figure 6: *PRIVAaaS* general architecture

In Figure **??**, *anonymization policies* surround what type of data fields must be anonymized and how. These policies may be based on privacy principles and laws (e.g., General Data Protection Regulation - GDPR), as well as specifications by data source owners.

Anonymization 1 is the phase where *raw data* is anonymized. Before anonymization, a *conjunction* of the anonymization policies is determined. The conjunction consists in verifying the set of policies field by field and sorting out those requiring anonymization. Then, an *AND* operation is applied to group similar fields that require anonymisation as a whole. This operation is applied only if all the fields require anonymization. This conjunctive process results in the less restrictive anonymization, maximizing the data utility in this phase.

Anonymization 2 is the phase where anonymization is applied during the *data analytics and mining* processes on intermediate results. Before anonymization, a *disjunction* of the policies is performed. The disjunction also consists in verifying the set of policies field by field, sorting out those requiring anonymization. However, at this time an *OR* operation is performed, which implies that all the fields must be anonymized according to the policies even if a single policy has done this configuration. This disjunction process results in most restrictive anonymization, guaranteeing that the protection established by the policies is accomplished in this phase. Data re-identification is the practice of matching anonymized data with publicly available information, or auxiliary data, in order to discover the person the data belongs to. Even when data is anonymized, some privacy attacks (e.g., background knowledge attack) can lead to data re-identification.

In the **Anonymization 3** phase, the final *output data* produced by the data analytics is evaluated regarding the re-identification risk and, if necessary, its anonymity level is increased in order to reduce this risk. This is done through the application of anonymization models (k-anonymity, l-diversity, etc.) before data is available for visualization.

PRIVAaaS provides a library and a REST service to perform data anonymization. When implemented as a service, it may be adapted to different platforms with less effort, addressing interoperability issues, and it may take advantage of the *EUBra-BIGSEA* infrastructure to scale.

In the case of Anonymization 1 and Anonymization 2, the two phases are similar from an implementation standpoint, so the same tool is used in both cases. The library/service receives two files as input: the data set to be anonymized and the anonymization policy (which can be the result of conjunction or disjunction process, for each respective phase). The policy must specify the fields to be anonymized and the anonymization technique that must be applied to each field. Then the library/service applies the anonymization techniques according to the policy and provides, as output, the file with the anonymized data set. The techniques that have been implemented in *PRIVAaaS* are *generalization* (attributes are replaced by some more generic ones, that are faithful to the original); *suppression* (attributes are completely removed to form the anonymized dataset); *encryption* (cryptographic schemes are applied to replace attributes with encrypted data) and *perturbation/masking* (attributes are replaced by dummy data).

The Anonymization 3 phase implementation also receives, as input, a data set to be anonymized (resulting from data mining and analytics, which would be released from the platform) and the anonymization policy. In this case, the policy must also define a *re-identification risk threshold*. The re-identification risk means the highest risk (in percentage) that a record can present among all records in the data set; in this work, it is calculated based on ARX tool functionalities [**?**]. So, the threshold will be used to decide whether the re-identification risk of the input data is acceptable or not.

After receiving the inputs, the re-identification risk of the data set is calculated. Then, a verification is performed: if this risk is higher than the threshold established in the policy, the value of k, from k-anonymity algorithm [**?**], initially set as $k = 2$, is increased and k-anonymity is applied with this new value of k. This is performed successively, until the re-identification risk is equal to or lower than the threshold (the higher the k, the lower the risk). When the threshold is reached, the data is made available for visualization outside the platform.

4. Cloud services

4.1. Elasticity

The *EUBra-BIGSEA* architecture implies a large set of components that interact together in a distributed infrastructure. For the convenience of the deployment, it has been automated using Ansible[**?**] recipes as Ansible roles and using the Infrastructure Manager[**?**] to interact with the cloud IaaS. The Infrastructure Manager is a TOSCA[**?**]-compliant platform-agnostic cloud orchestrator. By means of the Elastic Compute Clusters in the Cloud (EC3) tool [**?**], a complete cluster can be set up with minimal intervention. The client and the recipes are available as a Docker image [2] and in a GitHub [3] repository.

The *EUBra-BIGSEA* platform provides off-the-shelf automatic horizontal elasticity based on Cluster Energy Savings (CLUES) [**?**]. CLUES powers on and off resources as requested by an agent that polls the resource manager queues (Mesos[4] in the case of *EUBra-BIGSEA*). This elasticity is combined with the vertical elasticity provided by an actuator at the level of the hypervisor and a similar actuator at the level of the Marathon and Chronos frameworks [**?**]. In these cases, a fine-grain monitoring evaluates the progress of the application (which can be automatically obtained from Spark runtime) and changes the allocation of resources at the level of the CPU CAP or the framework request, speeding-up or down the jobs to optimize resources and fit the deadline.

4.2. Performance Prediction and Optimization Services

The Performance Prediction Service (PPS) is a key component in the *EUBra-BIGSEA* architecture both for planning and managing purposes. Indeed, the PPS is used by the optimization

[2] Docker Hub reference
[3] github repo of ec3client
[4] http://mesos.apache.org

service to identify the minimum number of nodes or cores to run an application within a specified deadline and it is also triggered by the proactive policies module to estimate the residual execution time of running jobs. This way, proactive policies can drive the automatic system reconfiguration to meet the applications dynamic needs, avoiding Service Level Agreement (SLA) violations.

The PPS goal is to efficiently estimate the *average execution time* of a target application (implemented on COMPSs, Spark, Tez or MapReduce), given the available resources. Given a target application, specified by a directed acyclic graph representing the individual tasks and their parallelism and dependencies, the purpose is to predict how long it will take for the application to run (on average) on a given resource deployment (described in terms of numbers of cores or nodes for instance).

PPS is based on an analytical queuing network (QN) model originally proposed in [**?**] for performance prediction of parallel application, which extends an Approximated Mean Value Analysis (AMVA) technique by modeling the precedence relationships and parallelism between individual tasks of the same job. This model explicitly captures the overlap in execution times of different tasks of the same job to estimate the average application execution time.

In [**?**], we demonstrated that the PPS is very accurate (the average absolute percentage error is around 2-8%) and can provide estimates in the order of milliseconds.

The Optimization Service of *EUBra-BIGSEA* is aimed at pursuing the respect of QoS guarantees and reducing the resource usage costs (e.g., due to energy).

Given an application, we characterize its deadline as *hard* or *soft*. Hard deadlines must be fulfilled. Soft deadlines have an associated priority and can be violated if the system does not have enough capacity. The Optimization Service implements two main functionalities: (i) it is able to provide the initial minimum capacity configuration for an application in a way that its QoS objectives can be achieved, (ii) under heavy load, it can determine how to re-balance the applications capacities (by reallocating the cluster nodes) by minimizing the weighted tardiness (i.e. the weighted sum of application exceeding time wrt. deadlines) of soft-deadline applications. In both cases, the optimization service implements a hill-climbing-like parallel local search, which adds/removes capacity or moves capacity from one application to another in the minimum tardiness scenario; it also evaluates the impact on the total application execution cost/tardiness by relying on PPS to estimate the performance impact.

The preliminary analyses on the *EUBra-BIGSEA* case study demonstrated that the optimization Service is effective to identify the optimal application capacity and can efficiently support the Proactive Policies module (heuristics algorithms run in few minutes on an eight cores commodity server).

4.3. Proactive Policies

As ensuring QoS is one of the major goals of the *EUBra-BIGSEA* architecture, the services that estimate the resources needed to complete a job by a desired deadline (described in the previous section) are orchestrated by a system that monitors progress and can trigger immediate adjustments when jobs are running late. The main components of the system are the Broker, the Monitor and the Controller. The components are described below.

