Alic, AS.; Almeida, J.; Aloisio, G.; Andrade, N.; Antunes, N.; Ardagna, D.; Badía, R.... (07-2). BIGSEA: A Big Data analytics platform for public transportation information. Future Generation Computer Systems. 96:243-269. https://doi.org/10.1016/j.future.2019.02.011

BIGSEA: A Big Data analytics platform for public transportation information

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

[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

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 data sources are available, and different data analytics tools do exist. However, processing such data requires downloading the data,

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

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 (QoS) 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 significant number of software requirements for three main use cases on public transportation data analysis.

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 formats at least.
  - Integrating and dealing with metadata from previous sources.
  - Data quality improvement by cross-correlation of data.
  - Supporting different Access Control Levels for the data and metadata.
- Descriptive Models. One fundamental abstraction of the descriptive models is the computation of trajectories, that is, the path traversed by each public transportation user while using public transportation:
  - Developing models to extract and characterise trajectories from vehicle movement data. Trajectories comprise not only dynamic spatial data but also 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 boundaries of the data to be processed.
- Predictive Models, involving the whole life-cycle and focusing on the trip analysis and trip selection.

[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)

  - Training, validating and building Predictive Models based on geographic (static and dynamic), social, and meteorological data.
  - Recomputing Predictive Models periodically.
  - Estimating the performance of the Predicted Models for different input data.
  - Project Predictive Models as a service.
  - Specifying data sources and regions of interest for any of the operations mentioned above.

1.2. Platform architecture

According to the requirement elicitation and the analysis of state of the art (included in section 7), EUBra-BIGSEA has proposed an infrastructure (see Figure 1) that addresses the following needs:

- Efficient and convenient development of data analysis applications for different access profiles (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).
- Application characterisation and performance prediction under different scenarios (larger or smaller number of resources) in parallel data analytic applications. This is achieved using a log analyser, 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 analysis applications by means of fine-grain monitoring, which triggers the deployment, power-on, contextualisation of computing resources and the dynamic allocation of resources to the jobs executed.
- Being able to characterise sensitive data thanks to policies with a granularity 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. A comparison of the components in EUBra-BIGSEA and other reference implementations in the literature, as well as the description of the innovation, is included in section 7. Summarizing, the innovation is sustained in the following key points:

- Platform described in the infrastructure-as-code approach in a standard language and managed by a platform-agnostic orchestration service.
- An optimisation service to estimate the resource allocation needed for meeting a specific deadline and a proactive service to dynamically tune resources to correct deviations.
- A workflow execution framework which extracts the tasks directly from code dependencies, linked to the infrastructure provisioning services.
- A scalable OLAP system with policy-based privacy preservation techniques.

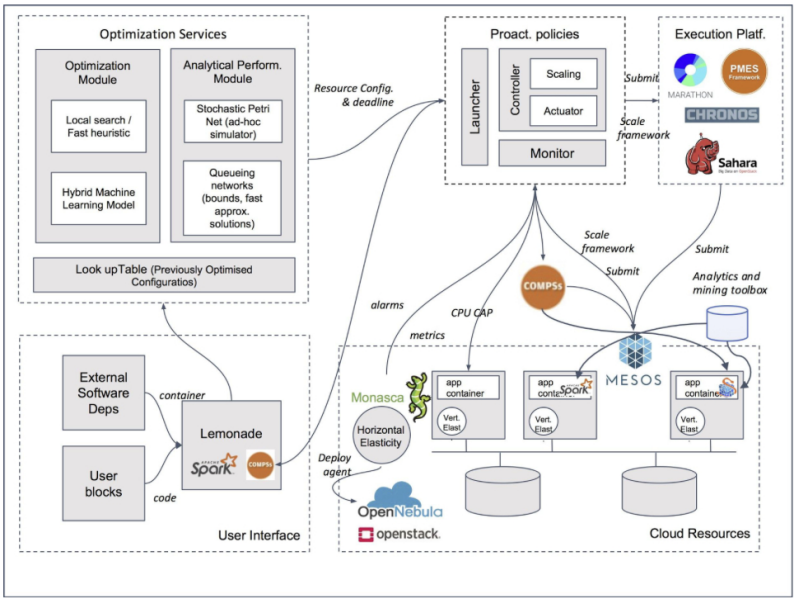


Figure 1: EUBra-BIGSEA software architecture.

- A user-friendly, graphical workflow creation system to build up data analytic pipelines which are translated into Spark or COMPSs code.
- A toolbox with services for different data analytics components, such as Entity Matching, Sentiment Analysis, traffic congestion estimation or crowdedness prediction.

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 EUBra-BIGSEA 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* [1] and *Apache Spark* [2]) 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 a Non-trivial Amount of Data from Everywhere) platform [3], 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 created explicitly 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 democratised the development of new applications, previously restricted to large companies and organisations. 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 performing such tasks requires a complex learning process to use a specific technical solution, the solution will not be effective and useful.

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 analytic functions) to enhance Spark code. Lemonade can process vast 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 microservices, 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 visualisation abstractions. The information stored in Tahiti includes configuration parameters for the algorithms, privacy and security constraints, visualisation and QoS requirements. New operations may 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. *Juicer* controls the actual execution of flows 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 visualisation of results through different metaphors is provided by *Caipirinha*. Finally, the security, privacy and access control is managed by *Thorn*. The interaction of the components is illustrated in Figure 2.

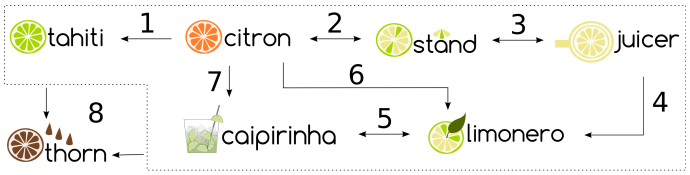


Figure 2: Communication between Lemonade components.

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 datasets that are created as a result of a flow execution (4). Limonero also interacts with Caipirinha to enable data visualisation (5) and back with Citron to make the new datasets available to the user. Visualisations can also be requested by

the user directly from *Citron* (7). Finally, *Thorn* encapsulates all security control, regulating user access, providing security keys, etc. (8).

2.2. COMPSs

COMPSs [1] [4] 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 cloud cluster or containers in Mesos.

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 3, 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.

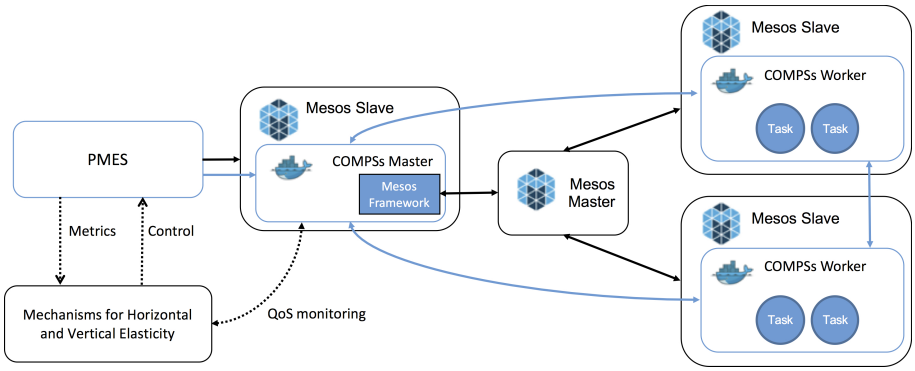


Figure 3: Integration of COMPSs in the EUBra-BIGSEA Platform

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, which takes the decision to profit from this change at runtime, 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 (both at CPU speed or memory 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 [5] [6] [7] 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 [8];
- 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 authorisation, 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 [7]. 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 4 and guarantees a good balance between the scalability of the cluster and deployment simplicity.

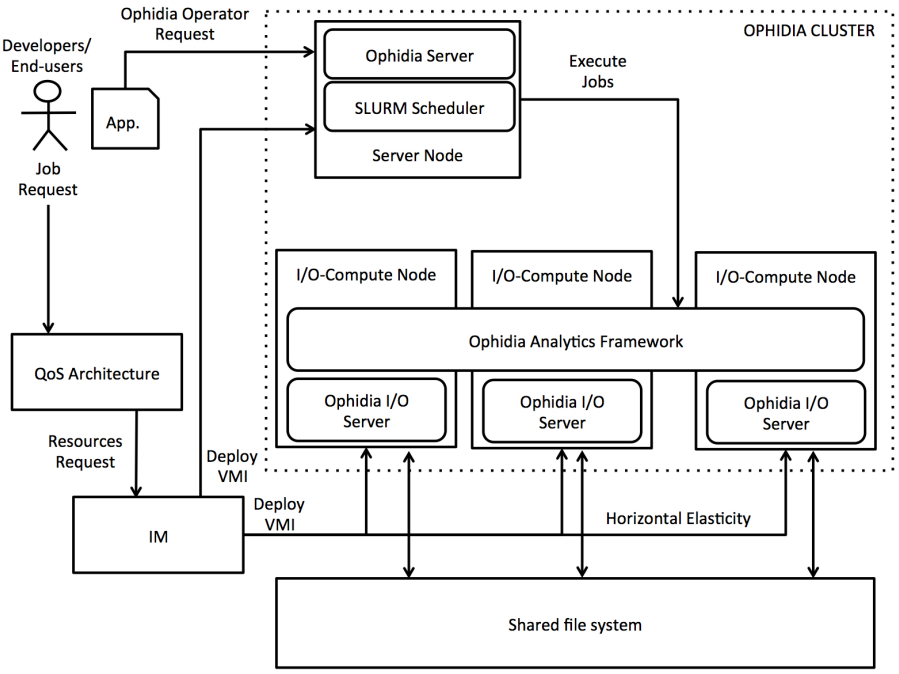


Figure 4: Ophidia deployment schema

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 [9] to automate the dynamic deployment of an Ophidia cluster. Such role is managed by the *EC3* and the *IM* (see section 4.1). The initial deployment of the cluster starts from a request issued to the *pro-active policies* services (i.e., the *broker API*), while the *optimisation service* provides the initial configuration size.

3. Security and Privacy Model

Static and dynamic data analysis 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, Authorisation 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 characterisation 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 characterisation* supports security measurement and includes the assessment and improvement of infrastructure components [10], the benchmarking and improvement of intrusion detection systems, and the proposal of metrics to characterise 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 1, that are most exposed to attacks and faults, namely: *COMPSs*, *OpenStack*, *Docker*, *virtualisation 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 virtualisation 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 on 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 information obtained 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 3.1) and *Privacy-as-a-Service* (presented in Section 3.2) respectively.

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

AAAaaS (Authentication, Authorisation 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.

EUBra-BIGSEA provides the following security functionalities through an Application Programming Interface and the web pages:

- 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.
- Authorisation – 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 5 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 in a DevOps. This web application handles all the HTTPS requests made to the service. Every request is then validated using secure methodologies. For instance, passwords are encrypted with salt functions to safeguard the storage of the passwords. 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.

