INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Real-time data analytics for Non-Functional Requirements satisfaction

RITA ROCHA DE SOUSA
outubro de 2021

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Real-time data analytics for Non-Functional Requirements satisfaction

Rita Sousa

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Information and Knowledge Systems

Supervisor: Doctor Luís Nogueira
Co-Supervisor: Doctor Fátima Rodrigues

Evaluation Committee:
President:
Doctor Jorge Santos, Professor, DEI/ISEP

Members:
Doctor Jorge Coelho, Professor, DEI/ISEP

Porto, October 15, 2021

Dedicatory

I want to express my special thanks of gratitude to my teachers Luís Nogueira, António Barros e Luís Miguel Pinho who allowed me to contribute to the ELASTIC project and Fátima Rodrigues, who helped me a lot to learn more on the topic of Machine Learning. Furthermore, I will always have to thank Cláudio Maia for all the hours of conversation that always help me make the best decisions and inspire me to want to be better, and Bruno Pinho for all the tireless support and calm that he gave me. Finally, I would like to thank my beloved family and friends, particularly Margarida Guerra, Gil Durão and Joel Teixeira, who helped me a lot in finalizing this project.

Abstract

Smart systems, more than ever, demands processing a massive amount of data generated by heterogeneous and distributed data sources. This dissertation presents the contribution made to a software architecture that processes big-data analytics from the edge to the cloud. The software architecture integrates both responsive data-in-motion (edge computing) and latent data-at-rest analytics (cloud computing) into a single solution, satisfying extreme-scale analytics performance requirements. This dissertation focused on fulfilling the non-functional properties inherited from smart systems, such as real-time and energy-efficiency, to ensure the performance of the software architecture first referred. The Non-Functional Requirements (NRF) Tool manages computing resources and detects Quality of Service (QoS) violations. The Global Resource Manager (GRM) helps the scheduler/orchestrator redeploy the applications through the NFR Tools' feedback. In addition, it can act reactively or proactively (recurring to Machine Learning techniques) to improve the system's health.

Keywords: Non-Functional Requirements, Quality of Service, Real Time, Machine Learning

Resumo

Os sistemas inteligentes, mais do que nunca, exigem processamento de grandes quantidades de dados gerados por fontes de dados heterogéneos e distribuídos. Esta tese apresenta a contribuição para uma arquitetura de software que processa análises de big-data desde a Edge até à Cloud. A arquitetura de software integra, numa única solução, dados em movimento responsivos (Edge computing) e dados analíticos em repouso latentes (Cloud computing), atendendo aos requisitos de desempenho de análises em escala extrema. O foco desta tese incide no cumprimento das propriedades não funcionais herdadas de sistemas inteligentes, como tempo real (Real Time) e eficiência energética, para garantir o desempenho da arquitetura de software inicialmente referida. A ferramenta de requisitos não funcionais (Non-Functional Requirements (NFR)) faz a gestão de recursos computacionais e deteta violações de Qualidade de Serviço (Quality of Service (QoS)). O gestor de recursos global ajuda o escalonador/orquestrador a reescalonar as aplicações através do feedback da NFR e pode agir de forma reativa ou proativa (recorrendo a técnicas de Aprendizagem de Máquina (Machine Learning (ML)) para melhorar a integridade do sistema.

Palavras-chave: Requisitos não funcionais, Qualidade de Serviço, Tempo Real, Aprendizagem de Máquina

Acknowledgement

This work has been financially supported by the European commission through the ELASTIC project (H2020 grant agreement 825473).

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence.
AMQP	Advanced Message Queuing Protocol.
C	Programming language.
CaaS	Containers as a Service.
CPU	Central Processing Unit.
ELASTIC	A software architecture for **E**xtreme-Sca**L**e Big-Data **A**nalytic**S** in Fog Compu**TI**ng e**C**osystems.
GRM	Global Resource Manager.
I/O	Input/Output.
IoT	Internet of Things.
LSN	List of Sorted Nodes.
ML	Machine Learning.
MQTT	Message Queuing Telemetry Transport.
NFR	Non-Functional Requirements.
OSs	Operating systems.
QoS	Quality of Service.
RAM	Random-Access Memory.
ROI	Return on Investment.
SA	Software Architecture.
SaaS	Software as a Service.
TCP/IP	Transmission Control Protocol/Internet Protocol.
TSF	Time Series Forecasting.
VM	Virtual Machine.
VPN	Virtual Private Network.

Chapter 1

Introduction

An increasing number of cities are investing in smart systems to improve citizens' lives, whether in mobility and transportation, environment, or infrastructure. According to [1], London and New York have been the top two smarter cities worldwide for the past six years. Independently of the implemented intelligent solution, vast amounts of data are being collected continually from a large variety of distributed physical sensors (also referred to as Edge Computing). It is then integrated into a system to be processed, modelled, evaluated, and next support decision-making activities that require high-performance computing (achieved in Cloud Computing) and big data analytics.

These heterogeneous systems raise several challenges. A significant one refers to non-functional properties inherited from the application domain, including real-time, energy-efficiency, quality of communications, and security. In cloud environments, providing real-time guarantees is arduous, and the communications are inconstant, unstable, and insecure since moving anything to the cloud increases the probability of suffering attacks that could potentially affect the safety assurance levels.