The *Broker* component receives requests from a command-line interface or from the Lemonade GUI and interacts with other services to request an estimate of the necessary infrastructure resources and to check authorization levels. It also triggers the start of the application, for example, in a new or preexisting Spark cluster, as well as also two internal components that monitor and react to delays in the execution. The Broker is configured with the execution plug-in, which indicates the type of Big Data framework and the underlying infrastructure. Examples of Big Data frameworks in the scope of *EUBra-BIGSEA* are Spark and COMPSs. The supported underlying cloud infrastructures are OpenStack and OpenNebula.

Next, the *Monitor* component is responsible for mapping application-specific progress to a normalized progress metric. The *Monitor* is triggered by the *Broker*, receiving an endpoint for collecting application metrics in a generic (e.g., using Spark progress interface) or custom (e.g., querying an application-specific log or API) fashion. The progress is then published in a standardized form in the cloud monitoring system (Monasca[5]). The formatting of the progress metrics is defined, for example, through the plug-in specified in the job description in the case of a CLI submission.

[5] http://monasca.io/

Finally, the *Controller* consumes publications disseminated by the monitoring system referring to its application (according to information received from the associated instance of the Broker). The *Controller* will then react when the application progress is deviating from the expected progress. By default, the *Controller* will use a hysteresis control that triggers a vertical scaling to the next instance size (considering IO capability and CPU speed). Nevertheless, the *Controller* can be customized through three plugins: the Controller plugin, which defines the actual control algorithm, such as the Hysteresis control mentioned above; the Actuator plugin, which defines how the scaling actions will be implemented in the infrastructure; and the Metric Source plugin, defining the source for the monitoring information. The default Metric Source plugin is Monasca. Two different actuator plug-ins are available: the actuator that works at the platform level (adjusting the resource allocation at the framework) and the actuator that works at the level of the infrastructure (adjusting the CPU CAP of the Virtual Machine). Examples of Actuator plug-ins are the following: Chronos (framework level), for repeatable tasks; Marathon (framework level), for long-living tasks; and KVM-over-OpenStack and KVM-over-OpenNebula (infrastructure level), when the adjustment of application performance will be done by adjusting the performance of the underlying VMs (e.g., disk IO per second and CPU speed cap).

5. Applications

The increasing urban population sets new demands for mobility solutions. The impacts of traffic congestions or inefficient transit connectivity directly affect public health (emissions, stress, for example) and the city economy (deaths in road accidents, productivity, commuting etc.). In parallel, the advances of technology have made it easier to obtain data about the systems which make up the city information systems. The result of this scenario is a large amount of data, growing every day and requiring effective handling in order to be transformed into integrated and useful information. Related applications may include speed limit enforcement, wheelchair route planning, traffic accident diagnosis, noise studies, among others.

Consider, for example, that it is necessary to plan the speed limit around a city. The Brazilian traffic enforcement legislation [6] [7] states that before and after installing the devices, technical studies involving the accident history should be carried out to assess their necessity and efficiency on site. For speed humps, a minimum distance between them is also foreseen, as well as restrictions to their installation on turns and pathways of the regular lines of collective transport (buses, for example). Similar regulations[8] can be found in other countries as well[**? ?**].

Figure **??**-A shows the location of speed cameras by neighborhood in Curitiba. Dark colors indicate a larger number of cameras. Figure **??**-B shows the result of the in-situ study along with the route of the bus line and the map of the city by the OpenStreetMap. The Downtown district (the darkest in the figure) has the highest number of speed cameras (26), followed by six districts in the south with less than half of the cameras (10 radars). By including the date of installation of the equipment in the existing data and an integration with accident data, the use of a GIS similar to the one developed here may facilitate the technical studies or allow these legal conditions to be verified.

As a second example, consider that the mobility data is integrated with traffic accident diagnosis, in order to understand which region in a city has the majority of historical record of accidents. Figure **??** presents the heatmap of accidents in the CIC neighborhood in Curitiba. The same figure presents the locations with the highest number of injuries, along with the locations of hospitals and health care units. Figure **??** presents another important issue: the post-cash care is impacted, since one out of the three hospitals located in the area, one was already closed, the other manages neurology specialties and the third is a small-sized hospital.

The computation of this information in a city (using historical data) may prevent accidents. Several other considerations might be suggested in order to reduce traffic accidents (not only in CIC, but in general):

1. Building a trustable database of injuries;

[6] http://www.denatran.gov.br/images/Resolucoes/Resolucao6002016_new - Last accessed on Jan. 7th, 2017.
[7] http://www.denatran.gov.br/download/Resolucoes/RESOLUCAO_CONTRAN_396_11 - Last accessed on Jan. 7th, 2017.
[8] http://www.state.nj.us/transportation/eng/documents/speedhumps/ – Last accessed on Nov. 27th, 2017

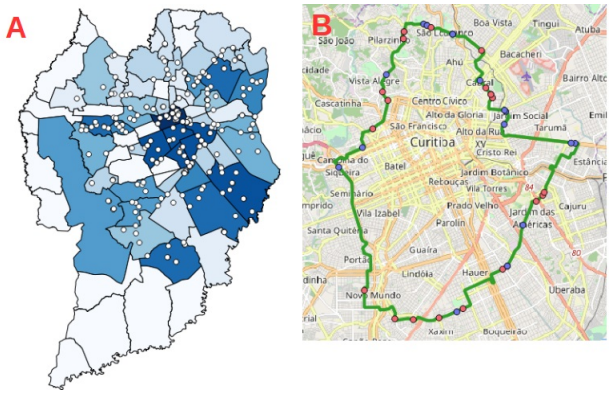


Figure 7: Location of speed cameras by district in Curitiba (A); and distribution of speed cameras (blue) and speed humps (red) in Interbairros II bus line (B).

2. Implementing the same methodology of approach by the entities responsible for collecting data, so that homogeneous information can be generated;
3. Including road signs where they are missing, and building and improving the infrastructure available to pedestrians;
4. Providing regular inspections so that the regulations are followed; and
5. Promoting population awareness.

Figure 8: The heatmap of accidents in CIC, zooming in on an interest area.

The results unveil challenges to overcome regarding file formats, reference systems, precision, accuracy and data quality, among others, that still need effective approaches to ease open data exploitation for new services.

5.1. End-user applications

On top of the *EUBra-BIGSEA* platform, several applications have been developed to process city transportation data.

5.1.1. Routes4People

The Routes for People Web application demonstrates the capabilities of our jointly developed infrastructure by directly or indirectly using the work done in other parts of the project. It consists of multiple services embedded in containers that communicate with each other through a Load Balancer proxy. Figure ?? displays the general architecture of the Routes for People Web application that runs on *EUBra-BIGSEA* infrastructure, along with its dependencies. We only present direct dependencies to simplify the schema.