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 containerised solutions. In the case of the AAA, this includes three main components: web server, web application container and database containers. The architecture of our service (Figure 5) 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

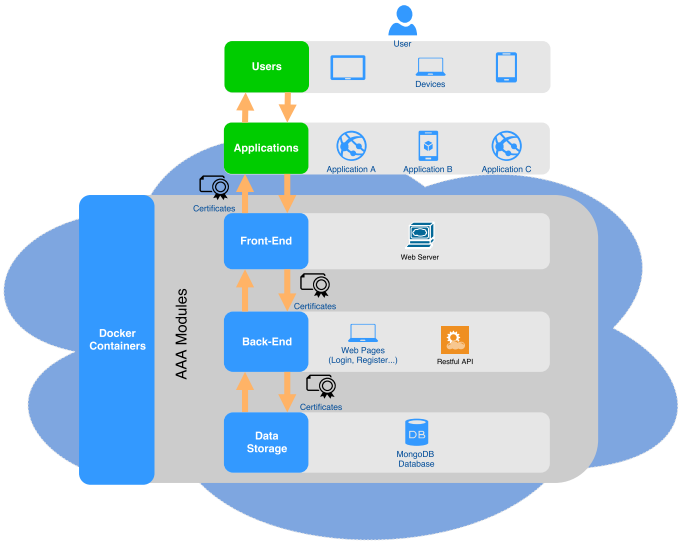


Figure 5: AAAaaS Architecture Overview

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 Authorisation). iAA deals with the authentication and authorisation 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 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 authorisation 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*) [11] 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 analysis algorithms. PRIVAaaS is based on anonymisation policies, i.e. it performs data anonymisation by enforcing the rules specified in the predefined policies. This allows improving, throughout the anonymisation process, the privacy laws compliance as well as the compliance of the privacy requirements provided by the data source owners.

PRIVAaaS provides multiple anonymisation phases and its integration in big data analysis platforms allows performing data anonymisation 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 6 shows the PRIVAaaS general architecture.

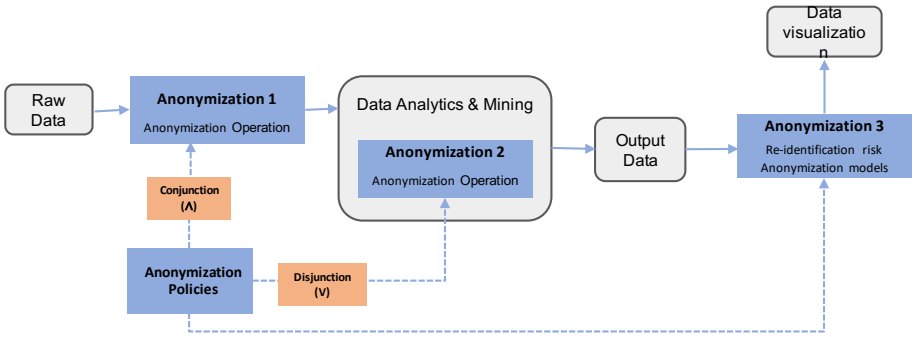


Figure 6: PRIVAaaS general architecture

In Figure 6, *anonymisation policies* surround what type of data fields must be anonymised and how. These policies may be based on privacy principles and laws (e.g., General Data Protection Regulation - GDPR [12]), as well as specifications by data source owners.

Anonymisation 1 is the phase where *raw data* is anonymised. Before anonymisation, a *conjunction* of the anonymisation policies is determined. The conjunction consists in verifying the set of policies field by field and sorting out those requiring anonymisation. 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 anonymisation. This conjunctive process results in the less restrictive anonymisation, maximizing the data utility in this phase.

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

In the *Anonymisation 3* phase, the final *output data* produced by the data analysis 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 anonymisation models (k-anonymity, l-diversity, etc.) before data is available for visualisation.

PRIVAaaS provides a library and a REST service to perform data anonymisation. 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 Anonymisation 1 and 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 dataset to be anonymised and the anonymisation policy (which can be the result of conjunction or disjunction process, for each respective phase). The policy must specify the fields to be anonymised and the anonymisation technique that must be applied to each field. Then the library/service applies the anonymisation techniques according to the policy and provides, as output, the file with the anonymised data set. The techniques that have been implemented in PRIVAaaS are *generalisation* (attributes are replaced by some more generic ones, that are faithful to the original); *suppression* (attributes are completely removed to form the anonymised dataset); *encryption* (cryptographic schemes are applied to replace attributes with encrypted data) and *perturbation/masking* (attributes are replaced by dummy data).

The Anonymisation 3 phase implementation also receives as input a data set to be anonymised (resulting from data mining and analytics, which would be released from the platform) and the anonymisation 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 [13]. 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 [14], initially set as $k = 2$, is increased and k-anonymity is applied with this new value of k. This is performed iteratively, 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 [15] recipes as Ansible roles and using the Infrastructure Manager [16] to interact with the cloud IaaS. The Infrastructure Manager is a TOSCA [17]-compliant platform-agnostic cloud orchestrator. By means of the Elastic Compute Clusters in the Cloud (EC3) tool [18], a complete cluster can be set up with minimal intervention. The client and server applications, as well as 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) [19]. CLUES powers on and off resources as requested by an agent that polls the resource manager queues (Mesos[4]

[2] Docker Hub reference

[3] https://github.com/eubr-bigsea/ec3client

[4] http://mesos.apache.org

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 [20]. Data is stored in an HDFS distributed storage that can grow horizontally. Masks jobs are connected by means of a Wave overlay network. 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 optimise resources and fit the deadline.

4.2. Performance Prediction and Optimisation 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 optimisation 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 or Spark), 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 [21] 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 [22], 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 Optimisation 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 characterise 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 Optimisation 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 optimisation 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 optimisation 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 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 authorisation levels. It also triggers the start of the application, as well as monitoring the execution and reacting to delays in the execution. The Broker is configured with the execution plug-in, which indicates the type of Big Data framework (e.g. Spark or COMPSs) and the underlying infrastructure (OpenStack and OpenNebula).

Next, the *Monitor* component is responsible for mapping application-specific progress to a normalised 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 standardised form in the cloud monitoring system (Monasca [23]). 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.

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 customised 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 impact 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.

For example, the Brazilian traffic enforcement legislation [24] [25] 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. Similar regulations [26] can be found in other countries as well [27, 28].

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.

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 the area, but in general):

1. Building a trustable database of injuries;
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.

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 7 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.

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

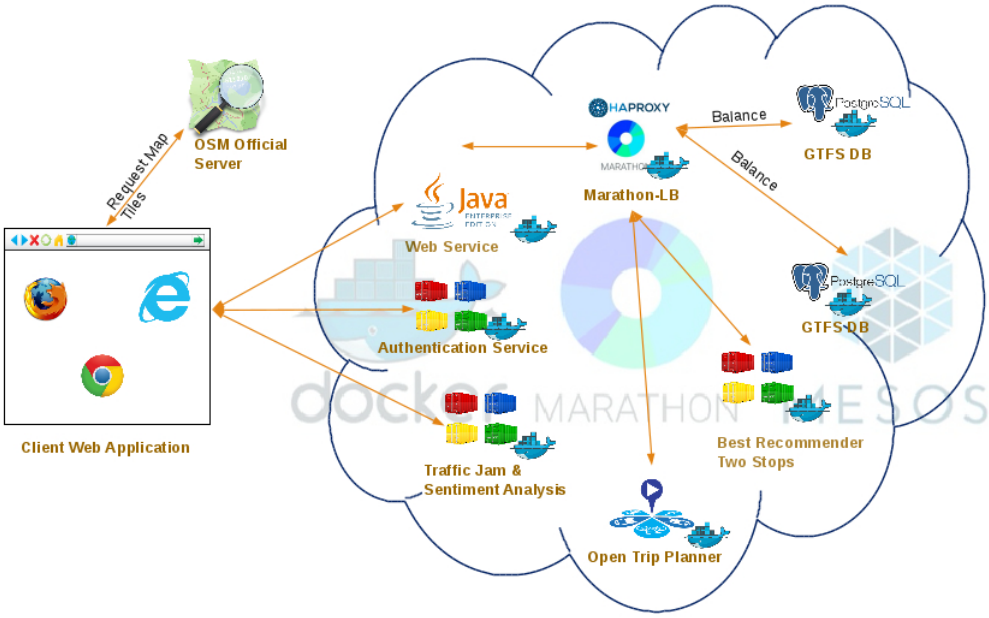


Figure 7: General architecture of the Routes for People Web and dependencies

- 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 the rest of the services.

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.

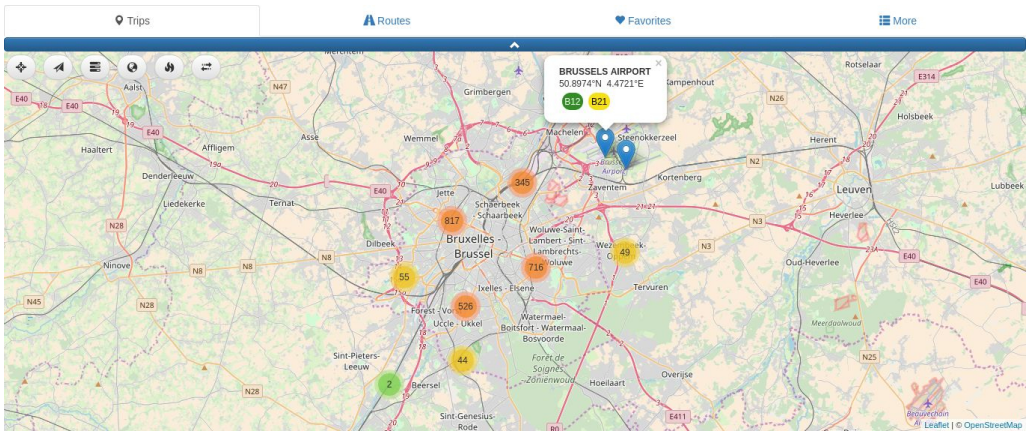


Figure 8: The first tab of the Routes for People Web application

The web application contains four tabs (Figure 8). 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.

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 9 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).

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. First, the anonymisation 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 an HDFS cluster, 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* [29] 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

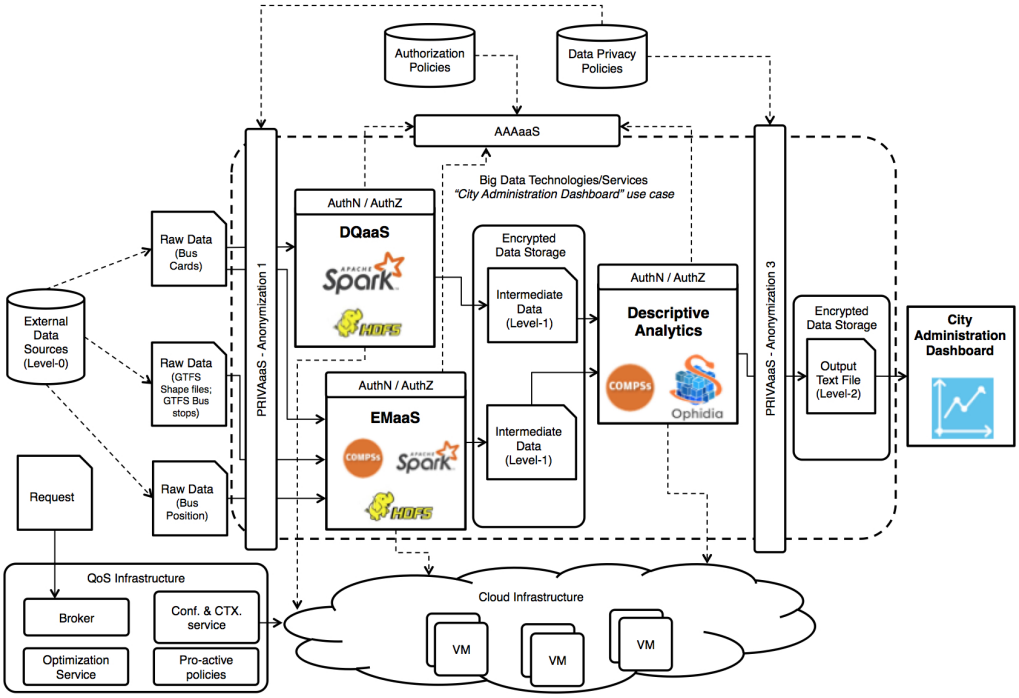


Figure 9: City Administration Dashboard - system architecture overview

to which all values are registered in the dataset), and *consistency* (highlights if the data dependencies are satisfied). The *EMaaS* (see 5.2.1) 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 anonymisation 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 visualisation 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