So, the software architecture must include mechanisms that enable the specification of the required non-functional properties. More specifically, it should incorporate: 1) an offline analysis which defines an initial configuration system after evaluating the most relevant parameters; and 2) an online analysis responsible for monitoring and analyzing the system dynamically, taking actions and decisions whenever a Quality of Service (QoS) violation is detected or, ideally, foreseen which raises another crucial challenge, computing resources forecast.

1.1 Context

This dissertation presents the work developed under the European Union's Horizon 2020 research project. It is termed A software architecture for **E**xtreme-Sca**L**e Big-Data **A**nalytic**S** in Fog Compu**TI**ng e**C**osystems (ELASTIC) [2]. The ELASTIC project has two main objectives. Firstly, the development of a software architecture that executes within a fully distributed environment so that smart systems can meet the high-performance requirements of applications that analyze extreme amounts of data (real-time big data) collected and integrated into a UNIX system. Secondly, the implementation of a realistic (yet visionary) smart mobility use-case in the City of Florence (Italy), where a large set of Internet of Things (IoT) sensors and computing resources, which are distributed along the Florence tramway network, will generate and send a huge amount of data for future (offline) analysis.

The ELASTIC expected goal is to reduce the number of incidents by 25% and the cost of prevention and maintenance in mobility systems by 30%. It also aims to improve the cities' traffic by 5%, thus enhancing the citizens' quality of life regarding safe mobility and service availability.

The ELASTIC Software Architecture (SA), the project's first objective, intends to provide the performance levels needed by extreme-scale analytics while fulfilling essential inherited non-functional requirements due to cyber-physical interactions. Figure 1.1 displays a summarized view[1] of the concepts used in the ELASTIC SA.

Represented on the left side is the workflow of two applications (Apps), A and B (green and blue, respectively), and below is the caption for the diagram on the right. The App A tasks' workflow starts in T1 (T means task), goes to T2, and finishes with T3. For App B, the workflow is more confused. It begins in T1; T2 and T3 are dependent on T1 and can execute in parallel; after T2 finishes, T4 starts running, and the same behavior happens for T4 and T5. On the right side is a possible configuration of both applications distributed in two different Nodes. As the caption describes, if a task has a darker color, it runs on the assigned Node. Considering App A, T1 and T2 are executed on Node A, while T3 on Node B. So, the workflow (outlined in red) starts in Node A and finishes in Node B.

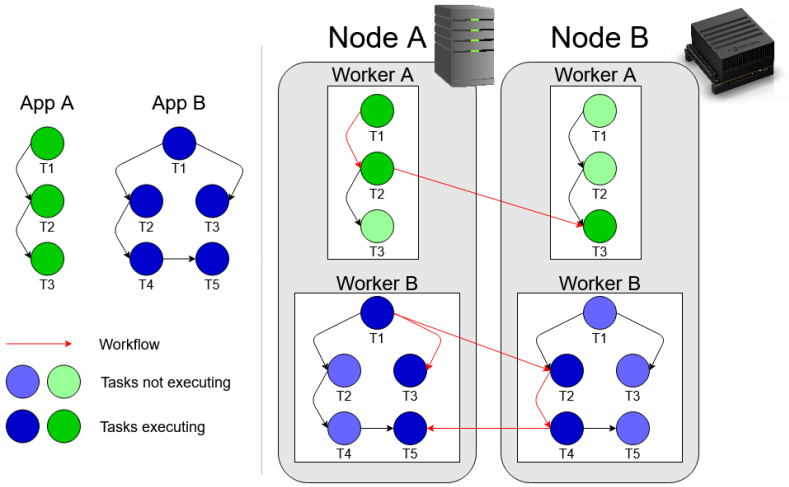

Figure 1.1: Conceptual view

Every Node is allocated somewhere across the compute continuum. In this example, Node A stays on a server (cloud computing) and Node B on the NVIDIA Jetson board (edge computing); the two applications (App A and B) are running spread over four different Workers; lastly, Workers are "dedicated" to execute tasks of a specific application.

One of the ELASTIC SA necessities is to guarantee that any Node does not commit any QoS violations. Suppose there is a QoS violation. In that case, the scheduler system must redistribute the application tasks to other Nodes, turning Workers "on" (initiating task execution) on the available Node and "off" (killing task execution) on the Node that committed QoS violations. However, this is not always the case; this dissertation pretends to add more approaches and solutions.

[1] This example only serves to explain the concepts further used throughout this dissertation. Section 2.2 addresses this with greater precision.

It is important to refer that the system scheduler is responsible for the tasks' distribution. Therefore, in this dissertation's context, Workers are black boxes, which means that the most precise granularity is at the Worker level; monitoring at the Worker level is the most accurate tracking possible. It is enough to have a single task running on a Worker to be active ("on") and consume computing resources. Otherwise, it will be inactive ("off"), with no resource consumption. Moreover, all the Nodes own the Non-Functional Requirements (NFR) Tool with the functionalities of obtaining metrics to monitor Workers locally properly (inside the same Node as the NFR Tool). When any QoS violation occurs, the NFR Tool acts according to heuristics/algorithms or policies, generally speaking, to advise the system scheduler, e.g., redirecting the tasks' workflow. Finally, the Workers and Nodes monitoring measures the resource usage of the executing applications. Consequently, having values to evaluate, it is possible to verify, for instance, if a Node is overloaded. The scenario previously presented will cause a re-schedule of the Workers, resulting in the disappearance or decrease in the number of violations.