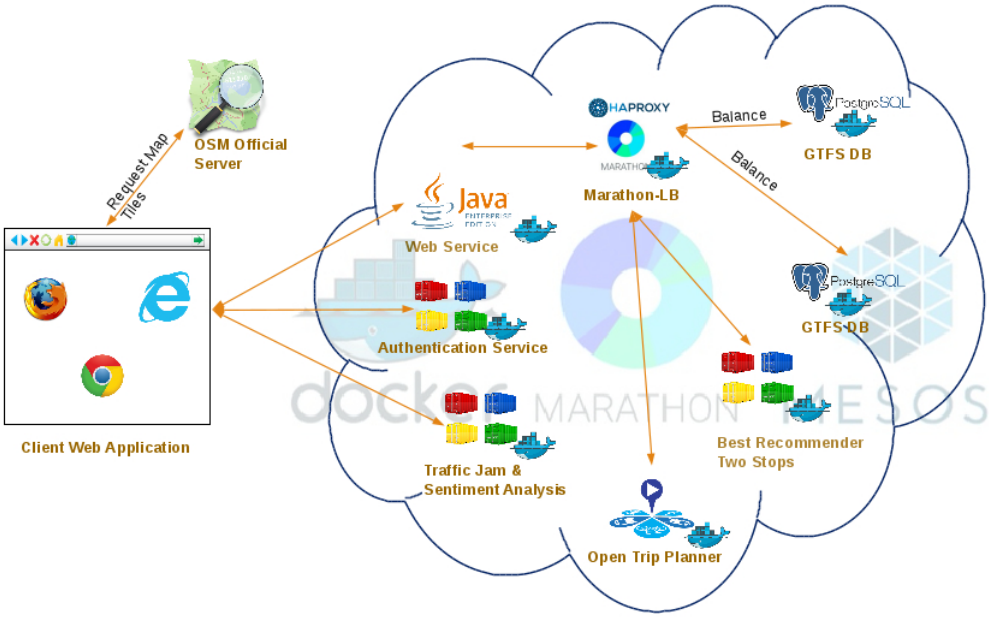


Figure 9: General architecture of the Routes for People Web and dependencies

The Routes for People web application runs on the user's browser. It connects our infrastructure to multiple services:

- The authentication service responsible for managing user identities;
- The traffic jam and sentiment analysis services that can be invoked on supported cities, showing a layer of intensities;
- The Java webserver responsible for everything else.

The Java web server retrieves information like the transportation stops, the actual routes, and their schedule from PostgreSQL GTFS database instances. These instances are load balanced by Marathon-LB, a utility that connect HAProxy with Marathon.

The web application contains four tabs (Figure ??). The *Trips* tab shows an OpenStreetMap (entered on a city we support in the project) on top of which the local transportation stops are drawn in clusters. This tab offers additional functionalities by means of the buttons in the top-left corner: user's geolocation, creating trips between two stops, viewing layers on the map, re-centering the map, sentiment analysis layer, and traffic jam layer. The *Routes* tab lists the transportation routes available for the selected city. Our users can also draw the route on the map or view the schedule for each route, using the buttons on the right side. The third tab, *Favorites*, becomes available once the user logs in. This tab contains the user's selected routes and trips for a city, along with the option to eliminate them. Finally, the *More* tab holds some additional features like city selector, language switch, log in, help, contact, and feedback. The last two features are only available after the user has successfully authenticated.

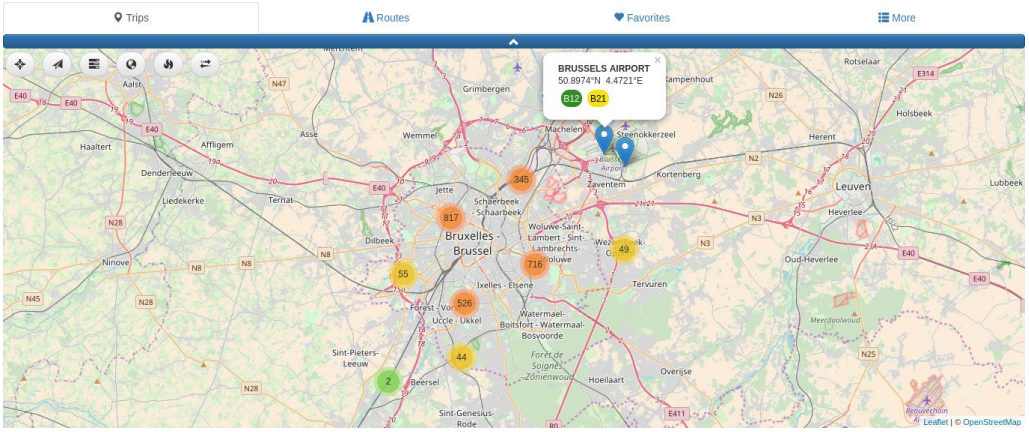


Figure 10: The first tab of the Routes for People Web application

5.1.2. Municipality Dashboard

The *City Administration Dashboard* application is aimed at identifying aggregate statistical trends in the bus usage that can be potentially exploited by the municipality for urban management and planning purposes (the application focuses on the city of Curitiba, in Brazil). Several components from the *EUBra-BIGSEA* platform are used to process the raw input data and create aggregate statistics by taking security, data privacy and QoS constraints into consideration. Figure **??** shows an overview of the system architecture related to the City Administration Dashboard application, the core building blocks and how they interact with each other, the security and privacy big data services extensions as well as the links both to the QoS and AAAaaS infrastructures. Given its complexity and the high number of components involved, this application represents a comprehensive example to demonstrate the data platform features and the level of integration among the different modules. In particular, the components exploited by this application are: Ophidia, Spark, HDFS, COMPSs, the *EUBra-BIGSEA* QoS infrastructure (e.g. Broker API, EC3/IM, etc.) and the security and data privacy services (i.e. AAAaaS and PRIVAaaS).

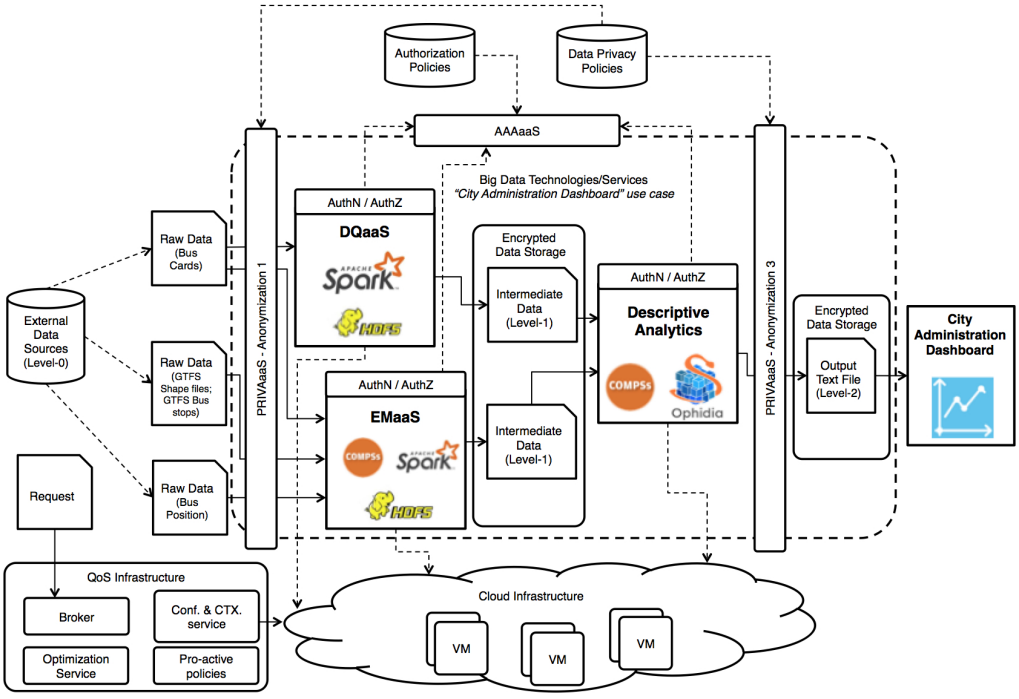


Figure 11: City Administration Dashboard - system architecture overview

The input data of the City Administration Dashboard application are related to the city of

Curitiba. More in detail, they are: *bus cards database*, *bus GPS position database*, *General Transit Feed Specification (GTFS) shape files* and *GTFS bus stops files*. The application performs several steps; a first anonymization stage on the bus cards input data (so called *level-0* data) is carried out through the PRIVAaaS tool; the output of this step, along with the remaining level-0 data, are used for the execution of the DQaaS and EMaaS to produce intermediate data (hereafter *level-1* data). The storage layer used by this application is provided by HDFS, which stores a heterogeneous (e.g. in terms of data format, data model) set of level-0 data sources.