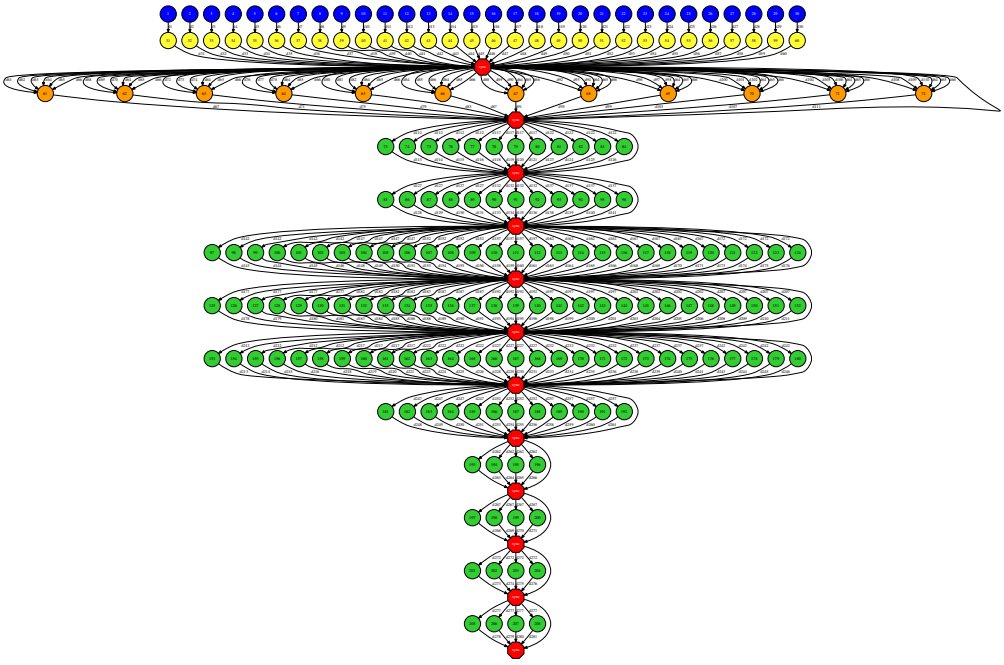


Figure 10: COMPSs simplified execution graph example (set of statistics computed by the application). Each circle represents a task executed on the remote node. Different colours denote different task types. Tasks in the same horizontal level can be executed concurrently. Arrows denote dependencies among tasks. A description is given in the text.

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 anonymisation 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 authorisation, all the data components and services used by this application (i.e. Ophidia, EMaaS and DQaaS) verify the validity of the token and the authorisation 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 output of the Optimisation services.

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 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 parallelises applications exploiting their inherent parallelism without the burden of parallel code implementation, while Ophidia provides support for a wide set of efficient parallel data analysis operations. Their integration into the QoS framework allows addressing QoS-based scenarios.

Figure 10 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 anonymisation 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

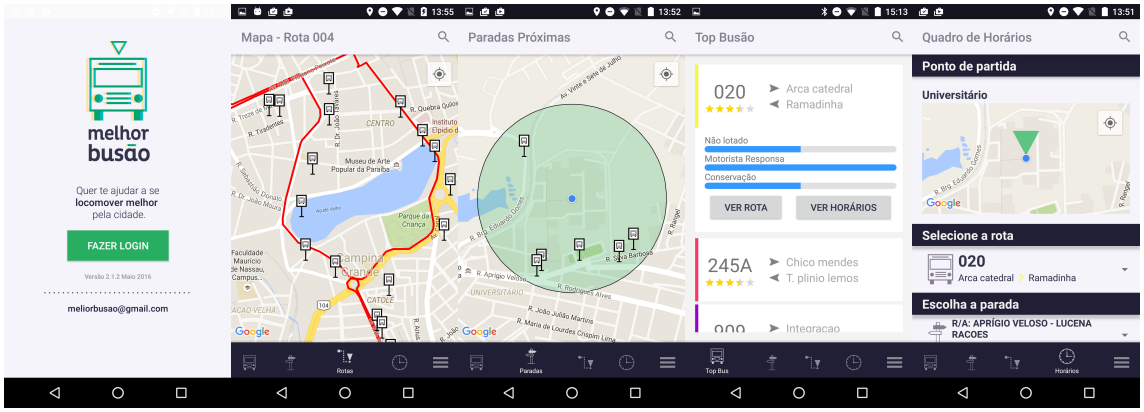
Melhor Busão (a Portuguese expression for Best Bus) is an application designed to present the results of the Transportation Data Processing and Analysis performed by the EUBra-BIGSEA infrastructure to transit users. It is implemented as a mobile application so as to simplify access and notifications 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 [30] [31].

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 analysis services of EUBra-BIGSEA. The application sends a request with user trip plan information to a Web Service named Best Trip Recommender (BTR). 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 visualise them. BTR 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 11 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 11: 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 to provide security to users transactions. Melhor Busão uses 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. Finally, Melhor Busão runs on top of the cloud infrastructure back-end services.

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 [32]. 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 [33]. 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 data quality 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 [34].p

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 utilised 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 2) 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 anonymised ticketing data as input and produces enriched historical bus GPS data. This means that the *BUSTE* computation is not influenced by the presence of anonymised 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 12. 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.

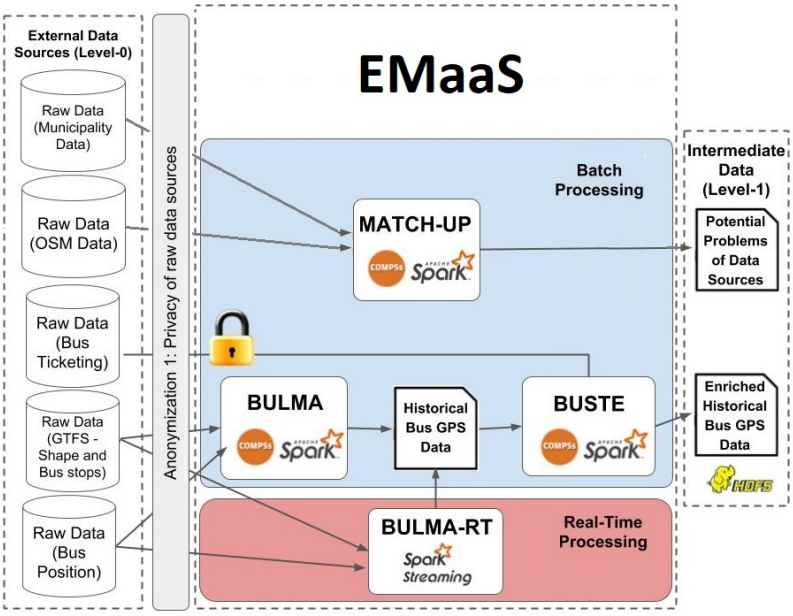


Figure 12: EMaaS Architecture

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[5] and Here[6], 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) realtime vehicle location and (iii) passenger boarding data. The first one is available in GTFS format[7] 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 5.2.1. 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.

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,

[5]https://maps.google.com
[6]https://wego.here.com/
[7]https://developers.google.com/transit/gtfs/reference/

we first discretise 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 discretised 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.

We model this binary variable through a regression that outputs a class probability using a response function, in our case a logistic regression, that "squashes" its argument into the range $[0, 1]$, allowing its interpretation as a probability. In practice, with each variable $Y_{s,t}$, we associate a latent (unobserved) variable $Z_{s,t}$ that models $Y_{s,t}$ as a logistic regression:

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

Letting x_t denote the proportion of neighboring cells experiencing some traffic jam at time t, the latent values from each grid cell $Z_{s,*}$, on the other hand, are modeled as a zero-mean Gaussian process ($\mathcal{GP}$) equipped with a covariance function (between input pairs t and t') that may be expressed as the sum of three components: a local (k_{local}), a periodic ($k_{periodic}$) and an adjacency (k_{adj}) component:

$$Z_{s,*} \approx \mathcal{GP}(0, k_{local}(t, t') + k_{periodic}(t, t') + k_{adj}(x_t, x'_{t'})) \tag{2}$$

The first component ($k_{local}(t, t')$) is expressed as a Matérn covariance function and is used to enforce some smoothness over the time dimension:

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

The second component is expressed as a periodic covariance function ($k_{periodic}(t, t')$), which exploits the periodicity over the time dimension observed within data:

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

The third component ($k_{adj}(x_t, x'_{t'})$) is used to enforce spatial dependencies between neighboring cells, by assuming a linear association between the proportion of neighboring cells experiencing traffic jams ($x_{t'}$ and the probability of observing a traffic jam at a given cell x_t:

$$k_{adj(x_t, x_{t'})} = \theta_F^2 \cdot x_t \cdot x_{t'} \tag{5}$$

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 maximisation using a Laplace approximation for each likelihood function θ_i. For more information regarding covariance functions, see [35], chapter 4, and, for a complete description of this process, see [35], 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 jams at the same time t, since we are interested in forecasting them within

one hour intervals. Therefore, our estimate $\hat{x}_t$ of x_t exploits the daily periodicity in data, as shown by 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 and $\hat{x}_t$ is assumed to be the average of x_t for each hour of the day t considering all 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 synchronisation 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 4, 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 4.1. The system first deploys the static nodes (a front-end with the master services for 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 contextualisation 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 13 shows the deployment time requested for a large-scale cluster with 100 cores and 50 Working Nodes.

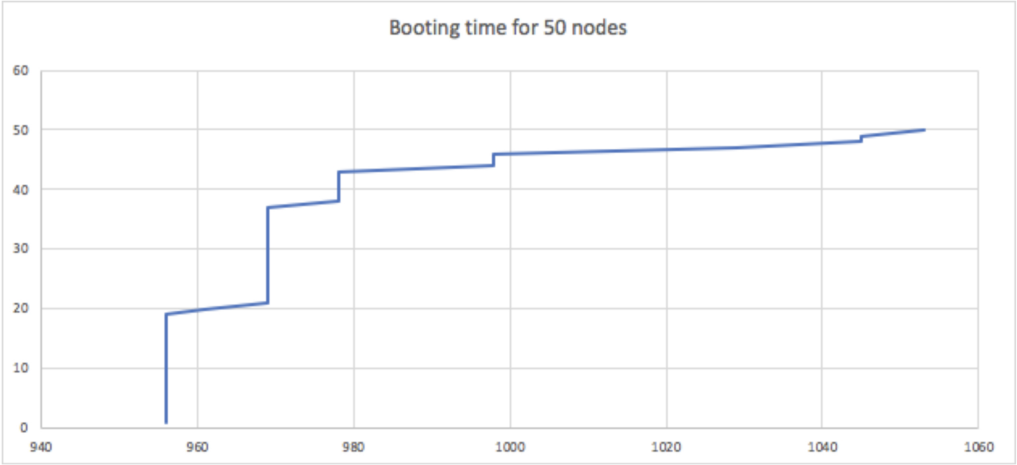


Figure 13: Deployment time for 50 Working Nodes. The vertical axis indicates the number of VMs ready and the horizontal axis the time in seconds.

6.2. Performance Prediction and Optimisation Services

The Performance Prediction and Optimisation Services (illustrated in section 4.2) have been validated considering the BULMA application (see section 5.2). As accuracy metric, assuming to optimise the BULMA initial deployment with a deadline D to meet, we focus on the execution time T measured in the cluster provisioned according to the number of VMs determined by the optimisation procedure and quantified the relative gap wrt. the deadline. Formally:

$$\%Error = \frac{D - T}{T},$$

note that, possible misses would yield a negative result.

Overall, we considered 56 cases, varying the deadline between 5,500 sec and 11,000 sec with step 100 sec, and ran the optimisation service to determine the optimal resource assignment to meet the deadline constraint on Microsoft Azure D4 v2 instances. Figure 14 plots the results we achieved.