1.2 Motivation

The fulfilment of the real-time, energy, communication, and security properties (the former two are the focus of this dissertation's work) required the development of a tool capable of monitoring the right metrics (CPU load, energy consumed, etc.) and evaluating the application's workload configuring the applications dynamically (online). For instance, the applications can provide accurate information on the tram position and assist drivers in the presence of obstacles in real-time. Hence, the need to monitor and evaluate the non-functional properties is fundamental for the proper functioning of applications avoiding delays, increasing efficiency, etc.

Although several heuristics and algorithms for resource management already exist, those that meet non-functional requirements and use smarter decisions are lacking. Based on this need, the primary motivation of this work is the development of heuristics to manage the workload of the applications (through the monitoring of non-functional requirements metrics) and at the same time make smarter decisions.

1.3 Objectives

The dissertation's main objective is to make decisions in the least reactive way possible, using the resource consumption forecast and making predictive decisions whenever possible. To be able to study and perform tests on the implemented algorithms, it is necessary to achieve other objectives, those being:

- Development of the Non-Functional Requirements (NFR) Tool that will operate during service execution, also referred to as online analysis (the offline analysis and the initial configuration/deployment system go beyond of this dissertation's objective);
- Build guarantees of the fulfillment of the system non-functional properties, specially real-time and energy, considering the potential trade-offs between performance, predictability, energy-efficiency, communication quality, and security;
- Identify concrete metrics as a basis for decision making and evaluation of the proposed solution which must support deployment decisions for the satisfaction of the system's overall requirements.

1.4 Contributions

The principal objective was focused on the study and experimentation of algorithms and heuristics to make decisions in the least reactive way possible. However, this dissertation brings other contributions, namely:

- Implementation of time and energy probes responsible for collecting useful metrics for the management of the resources;
- Implementation of the tool which scans the collected resource usage data locally and sends violations for a global tool;
- Implementation of the global tool in charge of making holistic decisions for better resource management;
- Study and implementation of intelligent algorithms and heuristics to support global and local decisions.

1.5 Dissertation Structure

The remainder of this document is divided into the following chapters beyond.

Chapter 1, *Introduction*, presented the dissertation's context, motivation, goals and contributions.

Chapter 2, *Context*, introduces the main concepts/technologies used throughout this dissertation and contextualizes in detail where the problem is on a larger framework, what are its limitations and dependencies as well as other significant background information. Finally, state-of-the-art presents several works and existing solutions/approaches considered vital for the achievement of the objectives of this dissertation.

Chapter 3, *Value Analysis*, presents the business value analysis made for this project.

Chapter 4, *Design*, proposes the problem solution design related with the tools implemented and how the tools inserted in the ELASTIC SA.

Chapter 5, *Proposed approach and experimentations*, gives an overview of the implementation of the tool with two behaviours, reactive and preventive/proactive.

Chapter 6, *Evaluation*, shows information related to the tests carried out to evaluate the implementation discussed above.

Chapter 7, *Conclusions*, summarizes and discusses the main results and contributions of this thesis.

Chapter 2

Context

Information Technology is overflowing with buzzwords, some better known than others, and others without an exact definition. To ensure this dissertation's understanding, Section 2.1 covers a few computing concepts used in this document. Furthermore, Section 2.2 details an overview of the ELASTIC project. Thus, the reader can understand the whole idea and, consequently, how this dissertation's contribution inserts in a real use case scenario, more specifically, in smart mobility. Section 2.3 describes the research methodology, serving almost as a warm-up for the Section 2.4, State of the Art.

2.1 Conceptual framework

2.1.1 Compute Continuum

In this context, compute continuum is nothing else but a concept that crosses the three layers presented in Figure 2.1, Cloud, Fog, and Edge computing, which complements each other. Each of these layers has its purpose, advantages/disadvantages, use cases, etc.

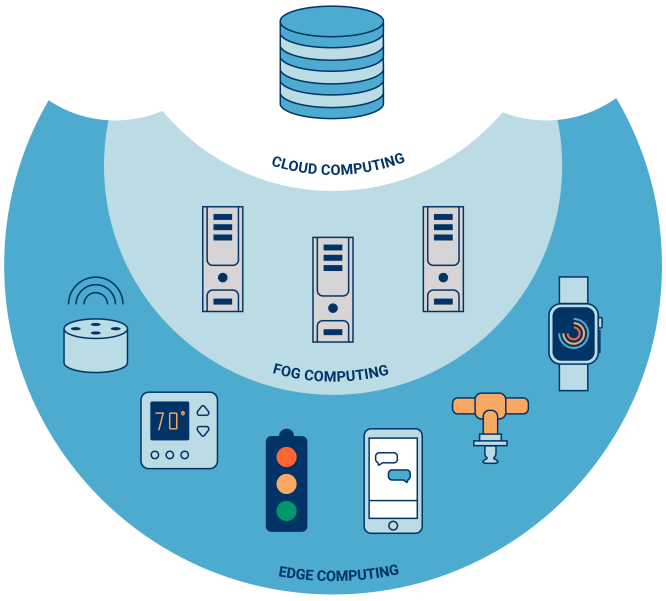

Figure 2.1: Compute continuum
Source [3]