Various types of intermediate data (level-1) are produced before the execution of the descriptive analytics model in the first stages (like pre-processing or ETL steps) of the application. The DQaaS produces *data quality-enriched bus card data* by adding additional fields to the original data sources to annotate the quality of the data. This information is used by Ophidia, in subsequent steps, to filter out, from the statistics computation, records which do not comply with the quality requirements. The data quality information annotated by the application includes: *timeliness* (the extent to which data are temporally valid), *completeness* (the degree to which all values are registered in the dataset), and *consistency* (highlights if the data dependencies are satisfied). The EMaaS produces *Enriched Historical Bus GPS Data* instead, by applying entity matching algorithms to the input data (to identify the bus trips) and enriching them with the bus stops as well as with the number of passengers boarding at each stop. These data provide, among others, information about the bus routes, their position and the number of passengers per bus stop along the route. After an additional ETL stage, the intermediate data are used as input for the descriptive analytics component developed with COMPSs and the Ophidia framework, which in turn produces the aggregate statistics (so called *level-2* data). Finally, these data are subject to an additional anonymization step through, once again, the PRIVAaaS tool (re-identification risk component) before being made available to the web application of the City Administration Dashboard for visualization purposes. Various types of output (level-2 data) can be produced according to the type of aggregation and metrics of interest:

- *Bus line-aggregate statistics* include aggregate information about the bus lines usage. For each bus line and time range, the minimum, maximum, mean and total number of passengers are computed. Statistics can be computed over different time ranges, such as the whole month, week, weekday (i.e., Monday, Tuesday, etc.), day or hour, or with different level of aggregations, such as for each bus line or over all bus lines (to get an aggregate view of the entire bus transportation system).

- *Bus user-aggregate statistics* include aggregate information about the bus users' usage. For each bus user, the minimum, maximum and total number of times the passenger took the bus and the number of days he/she took a bus in the given time range are computed. Statistics can be computed on a weekly or monthly basis.

As stated before, data privacy has been addressed at multiple levels using the policy-based PRIVAaaS component. More in detail:

- some anonymization techniques (e.g. encryption) are applied to those level-0 data (i.e. bus card data) that expose sensitive information, in order to remove the direct reference to the actual bus card user;

- the k-anonymity algorithm is applied to the output data produced by some types of statistical aggregation to reduce the re-identification risk; these data can actually include fields that might be used (even in combination) to re-identify the original user.

In terms of user authentication and authorization, all the data components and services used by this application (i.e. Ophidia, EMaaS and DQaaS) verify the validity of the token and the authorization rules through the AAAaaS module, before granting permission for the actual requested processing. Before running the application, the QoS infrastructure services are contacted, to perform the initial cluster deployment and setup, as well as during the application execution for the monitoring and, eventually, the dynamic resource adjustment to fit the QoS deadlines. In particular, in the case of Ophidia, the request is managed by a specific plugin for the Broker, which deploys the cluster through EC3 and IM based on the feedback coming from the Optimization service.

The output of the descriptive analytics component is produced by applying a sequence of multiple and parallel Ophidia operators. In particular, the key exploited operators are related to: data subsetting, time aggregation (for statistics computations), dimensionality reduction and data

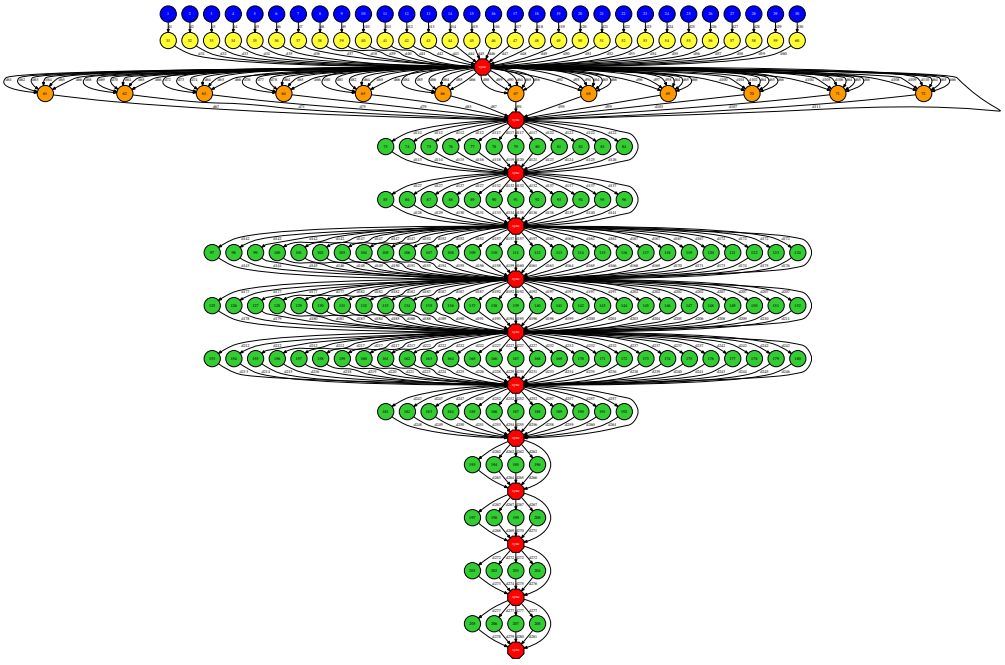


Figure 12: COMPSs simplified execution graph example (set of statistics computed by the application)

import/export. At the level of the programming models, the COMPSs parallel runtime engine is used to concurrently run the code calling the Ophidia operators. The descriptive analytics component is developed in Python using the PyOphidia and PyCOMPSs modules. In terms of benefits, COMPSs transparently parallelizes applications exploiting their inherent parallelism without the burden of parallel code implementation, while Ophidia provides support for a wide set of efficient parallel data analytics operations. Their integration into the QoS framework allows addressing QoS-based scenarios.

Figure **??** shows an example of a simplified execution graph (from the COMPSs perspective) of a set of statistics computed by the application (i.e. bus line-aggregate statistics). Each circle represents a COMPSs task executing a block of code with one or several instructions. The first phase relates to the application of data anonymization to the input data (blue circles), followed by the extraction (yellow circles) and the transformation (orange circles) steps of the ETL phase, whereas the following 10 stages refer to blocks of Ophidia operators (green circles). Each of these set of circles represents the computation of a different type of aggregation (both temporal or based on the bus line), while each circle represents a specific statistics computation (e.g. average for Monday, Tuesday, etc.). A more comprehensive benchmark and experimental results on the application are out of the scope of this paper.