We experienced a deadline miss only in 5 out of 56 cases (8.9%) of cases. Moreover, the relative error is always below 35%, with a worst case result of 35.12% and the average absolute error settling at 18.13%. From the plot, we observe some abnormal behaviours (rapid jumps) which are related to the change of the configuration deployment (number of required VMs) suggested by the optimisation service. In particular, the gap in terms of number of VMs required to fulfill the deadline D (evaluated a posteriori) is at most one VM and the accuracy is higher when the deadline is tight, i.e., for larger clusters. Since the initial deployment can also be updated by the pro-active policies module (described in section 4.3), overall we can conclude that the optimisation service is effective in identifying the minimum cost initial deployment, guaranteeing that deadlines are met as well.

6.3. 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 1. The infrastructure had to detect the registration of a Spark framework, to realise that there are not enough resources and power on one

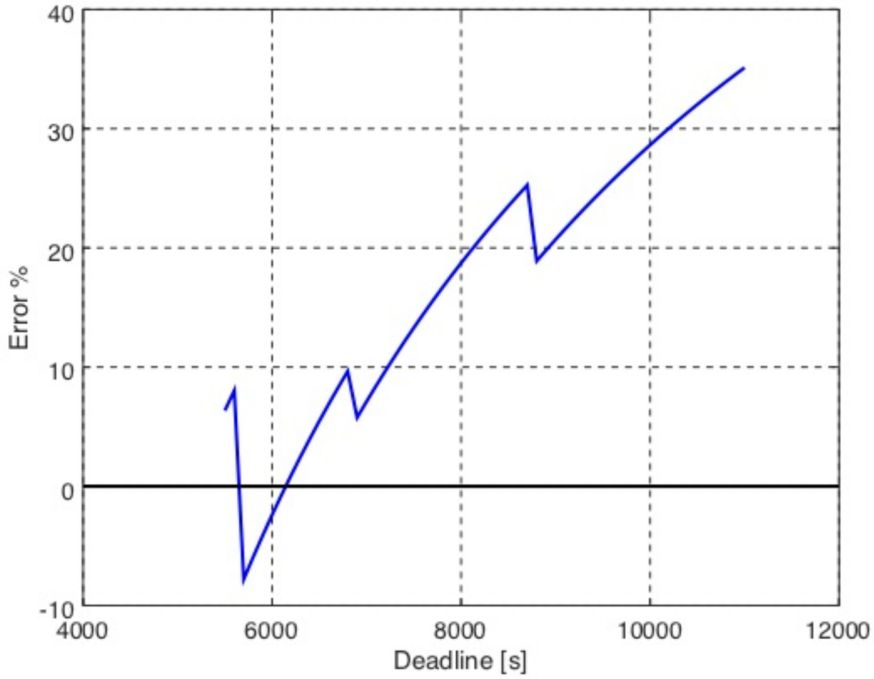


Figure 14: Optimisation Service validation on BULMA application

Job	Start		Job	Start		Job	Start		Job	Start
1	0:0:0		6	0:35:20		11	1:3:3		16	1:36:35
2	0:9:31		7	0:52:0		12	1:28:0		17	1:39:37
3	0:9:32		8	0:52:40		13	1:28:40		18	2:4:41
4	0:20:44		9	1:1:57		14	1:29:20		19	2:21:20
5	0:21:14		10	1:1:58		15	1:30:1		20	2:38:0

Table 1: Scheduling of the jobs to be executed.

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 15 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 contextualisation 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.

6.4. Vertical elasticity

Figure 16 depicts how the CPU adjustment and disk performance limitation impacts in the performance of applications. By adjusting the performance of both CPU (by controlling the mapping between virtual and physical CPUs) 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

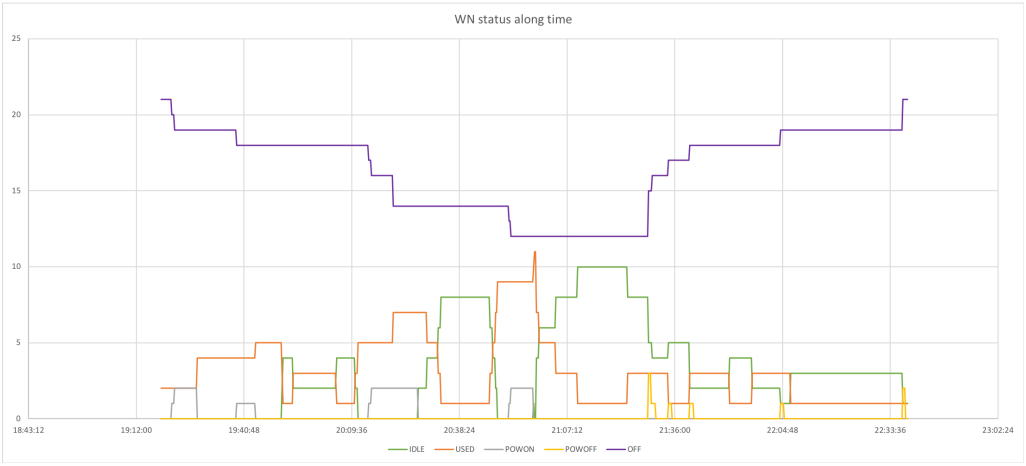


Figure 15: Evolution of the status of the WN over time

as the application progresses. Choosing such a value also enables a homogeneous performance when the cloud combines machines with different peak performance.

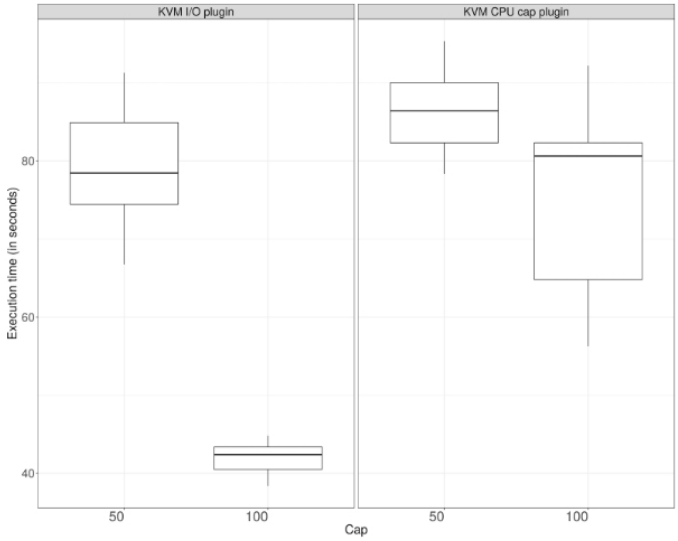
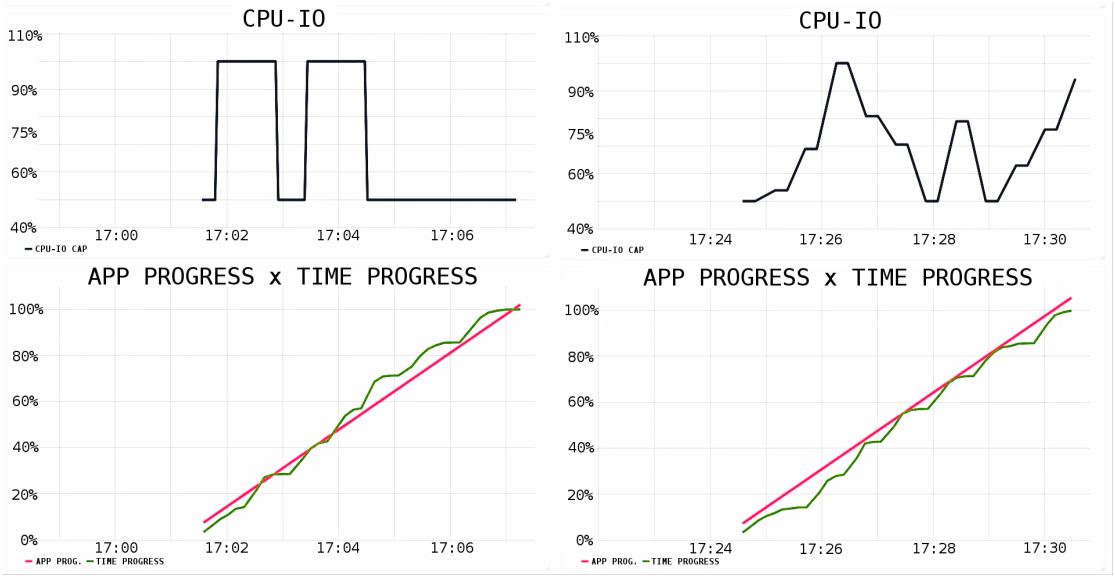


Figure 16: Examples of the usage of CPU and disk IO scaling

In order to have a closer look on the impact of controllers and actuation, and validate the proactive vertical scaling based on progress metrics, we performed experiments considering the EMaaS running in a OpenStack infrastructure with a strict deadline that was not achievable on a static infrastructure. Thus, in this experiment we consider the progress metric from the EMaaS and for the controller component, we consider a hysteresis control and the KVM-over-OpenStack actuator plug-in, which adjusts the CPU-IO cap (a combination of individual adjustments of CPU and IO cap compared above). Figure 17 shows the CPU-IO cap configured (first row) and progress of the application and time (second row) over time for two scenarios of controller configuration. The MIN-MAX approach adjusts the CPU-IO cap only with two values, a minimum and maximum level (50% and 100% for this case). The second scenario uses a proportional-derivative approach, that calculates



(a) MIN-MAX (b) PROPORTIONAL DERIVATIVE

Figure 17: Two executions of the EMaaS service with different controllers.

the quantity of resources to be adjusted based on the progress error and the error variation, provided by the last adjustment. As can be seen, the MIN-MAX approach (Figure 17a) enabled the EMaaS to execute with an acceptable runtime, meeting the deadline, with the second half of the time progress curve above the expected. In comparison, the proportional derivative (Figure 17b) ran the application always under the expected progress, leading to a small delay regarding the programmed deadline. Note that although the derivative controller missed the deadline, it also actuated much more smoothly on the infrastructure, which is also a desirable behavior.

6.4.1. PRIVAaaS Experimental Results

A case study with PRIVAaaS was performed in the context of the *City Administration Dashboard* use case (see Section 5.1.2). The anonymisation policy specifies that the bus card identifier - the unique ID number identifying the bus card user - must be anonymised (*Anonymisation 1 phase*). Then, the anonymised version of the input data undergoes some ETL steps in the Ophidia platform, which provides the resulting output from data analytics. As in this first stage less restrictive anonymisation was applied to provide better utility for analytic process, the resulted output may contain quasi-identifiers fields (birth date and gender) and additional anonymisation is required using the PRIVAaaS re-identification risk component (*Anonymisation 3 phase*).