To explore these three layers, we will peruse computing history with a high level of abstraction. In the first place, it stays centralized computing, composed with supercomputers in mainframes at a specific location where it was concentrated all the heavy lifting. Next, in the late 1970s and early 1980s, computing evolved to distributed computing; in late 1989, Tim Burners-Lee introduced World Wide Web [4], and grid computing takes place. Grid computing, in essence, manages heterogeneous networked resources to perform large tasks. It provides a grid of loosely coupled computers that can be accessed from anywhere, enabling resource sharing among the grid end-users [5][6]. Another concept very similar to grid computing vision, architecture, and technology, but different in several other aspects, is cloud computing [6].

Cloud Computing

According to [5, p. 1], "the industry says that Cloud computing is the fifth public resource ("the fifth utility") after water, electricity, gas, and oil" and it evolved from parallel computing, grid computing, utility computing, virtual computing, and Software as a Service (SaaS). In 2008, Larry Ellison, Andy Isherwood, and Richard Stallman insinuated that Cloud Computing (CC) is not a new technology or paradigm; it was just a set of technologies that already existed and were renamed Cloud computing without specifying a definition [7]. Compared with Grid computing, users of CC can stop worrying about infrastructure and concentrate on their applications and workload. Utility Computing consists of providing IT resources (e.g., computing and storage) within the user requirements, "users only pay according to their actual usage" [5, p.6]; and the way IT resources are exposed to the public in CC is significantly similar. Finally, SaaS, which is the delivery of the applications as services over the Internet, presents one of the three most known Service Models that Cloud Computing provides. The other two are Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

After extensive research, mostly based on papers of 2008, the authors in [8] identified 21 different definitions for Cloud Computing. There are numerous definitions, some more straightforward like "Cloud Computing is the sum of SaaS and Utility Computing" [7, p. 4] and others more complete like "Cloud computing is a model that enables ubiquitous, convenient, on-demand access to a shared pool of configurable computing resources that can rapidly be provisioned at any time and from any location via the Internet or a network" [9, p. 2].

One of the Cloud computing key incentives for this work is elasticity, which is the ability to increase or decrease resources quickly (e.g., storage, computing capacity) on-demand [10].

Continuing with the "history of computing", Cloud computing grants the infrastructure that has fueled the growth of mobile computing and Internet of Things (IoT) [11], and two new computing paradigms were conceptualized, edge and fog computing.

Edge Computing

Edge Computing (EC) is strongly linked with the Ubiquitous computing concept, which is fundamentally delivering computing power anytime from any device anywhere [11]. As Figure 2.1 suggests, EC is the layer of end-devices. As opposed to Cloud computing, EC is moving and distributing computing power closer to data sources, which gives a new processing capability at the edge of the network.

EC brings more efficiency to the implemented system since it can process data at the sources' proximity. Compared with CC, there will be lower network traffic/bandwidth, delays/latency, and response time, enabling faster (or even real-time) data processing and analysis and increases the efficiency of applications and network reliability and privacy/security.

In later 2017, Peter Levine affirmed that cloud computing was coming to an end [12]. He shared his thoughts like "what happens when cloud computing goes away?" and "I [Peter Levine] think it's actually happening right under our noses". He assumed that mobile devices would get more sophisticated. After seeing that this assumption would be a *cul-de-sac*, he stated that "it isn't mobile devices but all the other things [Internet of Things objects] that are going to be out at the edge that will truly transform cloud computing". Imagining a system with a network of security cameras to detect movement and edge computing is not used. All the data produced by the cameras have to be uploaded to the Cloud since there is no storage component in the cameras themselves. In the context of this dissertation, assume a tram driving and analyzing its surroundings through sensors installed in the tram. Even if it takes one or two seconds to send the data to the Cloud, analyzing it, and send it back to the edge, the tram will already be in a different position.

In conclusion, the tasks of measuring, monitoring, storing, analyzing, and processing the enormous amounts of data generated by devices, sensors, and applications create a significant challenge to be tackled with edge computing and another technology that end-devices are involved with, called fog computing.

Fog Computing

Fog computing takes place between the cloud and the edge/end-devices. This layer connects end-devices to the central servers aiding in minimizing latency, as in Edge computing. The ubiquitous computing term is also applied in Fog computing since it enables access to computing resources everywhere. However, Fog and Edge computing are different [11]. While edge computing focuses in putting computing power at the edge of a network, fog computing focuses on the infrastructure, that is, the network connections that enables the communication between edge devices and the cloud. This communication is performed with the so-called fog nodes, the core component in the fog computing layer [11]. Fog nodes can be physical components (such as switches, routers, or gateways) or virtual components (like virtual machines, cloudlets, etc.), and usually owns decent computing resources. Some of the few tasks that Fog computing could perform are: manage resources, perform data filtration, preprocessing, and security measures [13].