5.1.3. Melhor Busão

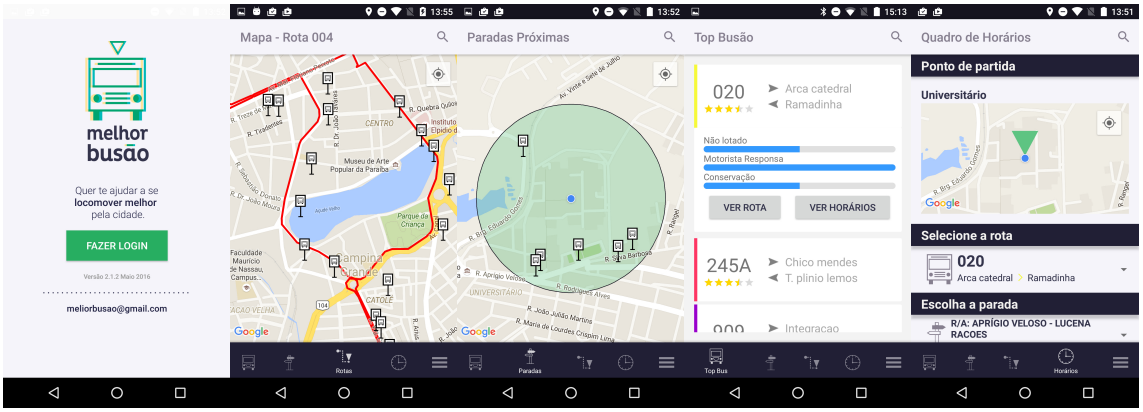
Melhor Busão (a Portuguese expression for Best Bus) is an application designed to present the results of the Transportation Data Processing and Analysis performed by the *EUBra-BIGSEA* infrastructure to transit users. It is implemented as a mobile application so as to simplify access and notifications when relevant information is available.

In its essence, *Melhor Busão* is an Advanced Traveler Information System, helping passengers to make a better use of the city Public Transportation System by providing both static information, such as routes and bus schedules, and – most interestingly – dynamic information about predicted trip duration and crowdedness for a planned trip. By predicting such trip features, the app allows users to make a more informed decision on which itinerary to take according to their priorities. Ultimately, forecasting multiple criteria about trips aims to help users increase their degree of satisfaction during the trip, a trend which has received increasing attention in the literature [**?**],[**?**].

In order to provide the user with these trip feature predictions, Melhor Busão relies on several applications and high-level services developed throughout the project. Such services have

been implemented on top of the above mentioned data analytic services of *EUBra-BIGSEA*. The application sends a request with user trip plan information to a Web Service named Best Trip Recommender (BTR) API. This Web Service has the most up-to-date version of the prediction models and runs the model with the received user info, returning the prediction results to the app in order for the user to visualize them. BTR API is also responsible for updating the models on a regular basis. For that purpose, it dispatches Spark jobs to the cloud data processing services of *EUBra-BIGSEA* with a deadline to be met in order to attend the model update frequency requirements. The models are trained with features built on top of the result of the Entity Matching service which processes the raw GPS and Ticketing data.

Figure ?? shows some screenshots which depict login, route map, nearby stops, top bus, and bus trip features prediction.



Login Screen | Route Map | Nearby Stops | Top Bus Screen | Trip Planning

Figure 13: MelhorBusão App Screenshots

The application is available in Portuguese language for the Brazilian cities of Campina Grande and Curitiba, and displays information obtained from several data sources, including: Bus GPS streaming data, City GTFS file and Ticketing Records from city buses. GTFS is the main source for the static data presented in the application (e.g. routes, shapes, bus stops, etc.). The GPS and Ticketing data is processed by back-end applications which run in the cloud using a parallel architecture to achieve faster processing.

Melhor Busão is, thus, highly integrated into the whole project architecture. It uses the security solutions to perform authentication in the app and provide security to users transactions; the Entity Matching algorithms to match bus GPS records to GTFS route shapes and thus to estimate when a bus passed by each stop during the day, and to match passenger boarding to bus GPS records, identifying which bus each passenger boarded on; it uses the data analysis solutions to provide the passenger with estimated trip duration and bus crowdedness for a given trip plan, by calling the Best Trip Recommender API; and the cloud infrastructure and services, on top of which all back-end services run.

5.2. High-level services

The applications described in the previous section rely on a set of services implemented on top of *EUBra-BIGSEA* services that provide a higher-level functionality for application building, such as static and dynamic route matching, route extraction, crowdedness estimation, etc. The most relevant services are described along this section.

5.2.1. Entity Matching as a Service

Geographical coordinates and maps (digital or paper-based) are a common feature of our daily life in order to provide a two-dimensional representation of geographic features in the real world, such as parks, bus stops, roads, rivers, buildings, and places. Such information is often referred to as geospatial or geographical data and plays an essential role in many governmental, economic and social domains, such as disaster response, urban planning and tourism [?]. Since the quality of life in a city greatly depends on the well-being of its citizens, the municipalities invest in information systems that assist the citizens (e.g., regarding urban mobility or the identification of points of interest) directly or indirectly [?]. In this sense, the large amount of data collected by

municipalities and map projects (e.g, OpenStreetMap) may be used to improve the efficiency of public transportation and infrastructure investments taking into account the proposition of new solutions or modifications to the current infrastructure. However, since geospatial data are prone to inconsistencies and quality issues, it is important to apply sophisticated Data Quality (DQ) approaches, such as Entity Matching, before using them in order to take valuable strategic decisions. In fact, an analysis based on incorrect information can lead to wrong decisions [**?**].

In the context of *EUBra-BIGSEA* services and resources, an Entity Matching as a Service (EMaaS) has been developed to address important problems of the data acquisition and descriptive models of geo-spatial trajectories use cases. The data acquisition problems tackled by EMaaS refer to the lack of accuracy and precision of official and non-official municipality data sources, which causes incoherences and unalignment of buildings, streets, and bus stops. EMaaS can support the detection and measurement of matching problems presented in the linkage of these data sources. Regarding the descriptive models, a fundamental abstraction are trajectories, i.e. the path traversed by each end user while using public transportation. Trajectories comprise not only dynamic spatial data, but also other types of data that enrich the trajectory information. Building such trajectories is a challenge by itself, since matching the various types of data to a specific end user trajectory may be tricky and demand advanced and complex techniques. Thus, some EMaaS approaches have also been developed to deal with trajectories matching and provide high-quality integrated geospatial-temporal training data to support the predictive machine learning algorithms for the predictive models utilized by the Melhor Busão application.

The Entity Matching as a Service (EMaaS) is capable of performing efficient (data-intensive) matching tasks using the programming models (described in Section **??**) and includes the implementation of the following main approaches:

- *BULMA* (BUs Line MAtching)

  Briefly, *BULMA* has been developed to address the task of identifying bus trajectories from the sequences of noisy geospatial-temporal data sources. It consists in performing the linkage between the bus GPS trajectories and their corresponding road segments on a digital map (i.e., predefined trajectories or shapes). In this sense, *BULMA* is a novel unsupervised technique capable of matching a bus trajectory with the "correct" shape, considering the cases in which there may exist multiple shapes for the same route (usual cases in many Brazilian cities, e.g., Curitiba and São Paulo). Furthermore, *BULMA* is able to detect bus trajectory deviations and mark them in its output.

- *BUSTE* (BUs Stop Ticketing Estimation)

  In order to enrich the *BULMA* output, *BUSTE* (BUs Stop Ticketing Estimation) is used to perform a time interpolation over the shapes (based on the *BULMA* output). Furthermore, *BUSTE* positions the bus stops over the interpolated shape and groups the passengers boarding according to each bus stop. In other words, the idea of *BUSTE* is to provide an estimate of the number of passengers boarding at each bus stop. *BUSTE* also provides rich and high-quality integrated geospatial-temporal data to support the City Municipality Dashboard application and predictive machine learning algorithms. Note that *BUSTE* receives anonymized ticketing data as input and produces enriched Historical Bus GPS data. This means that the *BUSTE* computation is not influenced by the presence of anonymized values in ticketing data.

- *MATCH-UP* (MATCHing of Urban Places)

  Regarding polygons (for instance, buildings, residential regions, parks, and forests) and points records (for instance, bus stops, points of interest, vehicle coordinates) matching, the similarities between the geospatial records can be measured through linguistic and geographic matchers. In this sense, *MATCH-UP* provides alternatives to execute efficiently the matching of polygons and points between official and non-official data sources.