In this case study the k-anonimity was applied to achieve risk thresholds between 1% to 5%. Figure 18 presents the information loss throughout the Anonymisation 3 phase. The outputted dataset starts without data loss and the re-identification risk is 100%. For the risk threshold we adopted for these experiments (1% to 5%), the k-anonymity reaches k=2 and the data loss increases to approximately 25.5%. Just for illustration, we represented in Figure 18 smaller thresholds (0.5% and 0.1%). This last threshold demands a higher value of k (in this case, k-anonymity reaches k=301) but implies a data loss of 100%. This k value increased considerably the anonymity

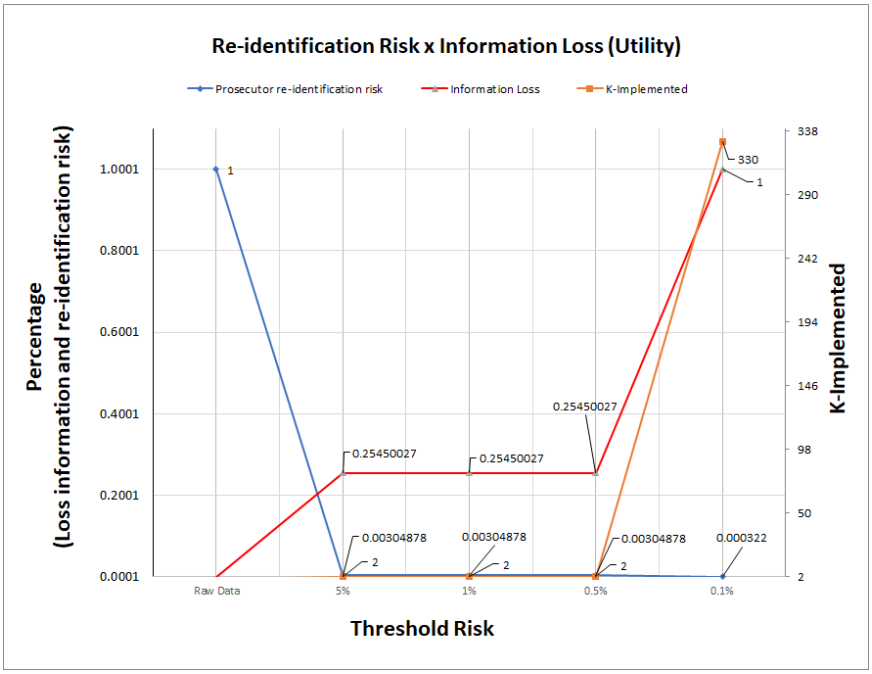


Figure 18: Re-identification Risk X Data Loss in the second stage.

level and the probability of re-identification goes from 0.003048 to 0.000322 (i.e. it improved 946%). It is important to note that for this dataset from 5% down to 0.5% of risk threshold there is no possible data anonymisation that renders the k-anonymity algorithm to reduce the re-identification risk and it remains the same value (0.00304878).

6.4.2. Entity Matching Experimental Results

In this section, we evaluate the *BULMA* technique against the *BoR-tech* (a state-of-the-art approach [36]). We ran our experiments on a commodity server machine that has one Intel I7 processor with four cores, 16GB of RAM and 1TB of hard disk. Among the softwares installed at the machine, Ubuntu 64-bit and Java 1.8 were utilised. Two data sources are used: DS-GPS contains one day of GPS information (2016-10-30) from the city of Curitiba (Brazil), and DS-shapes contains the GTFS specification from the same city. As a gold standard for the evaluation, ground-truth data was manually (visually) labeled by a human data specialist with all the trips of nineteen routes performed by the buses of Curitiba at 2016-10-30. It contains 215 trips performed by 15 buses. These trips comprise 147 trips by 9 buses on complementary shapes and 68 trips by 6 buses on circular shapes. To the map-matching effectiveness measure the effectiveness, we applied the three traditional quality metrics: i) recall; ii) precision; iii) F-Measure (FM).

Regarding the map-matching effectiveness of the two techniques, Table 2 show the map-matching effectiveness values of both techniques. The evaluation is organised in two scenarios. The first one is related to the classification performed by the techniques regarding the two types of shapes (complementary shapes (i.e., shapes that must join other shapes to form a complete route) and circular shapes (i.e., shapes that describe a complete route by itself). The second one is related to the classification performed by the techniques regarding the problematic routes existing in the DS-GPS dataset.

Table 2: Comparative table of common cases

Type of trajectory	EXECUTION TIME (s) BULMA	EXECUTION TIME (s) BoR-tech	F-MEASURE BULMA	F-MEASURE BoR-tech
Complementary	21	**13**	**0.98**	0.94
Circular	19	**13**	**0.87**	0.26
Overall	25	**17**	**0.94**	0.70

Thus, Table 2 shows the observed execution times along with the number of buses and F-measure collected values for the execution of the two techniques over common routes existing in the DS-GPS. As we can see, in all cases, none of the approaches achieve the (highest) F-measure of 1.0. The main reason for this result is related to the significant amount of sparse and missing GPS data within a few bus trips. Such situation leads the techniques to make erroneous decisions during the detection of start and finish points. In particular, there is a higher gain in effectiveness when using *BULMA* for map-matching circular shapes. *BULMA* achieved an F-measure of 0.87 against 0.26 of *BoR-tech* for the circular routes. This result shows the lack of robustness of techniques that do not account for detecting the correct shape among multiple shapes that refer to the same route.

Table 2 also shows that *BoR-tech* has shorter execution times than *BULMA*, as it has employs no computation to treat the problem of multiple trajectories in a same route. On the other hand, *BULMA* provides higher map-matching effectiveness than *BoR-tech* in all cases. Since *BULMA* selects the best sequence of shapes associated with the entire trajectory performed by a bus during a day, it is able to optimise the "best fit" sequence of shapes according to the trajectory performed by the bus. More details about BULMA approach can be seen in the work [37].

Regarding the scalability evaluation, we analyse the scalability gain of the two Spark-based approaches. We evaluate *S-BULMA* and Spark-based *BoR-tech* aim to investigate how they scale with the increasing number of available executors (nodes) n. To achieve this, we ran our experiments on a cluster composed of eight virtual machines. Each one has four cores, 8GB of RAM and 500GB of hard disk. Among the product of software installed on the machine, Ubuntu 64-bit and JAVA 1.8 were utilised. For each round of the techniques execution, a new instance of JVM was created to disable information reuse (in memory). We utilised DS-GPS2 and DS-shapes for this evaluation.

As shown in Figure 19, we can note that the approaches scales almost linearly up to five executors showing its ability to evenly distribute the workload across the workers. The load balancing was properly treated when the blocking key based on the combination of the route identifier and bus code was utilized as a partition splitter. The usage of the route identifier promotes a significant reduction in the number of shapes to be compared (at most six shapes). Such strategy increases the granularity of parallel tasks favoring thus the load balancing. However, with about seven workers, the parallelism was compromised due to the presence of a much larger amount of executors needed to process all workload generated. Note that the execution time stabilises when more than six workers were utilised. Also note that the execution time (of *S-BULMA*) decreases from almost 160 min (for one

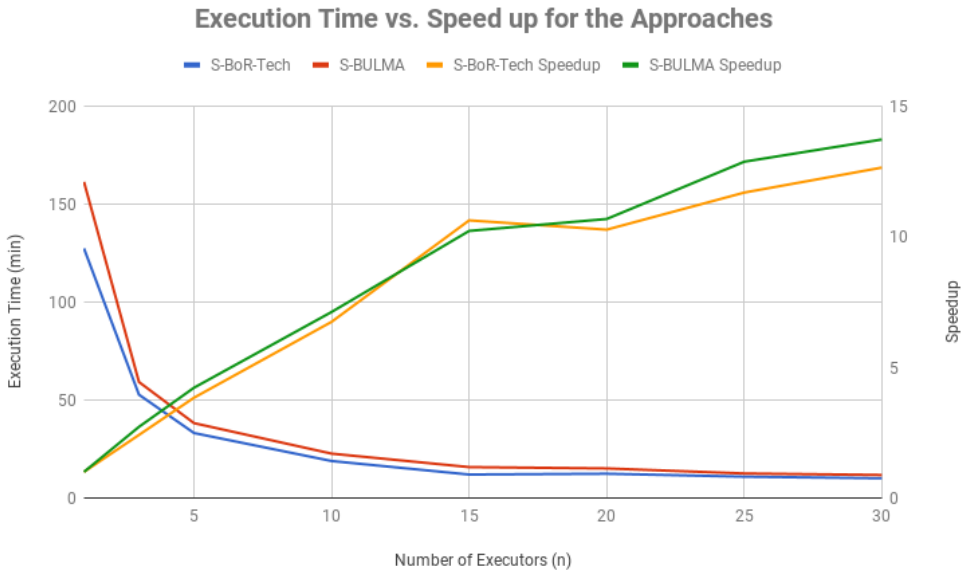


Figure 19: Execution time and speedup for the approaches.

worker) to about 18 min (for 15 workers).

7. Discussion

In this section we perform a review of the state of the art in different technologies that have been developed in EUBra-BIGSEA and described in the paper.

7.1. Deployment and Operation of Cloud Services

The deployment and configuration of scientific applications in cloud resources [38] has been tackled before by several authors, both for specific disciplines such as physics [39] [40], astrophysics [41], biomedicine [42] [43] and for general purpose [44] [45] [46]. Several open source orchestration tools and services exist in the market, but most of them come with the limitation of only supporting their own Cloud Management Platforms (CMPs) as they are developed within those project ecosystems. (e.g. OpenStack Heat and HOT [47], OneFlow [48]). Efforts on supporting platform-agnostic orchestration tools are Cloudify [49], which provides TOSCA-based orchestration across different Clouds, OpenTOSCA [50] and Apache ARIA [51], although any of them currently support OpenNebula. Other projects add the compatibility to multiple providers by supporting the Open Cloud Computing Interface (OCCI) [52], such as CompatibleOne [53], although the project has not been active in the last years. Infrastructure Manager (IM) [54] supports TOSCA-based deployments over a variety of cloud backends including OpenNebula and OpenStack, the most important commercial cloud providers, and the EGI Federated Cloud [55], a large-scale and pan-european federated IaaS Cloud to support scientific research.

The solution for Deployment and Operation of Cloud Services described in this article supports a wide range of public and on-premise cloud IaaS backends (including the OCCI standard) and uses standard descriptions of the application topologies in TOSCA and Ansible. IM also supports different contextualization modes depending on the number of resources to be simultaneously managed.

7.2. Big Data Analytics

Nowadays, several tools and frameworks are available to address the various challenges related to Big Data analysis. Some well-known Big Data frameworks include Apache Hadoop [56] and Apache Spark [2]. With respect to these general-purpose solutions, Ophidia is more oriented towards multi-dimensional array-based data analytics, joining together high performance database management and OLAP aspects. Additionally, being more tailored to scientific applications, Ophidia supports scientific formats, metadata management and numerical libraries.

On the other hand, as for eScience data management solutions, SciDB [57] represents an array-oriented distributed DBMS for scientific applications, while Rasdaman [58] is an array database suitable for storing and querying scientific multi-dimensional array-oriented data. These systems are similar to Ophidia since they both rely on n-dimensional arrays; yet, Ophidia (i) implements a datacube abstraction (with related algebra and OLAP operators) thus concealing the low-level array layer, and (ii) it provides a framework for the execution of parallel (MPI-based) operators with native support for in-memory, parallel (OpenMP-based) data analytics. Ophidia is a highly scalable and efficient data processing environment tailored for multilayer datacubes, such as Geographic Information data, where Ophidia's performance is very high. Through the extension described in this article, it provides self-managed elasticity to scale up and down processing nodes on top of the distributed parallel data management system. It also provides an API and a CLI to easily manage the resources allocated.

We consider work related to the integration of application frameworks or workflow management systems with container engines. Skyport [59] is an extension to an existing framework to support containers for the execution of scientific workflows. This framework differs from our proposal both in the programming model, because it requires to explicitly write the workflows' tasks in the form of JSON documents and in the integration with containers technologies; in COMPSs, besides Docker extensions to Singularity, and Mesos are also provided. In Makeflow [60] workflows are represented by chained calls to executables in the form of Makefiles and is tailored to bioinformatics applications; it does not provide any tool to build container images from the workflow code and the supported elasticity is done per task thus creating a new container and not reusing existing containers for different tasks. Similarly to Makeflow, Nextflow [61] proposes a DSL language that extends the Unix pipes model for the composition of workflows. Both Docker and Singularity engines are supported in Nextflow, but it has similar limitations to other frameworks, such as the manual image creation, a limited programming model that resembles command line executions of scripts and a limited elasticity provided only for Amazon EC2. COMPSs programming model is also infrastructure-agnostic, so the independent tasks can be executed on different types of resources, including batch queues and GPGPUs. This provides the application developers a high degree of flexibility.