The rise in need, mainly for the last two paradigms presented (fog and edge computing), is due to the increase of generated data and data flows of the Internet of Things.

2.1.2 Internet of Things

Internet of Things (IoT) backs more than 20 years and was born with the Radio-Frequency IDentification (RFID)[1] technology. According to [15], the majority of IoT definitions diversify between three focus: the things, Internet-related aspects (protocols and network) or IoT semantic challenges (storage, large volumes of information, ...). However, Cisco Internet Business Solutions Group (IBSG) considered that IoT is the moment that the number of

[1] "Allows microchips to transmit the identification information to a reader through wireless communication"[14]

"things" connected in Internet surpass the number of people [16], which does not fit into any of the three focuses presented above.

A respectable definition is "[IoT] is a self-configuring, adaptive, and complex network that interconnects "things" which have a physical and virtual representation on the Internet according to standard communication protocols" [11, p. 304] [17, p. 74] [18, p. 4]. This definition was chosen since the authors in [17] evaluated a very wide range of several types of documents and environments (books, white papers, research projects, industrial activities, etc.) and because it covers properly six of the nine essential characteristics of IoT. Each characteristic will be presented in the order that appears in the definition cited:

- Self-configurability: IoT devices must manage themselves, in hardware/software configurations, network and resource usage, because remote or cloud-based control seems to be limited for scalability due to the quantity and heterogeneity of the devices;
- Programmability: can assume a variety of programmed behaviors, adapting itself without physical changes);
- Interconnection of things: complex network of interconnected "things";
- Uniquely identifiable things: due to physical and virtual representation;
- Connection of things to the Internet: enabling the communication among themselves;
- Interoperable communication capabilities: which refers to communication based on standardized and interoperable protocols.

Other three important characteristics, that are not included in the chosen definition, are:

- Sensing and actuation capabilities;
- Embedded intelligence: e.g. being capable of responding to external stimuli, becoming an extension of the human body and mind;
- Ubiquity: the network is available anytime and everywhere, more specifically, when and where it is needed.

In [16], the authors presented one peculiar example about how IoT is improving the people's lives, entitled as "Holy Cow!". The idea is to implant sensors in the ears of cattle to support farmers tracking cattle's health and, through this monitoring, improve meat quality. In the context of this dissertation, which does not deviate much from the final purpose of the previous example (to improve certain aspects in people's lives), smart mobility (or more generally, smart cities) is another example of IoT application area. Traffic congestion is the main problem to solve in transportation and IoT can help to fix it by providing real-time traffic information to improve commuter mobility. It can also try to reduce freight truck congestion by diversifying routes or fully closing roads for heavy vehicles [19]. There are other challenges like travelling efficiently, which can be solved encouraging greater use of sustainable transportation or car-sharing and even bike-sharing [11].

The diversity, heterogeneity, and large volume of data generated by IoT devices, be it wearable, smartphones, cars or even entire smart cities, needs to be processed. Data is very useful, but without treatment or processing and analysis to discover trends, to make predictions, or to simply share, becomes useless. Due to the initial characteristics pointed about

the data generated from IoT and the unsuitable capability of traditional database management systems to extract meaning from a huge volume of data, raises the necessity to use big data analytics [20].

2.1.3 Big Data

Every year since 2013, DOMO [2] publish an info-graphic denominated by 'Data Never Sleeps'. It essentially shows how much data is being generated every minute across different platforms, such as Netflix, Google, Youtube or Social networks. Focusing on the first two, while in 2017 Netflix users stream 69.444 hours of video and Google conducted 3.607.080 searches [22]. In 2020, Netflix users video hours stream grew to 404.444 [23], and in 2019, Google searches have grown to 4.497.420 [24]. These numbers were measured by means of ad clicks, which is only one example of the variety of data sources. More and more devices will be connected to the Internet and the amount of digital generated data will keep growing. Recent forecasts from IDC [25] predict "that by 2025 there will be 41.6 billion connected IoT devices or 'things' generating more than 79 zettabytes (ZB) [10^{21} bytes] of data" [26]. Although, Big data is not just about data volume [27].

Big data, similarly to the previous concepts presented, has a variety of definitions. However, there are five characteristics, commonly referred as "V's of Big data", that describes big data [28][29][30].

1. Volume: the size of the information. A large amount of data is generated from different sources, and usually the amount of data is between terabytes and exabytes;
2. Variety: the type of generated data. The data can be either text, images, videos, etc.; it is generated in varied formats usually classified into three types: structured, semi structured and unstructured data;
3. Velocity: the speed that the data is generated, following an analysis, and its delivery (taking an action/decision) in (near) real time;
4. Veracity: the generated data is inherently uncertain due to noise, incompleteness, and inconsistency; so, intelligent techniques must be applied to get higher accuracies and trustworthiness from the generated data [31];
5. Value: analysing all the data must bring some value to the business at hand, the data must be transformed into valuable insights to get well-informed decision, otherwise data will not have much meaningful value.

In the case of smart mobility, Big data analytics can reduce the pollution and road traffic congestion by finding the least polluted way from start to end locations or with parking information since "urban traffic experts estimate that 30% of the vehicles in the inner areas of big cities are looking for a parking space, and an average of 7.8 minutes is needed to find one" [32].