The EMaaS architecture is depicted in Figure **??**. As we can see, the execution of all the approaches can be made through a Spark or COMPSs job (through the COMPSs interface). The *BULMA* output files are partitioned into n files assigned to COMPSs workers to be computed in parallel. The first step of BUSTE enriches the historical bus trips (generated by *BULMA*) by positioning the bus stops over the interpolated shape selected by *BULMA*. Afterwards, *BUSTE* groups the passengers boarding at each bus stop. Regarding the crowdedness prediction, i.e., a

feature of the Bus Trip Recommender application, it is also generated based on historical bus GPS data (generated by the EMaaS approaches *BULMA* and *BUSTE*). The prediction model is trained using a state-of-the-art machine learning technique based on Spark over the BUSTE output. Thus, the trained predictive model is used to predict future trip duration and crowdedness. *MATCH-UP* is used to address the problems of geospatial polygons and points matching.

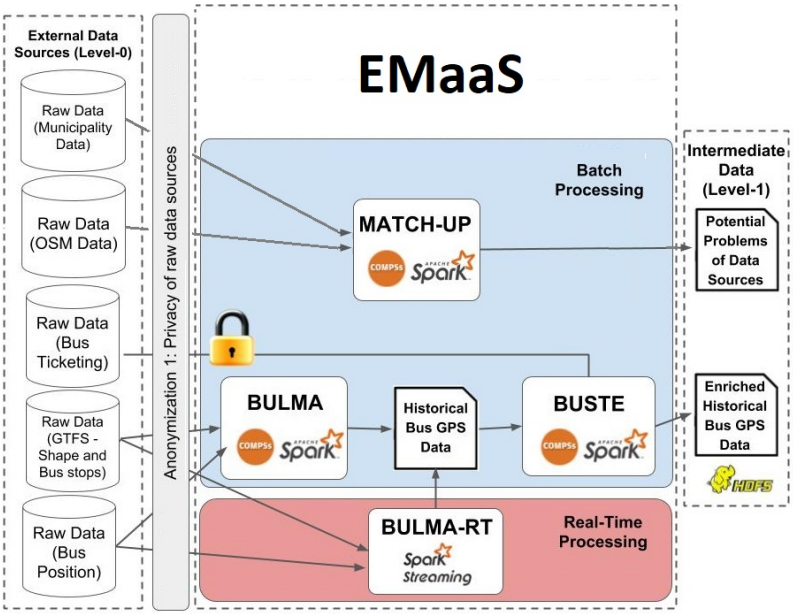


Figure 14: EMaaS Architecture

5.2.2. Trip duration and crowdedness estimation

Trip duration is a fundamental aspect of the user experience and it is often used as main goal when choosing how to use the transportation system. In other words, most people will try to get to the desired destination as fast as possible. Therefore, known applications like Google Maps[9] and Here[10], already provide trip duration information in order to support the user decision. Although this information can be helpful, in some cases the provided information is based on scheduled timetables which may not represent the true state of the system given its dynamic nature. In order to improve the quality of the information that is provided to the user, machine learning techniques were used to predict the duration and crowdedness of future trips based on past system performance.

The information used to build the predictive models is from Curitiba, Brazil. The data can be classified into three categories: (i) transit routes and schedules, (ii) real time vehicle location and (iii) passenger boarding data. The first one is available in GTFS format[11] and contains specification on how the service is expected to work under normal circumstances. The other two are collected on a daily basis while the system is working and can be used to assess the service performance. Real time vehicle location is available in GPS format and comprises the location of all buses every few seconds. Passenger boarding data contains information about every time a user boarded using a smart card.

All data goes through two preprocessing steps, where the first one is an entity matching process that is performed by BULMA and BUSTE, described in section **??**. The second step focuses on feature engineering and data formatting that builds the datasets used to create the predictive models. These datasets include categorical information such as trip route, shape, vehicle id, period of the day, weekday, week of the year, day of the month, month and also contains numerical information such as distance and number of passengers that boarded at the origin stop.

[9] https://maps.google.com
[10] https://wego.here.com/
[11] https://developers.google.com/transit/gtfs/reference/

Three machine learning algorithms have been tested and compared in order to choose the one with best results. The tested algorithms have been Lasso regression, random forests and gradient boosted trees. The best result has been achieved by the model built using gradient boosted trees with mean absolute error (MAE) of approximately 40 seconds for trip duration and almost 70 passengers for trip crowdedness.

5.2.3. Traffic congestion estimation

Traffic congestion is a frequent event in urban centers nowadays. It is often a consequence of the urban infrastructure not being able to keep up with the growth of the number of vehicles. Thus, it causes many drawbacks, such as stress, delays, and excessive fuel consumption. This application aims to identify traffic jams using data provided by Waze, an application widely used by drivers to obtain trajectories to destination or notifications regarding unusual traffic behavior, such as traffic jams, accidents or closed roads. To that end, we formulate a probabilistic graphical model equipped with Gaussian latent nodes.

In order to exploit spatio-temporal patterns associated with traffic congestion, we first discretize spatial and temporal dimensions. For the spatial dimension, we split the area of interest into an NxN grid. The temporal dimension, on the other hand, is discretized hourly. Here, we denote our variable of interest as $Y_{s,t} = -1, +1$, identified by the spatial grid cell s and the temporal index t, with $s = 1, 2, ..., N^2$ and $t = 1, 2, ..., T$, where T is the total number of hours from the dataset available for training. As indicated, $Y_{s,t}$ can assume two values: a negative value denotes that there is no traffic jam at cell s during time t, while a positive value denotes the opposite scenario. With each variable $Y_{s,t}$, we associate a latent (unobserved) variable $Z_{s,t}$. Letting x_t denote the proportion of neighboring cells experiencing some traffic jam at time t observed and latent variables are related as follows:

$$P(Y_{s,t}|Z_{s,t} = z) = \frac{1}{1 + e^{-z}} \tag{1}$$

$$Z_{s,*}\ \mathcal{GP}(\prime, \|_{\updownarrow\rfloor\dashv\updownarrow}(\sqcup, \sqcup') + k_{periodic}(t, t') + k_{adj}(x_t, x'_{t'})) \tag{2}$$

$$k_{local}(\tau = t - t') = \theta_A^2 * exp(\frac{\tau^2}{2\theta_B^2}) \tag{3}$$

$$k_{periodic}(\tau = t - t') = \theta_A^2 * exp(-2\frac{sin(\pi|\tau|/\theta_D}{\theta_E^2}) \tag{4}$$

$$k_{adj(x_t, x'_{t'}} = \theta_F^2 * x_t * x1_{t'} \tag{5}$$

The model above indicates that $Y_{s,t}$ is modeled as a logistic regression over latent values $Z_{s,t}$. Latent values from each grid cell, on the other hand, are modeled as a zero-mean Gaussian process equipped with a covariance function that may be expressed as the sum of three components: a local, a periodic and an adjacency component. The first component is expressed as a Matérn covariance function and is used to enforce some smoothness over the time dimension. The second component is expressed as a periodic covariance function, which exploits the periodicity over the time dimension observed within data. Finally, the third component is used to enforce spatial dependencies between neighboring cells, by assuming a linear association between the proportion of neighboring cells experiencing traffic jams and the probability of observing a traffic jam at a given cell. For more information regarding covariance functions, see [**?**], Chapter 4. The covariance functions used by the proposed model require the specification of hyperparameters $= \theta A, \theta B, \theta C, \theta D, \theta E, \theta F$. Here, we obtain them via likelihood maximization using a Laplace approximation for the likelihood function. For a complete description of this process, see [**?**], Chapter 3. Note that, in order to estimate the probability of experiencing a traffic jam at time t, the proposed model requires knowing which neighboring cells are experiencing traffic congestions at the same time t, since we are interested in forecasting traffic jams within one hour. However, we are not allowed access to this information. Therefore, we estimate x_t exploiting the daily periodicity in data, as shown in the expression:

$$\hat{x}_t = \frac{1}{D}\sum_{j=1}^{D} x_t - 24j \tag{6}$$

where D denotes the number of days available for training.