7.3. Monitoring and Vertical Elasticity

Monitoring applications and infrastructures is an essential step in other to control the behavior of systems and applications. In traditional cloud-based systems, monitoring is typically used to control infrastructure and applications, making provisioning decisions (e.g., autoscaling), balancing load, monitoring health status, and taking other actions related to quality of service and availability.

In EUBra-BIGSEA, monitoring goes far beyond typical metrics such as CPU and network usage. The proactive policy component offer services that facilitate the usage of any metric or of the consumption of any observable resource to control a running system. Having adaptable monitoring and actuation infrastructure is not a trivial task. Guidelines to define and formulate performance metrics across services with different purposes and types of resources to give a holistic view of a complex system remains an open issue [62]. Thus, the traditional requirements from a monitoring system, including performance monitoring, custom monitoring metrics, infrastructure monitoring, networking monitoring, and end-to-end monitoring, are just a starting point. The proactive monitoring, combined with the actuators ends up with an innovative solution to implement Quality of Service in terms of allocated CPU to meet execution deadlines. The combination of the actuation at hypervisor and container model are versatile and leverage the load rebalacing services that the infrastructure solutions provide.

When considering complex Big Data systems, such as the ones expected to be deployed in a EUBra-BIGSEA platform, desirable features are expanded to include the following [63]: (1) *autodiscovery*, capability to discover and identify new services automatically, adding the respective monitoring agents on the fly; (2) *system-wide health management*, detecting problems out of the scope of an individual component, such as system-wide latencies; (3) *artificial intelligence and predictive monitoring*, machine learning approaches are needed to separate minor performance glitches (e.g., due to irregularities in data processing) from an actual issues (e.g., workload incompatible with allocated resources), and for early detection of upcoming resource bottlenecks; finally, (4) *support for heterogeneity*, supporting different clouds platforms and different kinds of target goals (e.g., request response time regulation, deadline enforcement for batch jobs, throughput regulation), different kinds of input metrics (e.g., CPU usage, database bandwidth, result quality or accuracy in case of tunable algorithms), and different kinds of actuators (e.g., control the number of VMs/containers in a layer, control the contracted limiting rate of a remote serviced API, control the quality for tunable algorithms).

At the same time that EUBra-BIGSEA supports some of the features by default, such as the capability to incorporate heterogeneous platforms, metrics and controls, and the ability to integrate these with machine learning techniques for resource estimation and proactive adjustments, it also creates hooks that enable even more sophisticated solutions to be easily plugged in. For example, a sophisticated predictive control algorithm can be plugged into a deployed system, without much dependency on how the metrics are collected or how the actuation on a specific platform would be.

Although there are many popular tools for monitoring, most of them target simple, traditional infrastructures. For example, a popular monitoring system is Zabbix [64], which offers a rich set of agents for collecting metrics targeting metrics from networking, servers, clouds, and even generic KPIs and SLAs. Zabbix also includes the ability for building a diverse set of dashboards. Similarly, the combination of Sensu [65], InfluxDB [66] and ELK [67] (which stands for ElasticSearch, LogStash and Kibana) enable the collection of a diverse number of infrastructure metrics (Sensu), the storage of this metrics in a time-series database (InfluxDB), and the collection, aggregation and indexing of logs (through ELK). This combination of tools leads to a comprehensive, but complex system. Even though these systems are

large deployed in production, both Zabbix and the Sensu-Influx-ELK combination lack features for fine-grained control over the metrics dissemination and also features based on artificial intelligence to mitigate the problem of metric overload.

One of the monitoring systems supported in the EUBra-BIGSEA ecosystem is Monasca [23], which started as the Monitoring-as-a-Service component for OpenStack (the major open-source cloud management platform) and evolved to be used in other cloud-driven platforms such as Kubernetes [68]. Monasca is an open-source multi-tenant service that aims to be highly scalable, performant, fault-tolerant service. It uses a REST API for high-speed metrics processing and querying, and has a streaming alarm engine and a notification engine. Monasca also aims to provide artificial intelligence components such as the anomaly and prediction engine.

7.4. Applications on Traffic Data Analysis

Several applications are already available for mobility, such as Crowdbus [69], Bus Brasil[8], Cadê o Ônibus[9], Itibus[10] and Moovit[11].

Crowdbus uses resources of crowdsourcing technology to provide data about the public transportation in Recife and Maceió where the crowdsourced data are collected by users support and user's smartphones functionalities (e.g., compass, GPS, and accelerometer). Crowdbus uses data from speed, routes to generate a quality standing for each route of public transportation, processing the data to provide, in the future application, measurements of time to travel between bus stops [69].

Bus Brasil uses an application for Android smartphone that stores bus time tables from various cities in Brazil. The application provides schedules for the buses, with the closest bus coming from a determined place, such as a Bus Terminal. The application can store data in the device, providing some functionalities while the device is not connected to the Internet.

Cadê o Ônibus was developed in cross-platform modal (Android, IoS and Windows Phone). This application detects the position of buses in real-time, allowing a variable number of search requests by the user. This allows a user collaboration (feature that is only showed in the Moovit and the Crowdbus apps), thanks to the use of the real-time tracking can also predict the arrival of the bus on the bus stop.

Itibus is a web-plataform application which provides the schedule of lines, lines by its code or label, itinerary of the lines in a map, real-time location of the bus, near location of bus stops and lines by stops. Besides that, the application can provide news and the balance of the client's transport card.

Moovit is another example of application which uses GIS and processes data from external sources to generate knowledge. The application operates in more than 2,500 cities, with more than 200 millions users. Under the concept of Urban Mobility Analysis and Mobility as a Service (MaaS), the system provides a list with buses lines, various types of search, lines by stops, route creation, and buses that accept transport card. The system can predict the arrival and departure times of the lines in stops and terminals.

[8] http://www.busbrazil.com.br/ – Last accessed on Ago 22, 2018.
[9] http://www.cadeoonibus.com.br/CoO/SiteV2 Last accessed on Nov 4th, 2017.
[10] https://www.urbs.curitiba.pr.gov.br/mobile/itibus – Last accessed on Ago 22, 2018.
[11] https://moovit.com/ – Last accessed on Ago 28, 2018.

Compared to these online applications (listed in Figure 20), our approaches present the following advantages: open source licenses, integration with several data sources (e.g. Waze, twitter, mobility open data) along with Big Data services (among others).

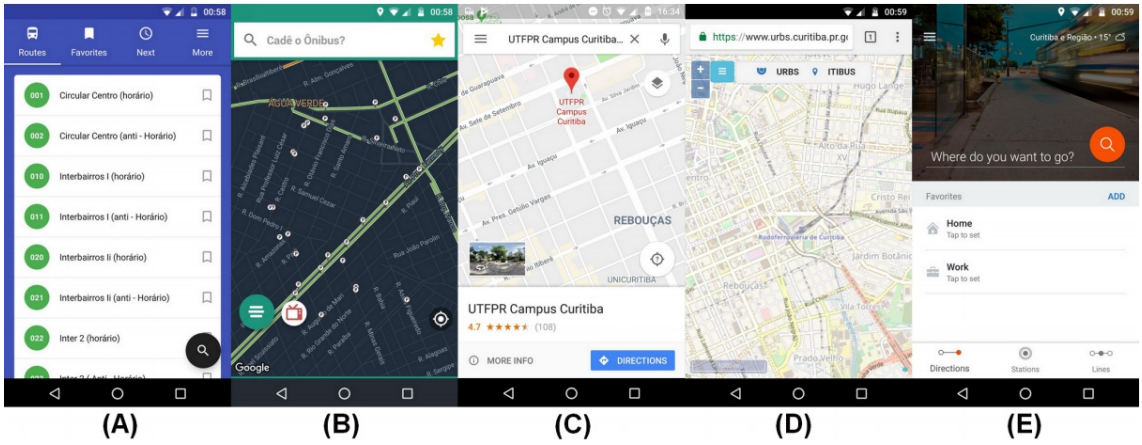


Figure 20: Home Screen of the applications: (A) Bus Brasil. (B) Cadê o Ônibus?. (C) Google Maps. (D) ItiBus. (E) Moovit. Source: [70]

For the case of a broad scope platform such as EUBra-BIGSEA, composed from different entities, built over different types of applications, on different cloud platforms, no one production solution presents the desirable features. Even worse, no existing solutions can be easily extended to cope with the challenges proposed. As highlighted by Natu et. al. [71], analytics-driven approaches need to be built, and they need to consider aspects that range from the scale and complexity of the monitored system, to the need to cope with sensitive metrics and incomplete views of the system. The EUBra-BIGSEA platform facilitates these developments.

7.5. Privacy Management

One of the features the platform offers is a way to handle sensitive data preserving data privacy and making the results compliant to laws and regulations such as GDPR. Some privacy-preserving frameworks for big data analytics platforms can be found in the literature. Drogkaris and Gritzalis [72], using hashing techniques to transform personal identifiers into digitaldata. Al-Zobbi et al. [73] proposed a framework for big data anonymity, implemented for data analytics, which provides an authorization method by applying anonymization in a finegrained access control. Basso et al. [74] proposed a framework which addresses anonymization challenges in a typical big data scenario. In those previous work there were no purposes for the analysis and implementation of a framework that uses policies to guide the anonymization processes of all components and a Data Utility/Re-identification Risk component acting in the final step of the analytics like the one proposed in the project. By using the policies and components the data utility is enlarged while data privacy is preserved.

7.6. Performance Prediction

From the performance perspective, big data applications have to share physical resources (processors, memory, bus, etc.). Competition for computational resources can occur among different applications (inter-application concurrency) or

among tasks of the same application (intra-application concurrency). Given system resource limitation, performance analysis techniques are important for studying fundamental performance measures, such as mean response time, system throughput, and resource utilisation. In this context, Queuing Networks (QNs) have been successfully used for studying the impacts of the resource contention and the queuing for service in the applications running on top of parallel systems [21, 75, 76] and have been adopted as reference models also in EUBra-BIGSEA.

The parallel execution of multiple tasks within higher level jobs is usually modeled in the QN literature with the concept of fork/join: jobs are spawned at a fork node in multiple tasks, which are then submitted to queuing stations modeling the available servers. After all the tasks have been served, they synchronise at a join node. Unfortunately, there is no known closed-form solution for fork-join networks with more than two queues, unless a special structure exists [77]. The work in [21] considers the issue of estimating performance metrics in parallel applications. The proposed method is computationally efficient and accurate for predicting performance of a class of parallel computations, which can be modeled as task systems with deterministic precedence relationships represented as series-parallel DAGs. While the models proposed in [78, 76] assume a fork-join abstraction to represent parallel behavior, here the authors focus on the precedence relationships resulting from tasks that must run sequentially, combined with those that may run in parallel.

An extension of this model, capturing not only intra-job, but also inter-job overlap to evaluate application response times, is presented in [75].

In our work, we apply the models proposed by the authors in [78] and [21], given that the parameters of both models are easily obtained (for instance, service demands and task structure) and results are obtained with low complexity cost. In [22] we demonstrated that QN models are accurate (we can predict average application execution times with 26% relative error in the very worst case and about 7% on average across both SQL-like and machine learning workloads). Moreover, models can be evaluated very quickly and hence are suitable to support run time adaptation decision of large Big Data clusters.