Big data analytics are expected to discover hidden patterns, unknown correlations, data trends, etc. that can bring valuable insights to support business decision making or to extract some knowledge and wisdom. And how to extract meaning and value from the data? In a broad perspective, through Artificial Intelligence techniques.

[2] Domo Inc. is a cloud software company specialized in business intelligence and data visualization. One of its competitors is Tableau Software [21]

2.1.4 Artificial Intelligence

Artificial Intelligence (AI) is a field where entities (i.e. machines) perform tasks similarly to humans and act with some intelligence. In 1959, Arthur Samuel coined defining it as, "the ability to learn without being explicitly programmed" [33]. Some examples of AI applications are autonomous vehicles (e.g. Waymo and Cruise are some of the Leaders), playing games (e.g. AlphaGo, Chess), image recognition (e.g. facial or object recognition), prediction of a judgement [34], etc. Searching and Planning, Reasoning and Knowledge Representation, Perception, Natural Language Process and Learning are the essential subfields in AI [35]. Needless to say, AI is full of techniques and algorithms to achieve a certain goal or, more usually, to solve a certain problem. One of the most recognised subfields in AI is Machine Learning (ML). ML is devoted to the study of algorithms that parse data, learn from it, and then decide or predict something from the data [36]. More and more concepts will be presented and more detailed since this work will explore some techniques to predict resource consumption. ML has already been stated in a very summarised way since the probability of the algorithms (that will be investigated) being from this sub-field of AI, is very high. By obtaining predictions with good accuracy, it may be possible to predict the best application reconfiguration to implement, instead of following a given limited list of reconfiguration options, which would be guided by simple heuristics that act in a reactive fashion. The State of the Art section presents some work already done on resource management that uses some "intelligent" algorithm. The next section will give a summarized view of types of Machine Learning, the presentation of the used algorithms and the phases of the ML process followed in this dissertation.

Machine Learning

There are several valid ML definitions, but consider the previously presented one and that is "a field of study that gives computers the ability to learn without being explicitly programmed" [33], which is one of the oldest definitions defined by Arthur Samuel. There at least four well known types of ML, which are Supervised, Unsupervised, Semi-supervised and Reinforcement Learning.

Supervised Learning has the data labelled, i.e. the correct/expected output (or the answers to the problem at hand) is already known, and the algorithms learn to predict the data from the input data. Contrary to the above, it is Unsupervised Learning, where all the data is not labelled, and the algorithms learn the inherent structure from the input data. In between these two types is Semi Supervised Learning where some of the data is labelled, but most of it is not. And finally, Reinforcement Learning which allows an agent to learn in an interactive environment through trial and error using feedback from its own actions and experiences [36].

The Figure 2.2 shows several algorithms in a tree structure that helps to understand in which subtype and type of Machine Learning they belong.

Since the data collected in this work contains the "responses" that are needed to be predicted, below stays a brief explanation of the experimented supervised algorithms, most of them usually (or uniquely) used in regression problems[3]:

[3]There are two types of problems in Supervised Learning, regression and classification. Regression has a continuous output (e.g. house price prediction) while classification has a small number of discrete outputs (e.g. tumor prediction, is it malign or benign).

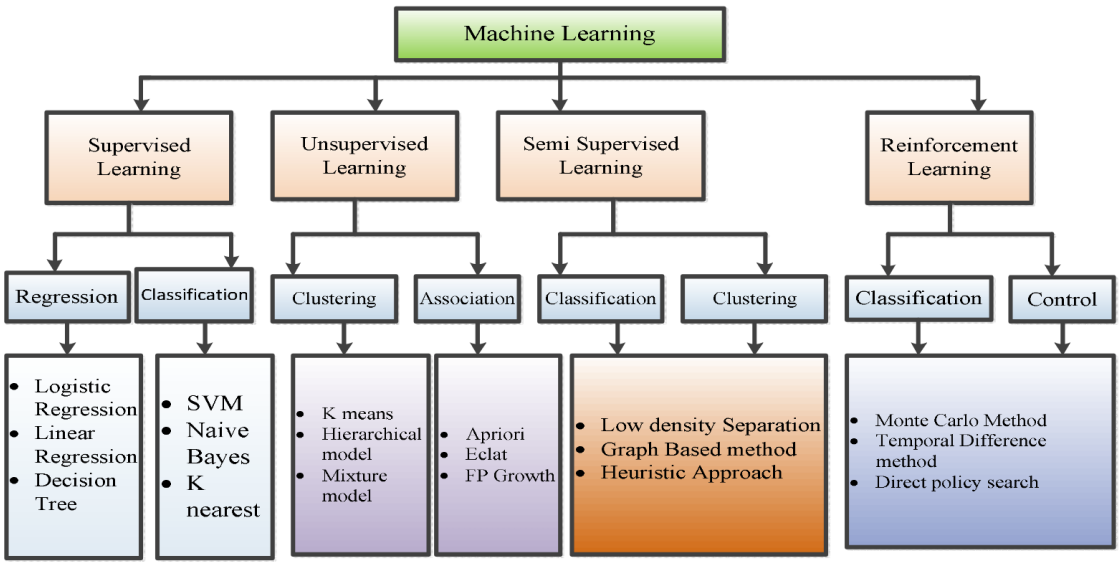

Figure 2.2: Types of ML algorithms [37]