5.2.4. Sentiment analysis

Sentiment analysis deals with the computational detection and extraction of opinions, beliefs and emotions in written text. It combines theories and methodologies from a diverse set of scientific domains, such as psychology, natural language processing and machine learning.

In the context of the *EUBra-BIGSEA* project and smart cities, sentiment analysis is used to transform social media data (textual) into a quantitative estimation of the citizens expressed sentiment. Such analysis may target a specific subject, for example, traffic situation or city services, or a population of a region.

To obtain the data, a Twitter account is required and API access and a Twitter application must be created. In the site `http://apps.twitter.com`, as soon as an application is created, Twitter will generate a set of credentials (keys and access tokens).

The sentiment analysis problem has been addressed through two different but complementary strategies: lexicon-based (using sentiment dictionaries where expressions have sentiment scores) and machine learning techniques.

When using lexicons, a semi-supervised model is built from a list of previously created expressions, targeting, in general, a specific language. It is interesting to note that a particular characteristic of today's online communication may be explored in order to create language independent models: *emojis* and *emoticons*.

Emojis are ideograms used in electronic messages and Web pages. *Emojis* are used like *emoticons* and exist in various types, including facial expressions, common objects, places and types of weather, and animals. An *emoticon* is a pictorial representation of a facial expression using punctuation marks, numbers and letters, usually written to express a person's feelings or mood. Being a smaller set when compared to the entire language vocabulary and language independent, *emojis* and *emoticons* may be used in the sentiment analysis task without requiring great effort. On the other hand, they are subject to misinterpretation in different cultures. There are approximately 1139 *emojis*, disregarding their variations. Most of them are rarely used.

Classification, a machine learning technique, may be used to perform sentiment analysis. Classification is a supervised technique and, as such, it requires a labeled input data set for training and model construction. In order to build the training data, a human must evaluate and label a set of examples. In this case, the human must evaluate whether the text is positive, negative or neutral. In some cases, more than one evaluation is performed for each text. For example, three or more people may evaluate each tweet and after that, a final evaluation emerges.

There are different classification algorithms implemented through different learning methods. For example, it can use random forests, vector machines, gradient boost trees and others. Their performance and accuracy vary in each case and none of them achieves the best results in every scenario. An alternative is to combine different classifiers into ensembles of classifiers. The goal of the ensemble is to integrate algorithms and generate more robust, precise and accurate system results. On the other hand, ensembles require more space, processing time and are less comprehensible. Training data consists of a lexicon list with 118 *emojis* with score varying from -4 (most negative) to +4 (most positive). A manually classified data set is available, formed by texts of 4000 tweets in Portuguese, randomly selected from those collected with geospatial information. These tweets were read by users and classified as positive, neutral or negative. We are not using any special context information or target entity. Thus, for example, a tweet with a positive sentiment for an entity and a negative one for another, could be classified as neutral.

The sentiment analysis for online social data is built as an ensemble of classifiers. We are using both approaches, lexicon with *emojis* and *emoticons*, and machine learning with 3 (three) different algorithms. Even though the use of ensemble of classifiers increases the storage and processing time requirements, this is a good strategy under the project perspective. The different classifiers may be executed in parallel, followed by a final ensemble synchronization step, which is a good test for infrastructure scalability.

6. Experimental results

This section shows some experimental results of the cloud services described in section **??**, covering platform deployment and horizontal and vertical elasticity.

6.1. Platform deployment

Platform deployment is based on Ansible roles and configuration recipes. Details are given in section **??**. The system first deploys the static nodes (a front-end with the master services for

Job	Start		Job	Start		Job	Start		Job	Start
1	19:18:31		6	19:53:51		11	20:21:34		16	20:55:06
2	19:28:02		7	20:10:31		12	20:46:31		17	20:58:08
3	19:28:03		8	20:11:11		13	20:47:11		18	21:23:12
4	19:39:15		9	20:20:28		14	20:47:51		19	21:39:51
5	19:39:45		10	20:20:29		15	20:48:32		20	21:56:31

Table 1: Scheduling of the jobs to be executed.

Mesos, Marathon, Chronos, Wave overlay network and Hadoop namenode), a user-defined number of data nodes (including OpenStack Monasca agents) and a Monasca master node. Then, working nodes are dynamically deployed as frameworks request resources.

New nodes are deployed and configured from scratch. There is no need to build neither pre-existing virtual machine images nor Docker containers. As this process may take several minutes, *EUBra-BIGSEA* provides two alternatives to speed-up the deployment:

- On the fly creation of a reference virtual machine image (golden image) with the first working node, to be used for the next working nodes to be deployed. This reduces the contextualization phase, and it can even take less by tailoring the configuration recipes.
- Stopping the VMs rather than destroying them. This is especially interesting if rapid elasticity is required, although it implies a higher consumption of resources than powering off them.

Scalability is a main issue in the configuration of large infrastructures. Figure **??** shows the deployment time requested for a large-scale cluster with 100 cores and 50 Working Nodes.

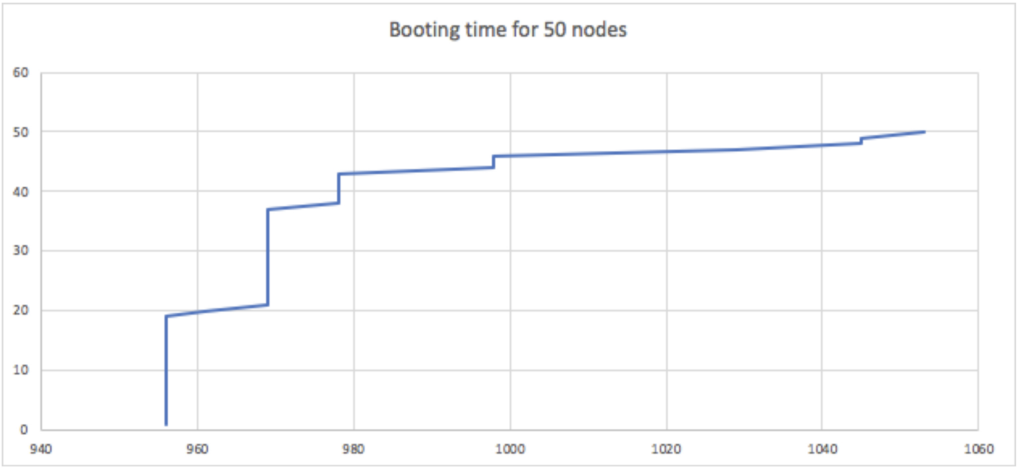


Figure 15: Deployment time for 50 Working Nodes

6.2. Horizontal elasticity

The experiment consisted in submitting 20 parallel jobs to an infrastructure that initially had only two nodes powered on. These jobs were submitted at different time steps as shown in table 10. The infrastructure had to detect the registration of a Spark framework, realize that there are not enough resources and power on one additional node per queued job. Jobs were prepared to run for approximately 11 minutes and were able to use up to 4 cores each. If Mesos offered them 2 cores, jobs will anyway start.