7.7. Data Analytics Interfaces

One of the goals of EUBra-BIGSEA is to provide a simple interface through which data scientists may 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 and machine learning tools support the creation of data flows visually by using building blocks on Graphical User Interface (*e.g.*, RapidMiner [79], Orange [80] and KNIME [81]). Most of these platforms do not include distributed execution features. Others, such as Microsoft Azure Machine Learning (ML) Studio [82] and ClowdFlows [83] allow tasks to be executed in distributed fashion, but do not include functionalities to exploit parallelism, manage a coherent authentication and authorisation model or provide data privacy annotation and enhancement tools. Moreover, they are bound to specific cloud deployments and providers.

Finally, it is important to state how the platform has been qualitatively assessed with respect to the requirements identified. The evaluation of the usability by final users is typically performed through standardised questionnaires [84]. For example, project VENUS-C deeloped a scientific PaaS and performed an analysis of the requirements fulfilment by collecting a set of CSUQ questionnaires from the 27 pilot

projects [85]. As BIGSEA platform have a reduced set of users (considering as users application developers and cloud offering administrators), a qualitative analysis was performed requesting the evaluation of each one of the 18 technical requirements [86].

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

The partners addressed a significant 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 characterise 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 a data scientist to describe their processing tasks. Our advantages against the competition include functionalities to exploit parallelism, a coherent authentication and authorisation model, and tools for data privacy annotation and enhancement, all in one package.

To satisfy the use cases and achieve our goals, we have proposed the infrastructure detailed in Figure 1. We address the need for 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 characterise 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 characterise 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 value for the scientist, the regular citizen, and municipality entities.

9. Acknowledgements

The work shown in this article has been funded jointly by the European Commission under the Cooperation Programme, Horizon 2020 grant agreement No 690116 (textitEUBRa-BIGSEA) and the Ministério de Ciência, Tecnologia e Inovação (MCTI) from Brazil.

The work is the result of a large collaborative project, and therefore the number of authors listed in the paper reflects this extensive collaboration. The authors want to state that:

- All authors have substantially contributed to the work, as it is explicitly described at the end of this section. The authors have agreed to be listed in alphabetic order.
- The article has been written collaboratively, and all authors have contributed to the text included in it, accepting to be accountable for any aspect related to the accuracy or integrity of the work.
- All authors have reviewed and approved the final version of the document.

The individual contributions of each one of the authors are listed next.

- Andy S Alic developed the Routes4tp web application and web service presented in section 5.1.1. He also managed the backend PostgreSQL database holding the GTFS information viewable in the web application, developed the Open Trip Planner Docker containers for various cities accessible from the Routes4tp web interface, administered the Marathon-LB load balancer, and assisted on the operation of the Mesos/Marathon cluster.
- Jussara Almeida developed and validated performance models (see sections 4.2 and 6.2) for evaluating the performance of big data applications. She explored several alternatives trading-off models accuracy and estimation time in various scenarios based on different applications and infrastructure setups.

- Giovanni Aloisio contributed to this work in the definition of the adaptation strategies of the Ophidia framework, targeting the elasticity of the deployment and its dynamic scalability, required for the integration of the framework in the textitEUBRa-BIGSEA QoS environment, together with the definition of City Administration Dashboard application architecture.

- Nazareno Andrade led the development of the Municipality Dashboard, of the backend services for the Routes4People application, and of the Melhor Busão app (all described in Section 5.1), as well as the predictive models described in Section 5.2.3.

- Nuno Antunes led the security evaluations and contributed to the development of the AAAaaS, as defined in Section 3. He was also involved in the definition and implementation of the PRIVaaS solution.

- Danilo Ardagna contributed to this work in the definition of the optimisation service (presented in sections 4.2 and validated in section 6.2) for identifying the cloud deployment of minimum cost which also fulfils an a priori deadline.

- Rosa M. Badia contributed in section 2 on the definition of the programming model interface that allows the integration with Lemonade and Ophidia.

- Tania Basso contributed to the definition of the Privacy as a Service (PRIVAaaS) architecture (described in section 3.2), as well as to the development of the solution and integration with data analytics platforms (Ophidia and Lemonade). It includes the definition of an anonymisation policy that helps to guarantee GDPR compliance.

- Ignacio Blanquer contributed to this work in the definition of the back-end cloud architecture and the implementation of the recipes for the automated deployment of the backend, as well as in the testing of the horizontal elasticity of this backend system on an OpenNebula cloud on-premises.

- Tarciso Braz contributed to the development of the Municipality Dashboard and led the creation and evaluation of the descriptive models for the Routes4People application, as well as the predictive models described in Section 5.2.3.

- Andrey Brito contributed to the specification and of the proactive policies component and helped on the integration of this component with the applications, especially regarding the OpenStack validation.

- Donatello Elia contributed to this work with the implementation of the "City Administration Dashboard" application blocks regarding parallel statistics computation and ETL procedures, the integration of the data privacy components, the deployment and testing of this application, as well as the implementation and testing of the recipes for the automated deployment of the Ophidia framework.

- Sandro Fiore contributed to this work in the integration of the Ophidia framework with the QoS-oriented cloud platform developed in the project and the extension of the framework for horizontal elasticity and job monitoring, as well as the design of the City Administration Dashboard application, including the

definition of data life-cycle and its management, and the integration of the application main components.

- Dorgival Guedes contributed to this work in the definition of the architecture of the Lemonade system, in particular in its integration with the COMPSs framework.

- Marco Lattuada participated in the development of the optimisation service (presented in section 4.2) and in the integration and validation of the project infrastructure (co-authoring section 6.2).

- Daniele Lezzi contributed on section 5.1.2 for the development of the COMPSs workflow using the Ophidia API, its execution in the Mesos cluster and the tests of the application; and section 4.2 on the tests to support the performance prediction using the QoS tools.

- Matheus Maciel was the main developer in the development of the Municipality Dashboard, of the Backend services for the Routes4People application, and of the Melhor Busão app (all described in Section 5.1), as well as the predictive models described in Section 5.2.3.

- Wagner Meira Jr. contributed to this work in the definition and development of the Lemonade platform, as well as the applications implemented on top of the platform.

- Demetrio Mestre contributed in section 5.2.1. on the modelling, development and evaluation of the performance of parallel Entity Matching approaches of the EMaaS (Entity Matching as a Service).

- Regina Moraes was responsible for the definition of the PRIVAaaS component, which adapts the anonymisation process including the re-identification risk and information loss calculation in the context of the project. These measurements help to guarantee the GDPR compliance. The main contribution is on Section 3, particularly in PRIVAaaS subsection.

- Fabio Morais was the developer lead for the vertical scalability and OpenStack integration and was responsible for the design, validation and integration of the components as shown in the paper.

- Carlos Eduardo Pires contributed in section 5.2.1. on the modelling and development of the EMaaS (Entity Matching as a Service).

- Nádia P. Kozievitch and UTFPR contributed: 1) acquiring and integrating public urban data, 2) working the analysis of specific use cases (such as speed cameras, accidents, pollution and traffic).

- Walter dos Santos contributed to the definition and development of Lemonade Platform, integration with Apache Spark, COMPSs, HDFS and Apache Mesos.

- Paulo Silva contributed to this article in the definition of the Authentication, Authorization and Accounting as a Service (AAAaaS) architecture, as well as to the development and integration with applications and infrastructure (described in section 3.1).

- Marco Vieira contributed to the definition of the security and privacy model presented in Section 3, particularly in what regards the definition of architecture for the AAAaaS and the trustworthiness characterisation and security assessments.

[1] Francesc Lordan, Enric Tejedor, Jorge Ejarque, Roger Rafanell, Javier Álvarez, Fabrizio Marozzo, Daniele Lezzi, Raül Sirvent, Domenico Talia, and Rosa M. Badia. Servicess: An interoperable programming framework for the cloud. *Journal of Grid Computing*, 12(1):67–91, Mar 2014.

[2] Zaharia, M. and Chowdhury, M. and Franklin, M. J. and Shenker, S. and Stoica, I. Spark: Cluster computing with working sets. In *Proceedings of the 2nd HotCloud Usenix Conference on Hot Topics in Cloud Computing*. USENIX, 2010.

[3] Lemonade: Live exploration and mining of a non-trivial amount of data from everywhere. http://www.lemonade.org.br/. Accessed: 2018-10-30.

[4] Enric Tejedor, Yolanda Becerra, Guillem Alomar, Anna Queralt, Rosa M Badia, Jordi Torres, Toni Cortes, and Jesús Labarta. Pycompss: Parallel computational workflows in python. *The International Journal of High Performance Computing Applications*, 31(1):66–82, 2017.

[5] S. Fiore, A. D'Anca, C. Palazzo, I. Foster, D.N. Williams, and G. Aloisio. Ophidia: Toward big data analytics for escience. *Procedia Computer Science*, 18:2376 – 2385, 2013. 2013 International Conference on Computational Science.

[6] Ophidia, a big data analytics framework for escience. http://ophidia.cmcc.it. Accessed: 2018-02-11.

[7] Donatello Elia, Sandro Fiore, Alessandro D'Anca, Cosimo Palazzo, Ian Foster, and Dean N. Williams. An in-memory based framework for scientific data analytics. In *Proceedings of the ACM International Conference on Computing Frontiers*, CF '16, pages 424–429, New York, NY, USA, 2016. ACM.

[8] Pyophidia, python bindings to ophidia. https://anaconda.org/conda-forge/pyophidia. Accessed: 2018-02-11.

[9] Ophidia cluster ansible role. https://galaxy.ansible.com/OphidiaBigData/ophidia-cluster/. Accessed: 2018-02-11.

[10] Ivano Alessandro Elia, Nuno Antunes, Nuno Laranjeiro, and Marco Vieira. An analysis of openstack vulnerabilities. In *2017 13th European Dependable Computing Conference (EDCC)*, pages 129–134. IEEE, 2017.

[11] Andre Ferreira, Tania Basso, Hebert Silva, and Regina Moraes. Priva: a policy-based anonymization library for cloud and big data platform. In *Proceedings of the XVIII Workshop de Testes e Tolerância a Falhas (WTF)*, pages 1–11, 2017.

[12] Council of European Union. Regulation (eu) no 2016/679 of the european parliament and the council. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679. Accessed: 2018-10-09.

[13] Fabian Prasser and Florian Kohlmayer. Putting statistical disclosure control into practice: The arx data anonymization tool. In *Medical Data Privacy Handbook*, pages 111–148. Springer, 2015.

[14] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[15] Ansible home page. https://www.ansible.com. Accessed: 2018-02-05.

[16] Miguel Caballer, Ignacio Blanquer, Germán Moltó, and Carlos de Alfonso. Dynamic management of virtual infrastructures. *Journal of Grid Computing*, 13(1):53–70, Mar 2015.

[17] Oasis topology and orchestration specification for cloud applications (tosca). https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca. Accessed: 2018-02-05.

[18] Amanda Calatrava, Eloy Romero, Germán Moltó, Miguel Caballer, and Jose Miguel Alonso. Self-managed cost-efficient virtual elastic clusters on hybrid cloud infrastructures. *Future Generation Computer Systems*, 61:13 – 25, 2016.

[19] F. Alvarruiz, C. de Alfonso, M. Caballer, and V. Hern'ndez. An energy manager for high performance computer clusters. In *2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications*, pages 231–238, July 2012.

[20] Ignacio Blanquer Sergio López-Huguet. Vertical elasticity on marathon and chronos frameworks - under review. *Journal of Parallel and Distributed Computing*.

[21] V. W. Mak and S. F. Lundstrom. Predicting performance of parallel computations. *IEEE Trans. Parallel Distrib. Syst.*, 1(3):257–270, July 1990.

[22] D. Ardagna, E. Barbierato, A. Evangelinou, E. Gianniti, M. Gribaudo T. B. M. Pinto, A. Guimarães, A. P. Couto da Silva, and J. M. Almeida. Performance Prediction of Cloud-Based Big Data Applications. In *ICPE 2018 Proceedings*, pages 192–199, 2018.