- Multiple Linear Regression (MLR) is a form of Linear Regression where instead of having just one predictor (Simple Linear Regression), it can have two or more. Briefly, it measures the relationship between more than two variables by fitting a linear equation to the input data;
- Polynomial Regression (PR) is also a form of Linear Regression where the linear model can "bend" normally in order to increase the accuracy of the model, as Figure 2.3 shows; the higher the degree of the function the closer the model is to the values that are intended to be predicted, however, care must be taken as the model may suffer from other problems, such as overfitting;

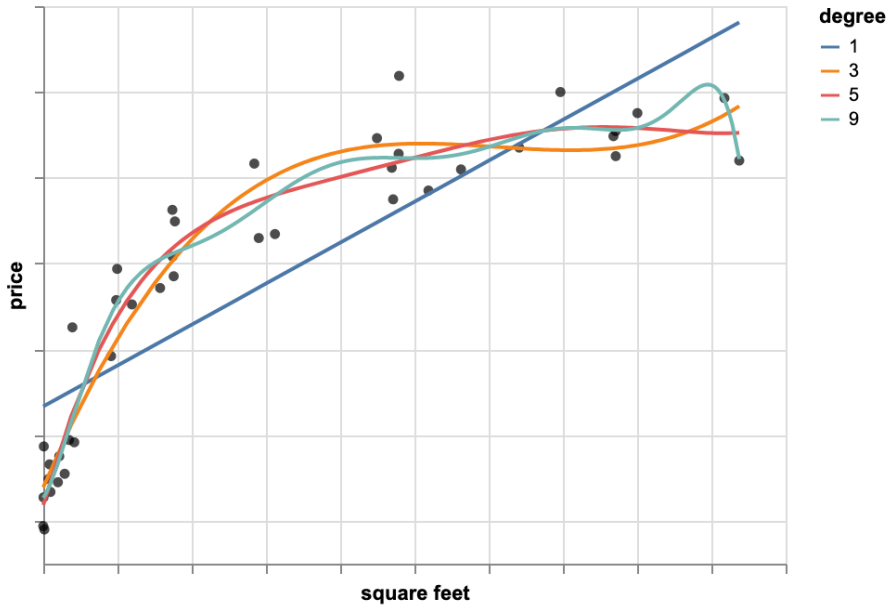

Figure 2.3: (Multi)linear vs. Polynomial Regression [38]

- Regression Trees (RT) is another example of nonlinear predictive model, which as the name suggests, it uses a tree-like model of decisions to solve situations where the data has many nonlinear features and the solution is subdividing the decisions into smaller regions until a point where simple models can be fitted in the smaller regions [39]; Figure 2.4 presents in the left a very simple tree, to find out how many points a basketball player will score based on his weight and height, and in the right the partitioned space (the first partition is represented by the blue line and the red lines the following partitions);

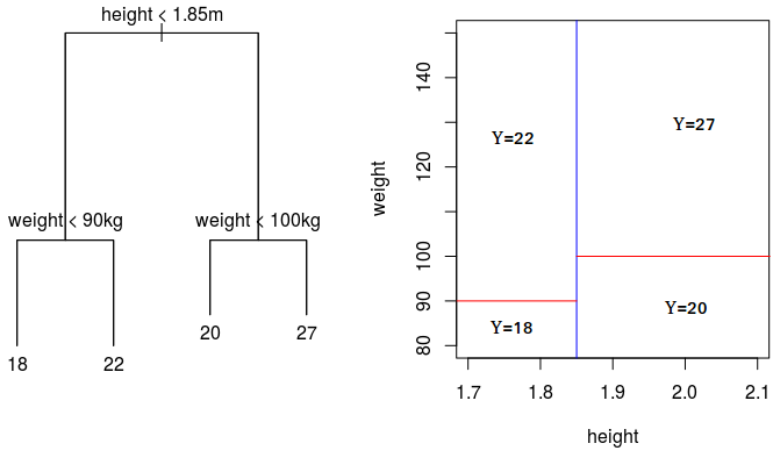

Figure 2.4: Simple example of a Regression Tree [40]

- Bagged Trees (BT) are basically a combination of decision/regression trees (obtained through bootstrapping which split data into smaller subsets/subtrees) and the result is the average of the resulting predictions of the several trees [41];
- Random Forest (RF) is based on ensemble learning, similar to BT. Unlike BT, RF force the tree to use only a subset of its available predictors to split in the growth phase. "Trees have one aspect that prevents them from being the ideal tool for predictive learning, namely inaccuracy" [41], but since RF is built with multiple decision trees and each tree is built on a different random subset, this leads to greater accuracy compared to using only one decision tree, and even bagged trees [41];
- Support Vector Regression (SVR) provides both linear and non-linear regression. For its full understanding, it is important to know the terms hyperplane (a line which help to predict continuous values), kernel (function that allows conversion between different dimensional spaces to aids in the search for a hyperplane) and boundary lines (two lines around the hyperplane to create a margin between the data points). The final goal is to incorporate the maximum possible data points within the boundary lines, as Figure 2.5 suggests;
- K-Nearest Neighbor (kNN) is usually used in classification problems but it can be also used in regression problems; if the model already has a lot of classified data, the distances (euclidean, manhattan or minkowski) of the K nearest neighbours are calculated to predict the numerical target; being the big question: which K should be chosen (one way to discover it is by using cross-validation) [43].