Figure **??** shows the evolution of the status of the WN along time. IDLE indicates powered on nodes with no allocated job. USED means nodes running jobs. POWON means nodes that are being powered on and they have not yet become eligible for running jobs (the contextualization process has not been completed). POWOFF refers to nodes that are being powered off as they have been idle longer than a predefined threshold. Finally, OFF nodes are those that have not been powered on yet. The maximum capacity of this experimental cluster is 20 nodes of 2 vCPUs each.

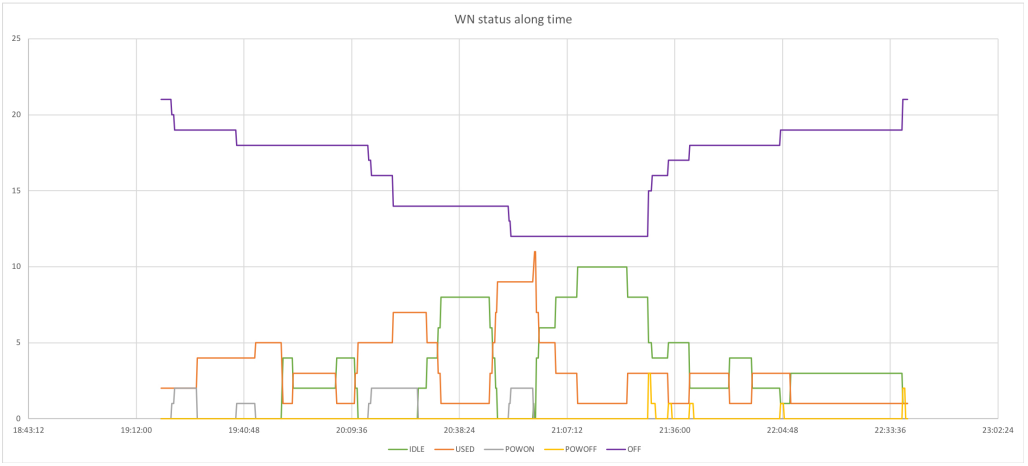


Figure 16: Evolution of the status of the WN over time

6.3. Vertical elasticity

Figure **??** depicts how the CPU adjustment and disk performance limitation can be used to control the performance of applications. By adjusting the performance of both CPU and IO operations, a large range of applications will react with an improvement in performance. The default initial values of 50% capacity are selected and it is dynamically adjusted as the application progresses.

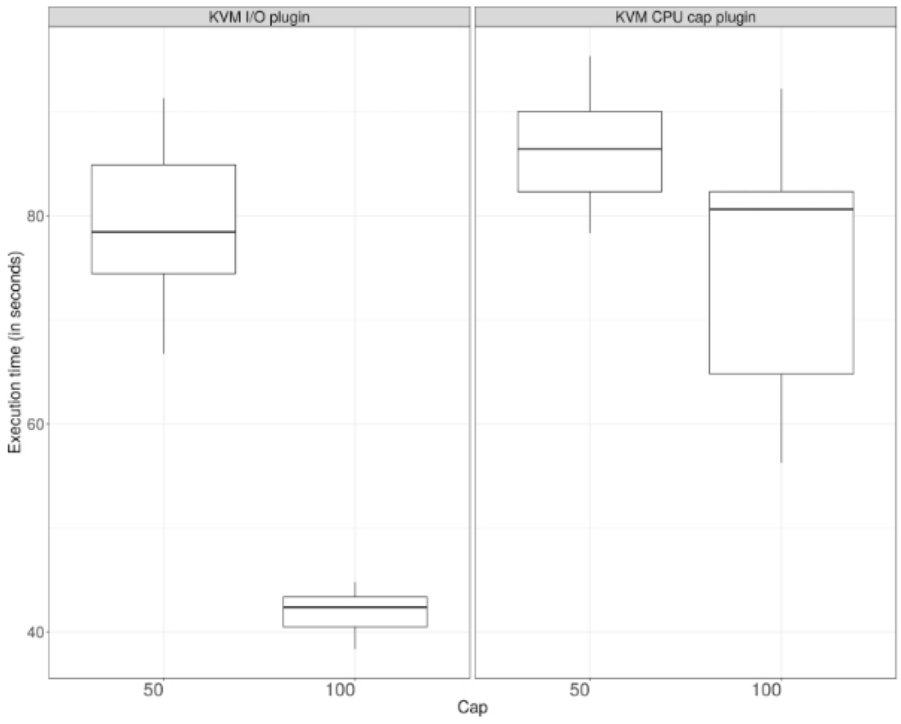


Figure 17: Examples of the usage of CPU and disk IO scaling

7. Conclusions

The partners in the *EUBra-BIGSEA* collaboration have developed cloud-based services mainly focused on data analytics for public transportation data. These services, employing auto-parallelizable

programming models, process the data under restrictions such as Quality of Service constraints and Privacy-awareness.

The partners addressed a major number of software requirements for three use cases on public transportation data management through their solution at different levels of the collaboration.

The first use case, Data Acquisition, dealt with the integration of multiple datasets types and formats (like GTFS from Google and Open Street Maps raw data used to create trips). It also sports support for access control level (in the case of the actual data and metadata) similar to we have implemented inside Lemonade using Thorn. Finally, in the context of the same use case, we have improved the data quality by enriching the existing data like the City Administration Dashboard application, which generates data quality-enriched bus card data.

The next use case, creation end execution of Descriptive Models, included models supposed to extract and characterize trajectories from vehicle movement data. We have also improved the quality of the models by determining correlations and cluster trajectories, as explained for the Entity Matching as a Service (EMaaS) case with its approaches (BULMA, BUSTE, and MATCH-UP). Finally, we have defined the areas of interest in such a way that the models still made sense and were useful but we have reduced the burden of data acquisition and processing. This last case is best evidenced in the Traffic Congestion Estimation and Sentiment Analysis, where we split our data on a grid above the selected cities.

The final selected use case, the creation and execution of Predictive Models, included the training, validation, and building of the models based on multiple sources of data (geographic, social, and meteorological data). For instance, in the case of trip duration and crowdedness predictions, we created a model based on historical bus trips information, passenger boarding information, and points of interest. Due to the continuous modification and evolution of the environment, the models are continuously updated on our infrastructure, taking advantage of predictable execution time from earlier builds and data information. These models would be useless on their own, therefore we expose them to external access. Two demo applications developed in the context of the project, Melhor Busão and Routes for People exploit these models by (for example) proposing a list of three best trips between two stops using predictive models in the supported cities. Finally, this use case also includes the specification of the data sources and regions of interest, like the GTFS data which is considered for a certain number of cities (not the whole planet) from both the EU and Brazil.

In the frame of the *EUBra-BIGSEA* collaboration, we offer a simple interface for data scientist to describe their processing tasks. Our advantages against the competition include functionalities to exploit parallelism, a coherent authentication and authorization model, and tools for data privacy annotation and enhancement, all in one package.

In order to satisfy the use cases and achieve our goals, we have proposed the infrastructure detailed in Figure **??**. We address the need of efficient and convenient development of data analytic applications by allowing application building based on graphical interfaces, using both general purpose and data-analytic specific programming languages. Furthermore, we offer the capability to predict the performance and characterize parallel data analytic applications by leveraging a log analyzer, a performance prediction service, and an optimizer module. Additionally, our infrastructure possesses the ability to scale elastically both horizontally and vertically (cloud resources level dealing with the data analytic applications). Finally, we offer the means to characterize sensitive data using a framework that permits annotations of parts of datasets and also implements privacy enhancement policies.

We managed to address all our use cases, with a result consisting of an infrastructure capable of running services in a scalable way, and that offers a value for the scientist, the normal citizen, and municipality entities.