[23] Monasca. monasca, an openstack community project. http://monasca.io. Accessed: 2018-03-12.

[24] http://www.denatran.gov.br/images/Resolucoes/Resolucao6002016_new. Accessed: 2018-10-31.

[25] http://www.denatran.gov.br/download/Resolucoes/RESOLUCAO_CONTRAN_396_11. Accessed: 2018-10-31.

[26] https://www.state.nj.us/transportation/eng/documents/speedhumps/. Accessed: 2018-10-31.

[27] Department of Transport - Regional development of Northern Ireland. Traffic Calming. Technical report, 2007.

[28] Columbia District Department of Transportation. Ddot speed hump engineering guide request procedures and lines. Technical report, 2010.

[29] Tiago Brasileiro Araújo, Cinzia Cappiello, Nadia Puchalski Kozievitch, Demetrio Gomes Mestre, Carlos Eduardo Santos Pires, and Monica Vitali. Towards reliable data analyses for smart cities. In *Proceedings of the 21st International Database Engineering & Applications Symposium*, IDEAS 2017, pages 304–308, New York, NY, USA, 2017. ACM.

[30] Daniele Quercia, Rossano Schifanella, and Luca Maria Aiello. The Shortest Path to Happiness: Recommending Beautiful, Quiet, and Happy Routes in the City. *Proceedings of the 25th ACM conference on Hypertext and social media*, pages 116—-125, 2014.

[31] Danhuai Guo, Ziqi Zhao, Wei Xu, Jinsong Lan, Tao Zhang, and Shuguang Liu. How to Find a Comfortable Bus Route - Towards Personalized Information Recommendation Services. pages 1–11, 2015.

[32] Heshan Du, Natasha Alechina, Michael Jackson, and Glen Hart. A method for matching crowd-sourced and authoritative geospatial data. *Transactions in GIS*, 21(2):406–427, 2017.

[33] Tiago Brasileiro Araújo, Cinzia Cappiello, Nadia Puchalski Kozievitch, Demetrio Gomes Mestre, Carlos Eduardo Santos Pires, and Monica Vitali. Towards reliable data analyses for smart cities. In *Proceedings of the 21st International Database Engineering & Applications Symposium*, pages 304–308. ACM, 2017.

[34] Peter Christen. *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection.* Springer Science & Business Media, 2012.

[35] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning).* The MIT Press, 2005.

[36] Rudy Raymond and Takashi Imamichi. Bus trajectory identification by map-matching. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 1618–1623. IEEE, 2016.

[37] T. Braz, M. Maciel, D. G. Mestre, N. Andrade, C. E. Pires, A. R. Queiroz, and V. B. Santos. Estimating inefficiency in bus trip choices from a user perspective with schedule, positioning, and ticketing data. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–12, 2018.

[38] Miguel Caballer, Sahdev Zala, Álvaro López García, Germán Moltó, Pablo Orviz Fernández, and Mathieu Velten. Orchestrating complex application architectures in heterogeneous clouds. *Journal of Grid Computing*, 16(1):3–18, Mar 2018.

[39] Campos I., Fernández del Castillo E., Heinemeyer S., and et al. Phenomenology tools on cloud infrastructures using openstack. *The European Physical Journal C*, 73:2375, 2013.

[40] S Toor, L Osmani, P Eerola, O Kraemer, T Lindén, S Tarkoma, and J White. A scalable infrastructure for cms data analysis based on openstack cloud and gluster file system. *Journal of Physics: Conference Series*, 513(6):062047, 2014.

[41] Sánchez-Expósito S., Martín P., J.E. Ruiz, and et al. Web services as building blocks for science gateways in astrophysics. *J Grid Computing*, 14(4):673–685, 2016.

[42] Michael T. Krieger, Oscar Torreno, Oswaldo Trelles, and Dieter Kranzlmüller. Building an open source cloud environment with auto-scaling resources for executing bioinformatics and biomedical workflows. *Future Generation Computer Systems*, 67:329 – 340, 2017.

[43] Stockton D.B. and Santamaria. Automating neuron simulation deployment in cloud resources. *Neuroinformatics*, 15(1):51–70, 2017.

[44] S. Distefano and G. Serazzi. Performance driven ws orchestration and deployment in service oriented infrastructure. *J Grid Computing*, 12(2):347–369, 2014.

[45] Kacsuk P., G. Kecskemeti, Kertesz A., and et al. Infrastructure aware scientific workflows and infrastructure aware workflow managers in science gateways. *J Grid Computing*, 14(4):641–654, 2014.

[46] Yong Zhao, Youfu Li, Ioan Raicu, Shiyong Lu, Wenhong Tian, and Heng Liu. Enabling scalable scientific workflow management in the cloud. *Future Generation Computer Systems*, 46:3 – 16, 2015.

[47] OpenStack Foundation. Heat orchestration template (hot) guide. https://docs.openstack.org/heat/latest/template_guide/hot_guide.html, 2018.

[48] Opennebula: Oneflow. http://docs.opennebula.org/5.2/advanced_components/application_flow_and_auto-scaling/index.html, 2017.

[49] Cloudify. http://getcloudify.org, 2018.

[50] Opentosca. http://www.opentosca.org/, 2018.

[51] Aria. http://ariatosca.incubator.apache.org, 2017.

[52] T. Metsch and A. Edmonds. Open cloud computing interface-restful http rendering, 2011.

[53] S. Yangui, IJ. Marshall, JP. Laisne, and et al. Compatibleone: The open source cloud broker. *J Grid Computing*, 12(1):93–109, 2014.

[54] Infrastructure manager. http://www.grycap.upv.es/im, 2017.

[55] Egi fedcloud. https://www.egi.eu/federation/egi-federated-cloud, 2018.

[56] Hadoop. http://hadoop.apache.org. Accessed: 2018-10-09.

[57] Poliakov A. Stonebraker M., Brown P. and Suchi R. The architecture of scidb. In Heidelberg Springer-Verlag, Berlin, editor, *Proceedings of the 23rd international conference on Scientific and statistical database management (SSDBM'11)*, pages 1–16, 2011.

[58] P. Baumann, A. Dehmel, P. Furtado, R. Ritsch, and N. Widmann. The multidimensional database system rasdaman. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 575–577, New York, NY, USA, 1998. ACM.

[59] Wolfgang Gerlach, Wei Tang, Kevin Keegan, Travis Harrison, Andreas Wilke, Jared Bischof, Mark D'Souza, Scott Devoid, Daniel Murphy-Olson, Narayan Desai, et al. Skyport: container-based execution environment management for multi-cloud scientific workflows. In *Proceedings of the 5th International Workshop on Data-Intensive Computing in the Clouds*, pages 25–32. IEEE Press, 2014.

[60] Charles Zheng and Douglas Thain. Integrating containers into workflows: A case study using makeflow, work queue, and docker. In *Proceedings of the 8th International Workshop on Virtualization Technologies in Distributed Computing*, pages 31–38. ACM, 2015.

[61] Paolo Di Tommaso, Emilio Palumbo, Maria Chatzou, Pablo Prieto, Michael L Heuer, and Cedric Notredame. The impact of Docker containers on the performance of genomic pipelines. *PeerJ*, 3:e1273, jul 2015.

[62] Maria Fazio, Antonio Celesti, Rajiv Ranjan, Chang Liu, Lydia Chen, and Massimo Villari. Open issues in scheduling microservices in the cloud. *IEEE Cloud Computing*, 3(5):81–88, 2016.

[63] Alois Mayr, Peter Putz, Dirk Wallerstorfer, and Anna Gerber, 2017.

[64] Zabbix - monitor anything (solutions for any kind of it infrastructure, services, applications, resources). https://www.zabbix.com. Accessed: 2018-03-12.

[65] Sensu. full-stack monitoring for today's business. https://sensuapp.org/. Accessed: 2018-03-12.

[66] Influxdata - the modern engine for metrics and events (the complete time series platform). https://www.influxdata.com/. Accessed: 2018-03-12.

[67] Elastic. the open source elastic stack. https://www.elastic.co/products. Accessed: 2018-03-12.

[68] Kubernetes. production-grade container orchestration. https://kubernetes.io/. Accessed: 2018-03-12.

[69] Sotero Rocha de Sousa Junior, Rodrigo dos Santos Lima, and Rodrigo Augusto Honório da Cunha. Crowdbus: Aplicativo crowdsourcing para informação, localização, avaliação e fiscalização de frotas de ônibus. *SEGeT : Simpósio de Excelência em Gestão e Tecnologia*, XI, 2014.

[70] A Calandre, B Pasquim, E Santos, and J Oliveira. *MYURB: Assistente de Mobilidade em Curitiba através do Transporte Público*. monography, UTFPR, 2018.

[71] Maitreya Natu, Ratan K Ghosh, Rudrapatna K Shyamsundar, and Rajiv Ranjan. Holistic performance monitoring of hybrid clouds: Complexities and future directions. *IEEE Cloud Computing*, 3(1):72–81, 2016.

[72] P. Drogkaris and A. Gritzalis. A privacy preserving framework for big data in e-government environments. In *Proceedings of the International Conference on Trust and Privacy in Digital Business*, pages 210–218. Springer, 2015.

[73] M. Al-Zobbi, S. Shahrestani, and C. Ruan. Implementing a framework for big data anonymity and analytics access control. In *Proceedings of Trustcom/ BigDataSE/ ICESS*, pages 873–880. IEEE, 2017.

[74] Tania Basso, Roberta Matsunaga, Regina Moraes, and Nuno Antunes. Challenges on anonymity, privacy, and big data. In *Proceedings of the Seventh Latin-American Symposium on Dependable Computing (LADC)*, pages 164–171. IEEE, 2016.

[75] Satish K. Tripathi and De-Ron Liang. On performance prediction of parallel computations with precedent constraints. *IEEE Transactions on Parallel & Distributed Systems*, 11(undefined):491–508, 2000.

[76] Don Towsley, John C.S. Lui, and Richard R. Muntz. Computing performance bounds of fork-join parallel programs under a multiprocessing environment. *IEEE Transactions on Parallel & Distributed Systems*, 9(3):295–311, 1998.

[77] Deron Liang and Satish K. Tripathi. On performance prediction of parallel computations with precedent constraints. *IEEE Trans. Parallel Distrib. Syst.*, 11(5):491–508, 2000.

[78] Randolf D. Nelson and Asser N. Tantawi. Approximate analysis of fork/join synchronization in parallel queues. *IEEE Trans. Computers*, 37(6):739–743, 1988.

[79] Ingo Mierswa et al. YALE: Rapid Prototyping for Complex Data Mining Tasks. In *Proc. of the 12th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining (KDD'06)*, pages 935–940, New York, NY, USA, 2006. ACM.

[80] Janez Demšar et al. Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14:2349–2353, 2013.

[81] Michael R. Berthold et al. KNIME - the Konstanz Information Miner: Version 2.0 and Beyond. *SIGKDD Explor. Newsl.*, 11(1):26–31, November 2009.

[82] Microsoft. Microsoft Azure: Machine Learning. https://azure.microsoft.com/pt-pt/services/machine-learning/, 2016. Visited on: 2016-12-12.

[83] Janez Kranjc et al. ClowdFlows: Online workflows for distributed big data mining. *Future Generation Computer Systems*, 68:38–58, 2017.

[84] J. R. Lewis. Ibm computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. international. *Journal of Human-Computer Interaction*, 7:57–78, 1995.

[85] Ignacio Blanquer, Goetz Brasche, and Daniele Lezzi. Requirements of scientific applications in cloud offerings. In *6th Iberian Grid Infrastructure Conference Proceedings*, pages 173–182. IBERGRID, 2012.

[86] I. Blanquer, A.S. Alic, A. Calatrava, S. Fiore, W. dos Santos, and W. Meira Jr. D7.6: Validation of the requirements, 2018.