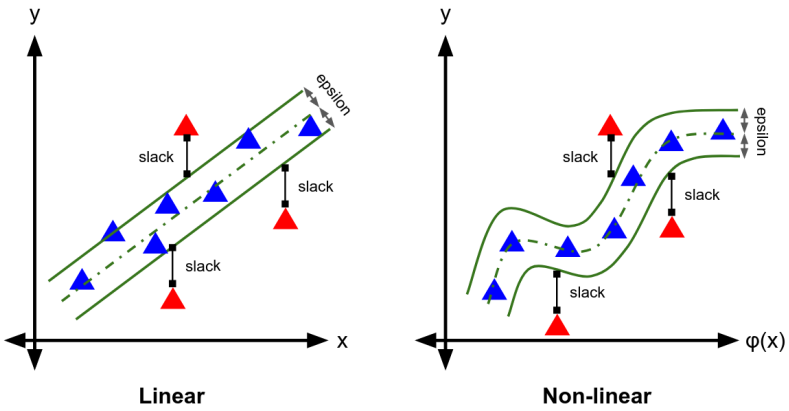

Figure 2.5: Example of Support Vector Regression with linear and non-linear kernel [42]

Beyond Supervised ML algorithms, there is another important area in ML,Time Series Forecasting (TSF), which is another way to predict the future, given that the data is a time series (it contains a time dimension). TSF uses historical time data, fit a model to that data and then extrapolate the future from the time series that presents only what has already happened. It is fundamental to know at least the following three components of the time series: trend (the data has some behaviour, e.g. increasing or decreasing, over time), seasonality (the data repeats patterns over time) and noise (some data that cannot be explained by the model) [44].

Additionally, TSF can be used as a supervised learning problem through the sliding window method. In Figure 2.6, it is possible to see what this method does, it restructure a time series dataset (right side) in order to use prior time steps to predict the next time steps (left side), being X the input for the model and y the output.

X,	y
?,	100
100,	110
110,	108
108,	115
115,	120
120,	?

time,	measure
1,	100
2,	110
3,	108
4,	115
5,	120

Figure 2.6: Example of time series dataset as supervised learning [44]

Finally, there are a methodology that must be followed in order to build a forecast model. According with Rob J Hyndman and George Athanasopoulos [45], they are:

- Problem Definition: It is considered the most difficult part of the process where a) who requires the forecast and b) for what the forecast will be used, must be defined;
- Gathering Information: Get data and the data expertise or the domain expert;
- Preliminary (exploratory) analysis: Graphing and summary statistics should be used to understand/discover patterns, trends, seasonality, and to spot any outliers in the data or data missing/corruption, etc.;

- Choosing and fitting models: Depending on the problem to solve and on the availability of historical data, choose and then evaluate a suite of models. The most important goal when building a model is to assure that the model should make predictions that will hold true when it is applied to yet unseen data. For that, the data needs to be split into two distinct sets: (1) the training set and (2) the test set. Models are created using the training set. Next, the performance of the models are estimated using the test set. The test set is entirely separate and distinct from the training set and is used to assess the expected accuracy of the model when it is applied to data outside the model set.
- Using and evaluating a forecasting model: Once the model produces forecast, evaluate/measure the performance of the forecast (the metrics to evaluate the model are better described in Chapter 4.

After the presentation of the essential concepts for this dissertation's understanding, the next section describes the ELASTIC project in order to understand how this dissertation's contribution fits in.

2.2 ELASTIC project overview

A software architecture for **E**xtreme-Sca**L**e Big-Data **A**nalytic**S** in Fog Compu**TI**ng e**C**osystems[4] is a project under the scope of the European Union's Horizon 2020. The ELASTIC project's ambition is to have full coordination between the edge and the cloud environment. It aims to develop a novel software architecture capable of:

- increasing software development and deployment productivity of systems requiring data-analytics distributed across the compute continuum, as it is needed in smart cities;
- increasing the level of performance of extreme-scale big-data systems by efficiently combine data-in-motion and data-at-rest analytics into a single complex workflow;
- guaranteeing real-time, energy-efficiency, quality of communication and, security non-functional properties.

Furthermore, some of the ELASTIC objectives are:

- to demonstrate the productivity, stressing the ELASTIC novel software architecture on a smart mobility use-case, specifically, in the tramway network of the city of Florence, Italy;
- to increase portability/scalability, extending elasticity across the compute continuum;
- to maximize performance, using all the performance capabilities across the compute continuum to reduce data latency and increase throughput speed-up.

Figure 2.7 presents a global view of the ELASTIC Software Architecture. Each component is described below mentioning the individual and complementary contribution to the realisation of the software architecture in the following fashion:

- **Distributed Data Analytics Platform** supports different kinds of analytic applications in Edge and Cloud environments. This platform must ensure application data accessibility wherever required and either based on real-time or historical values. Two

[4]Henceforth, the acronym ELASTIC will be used

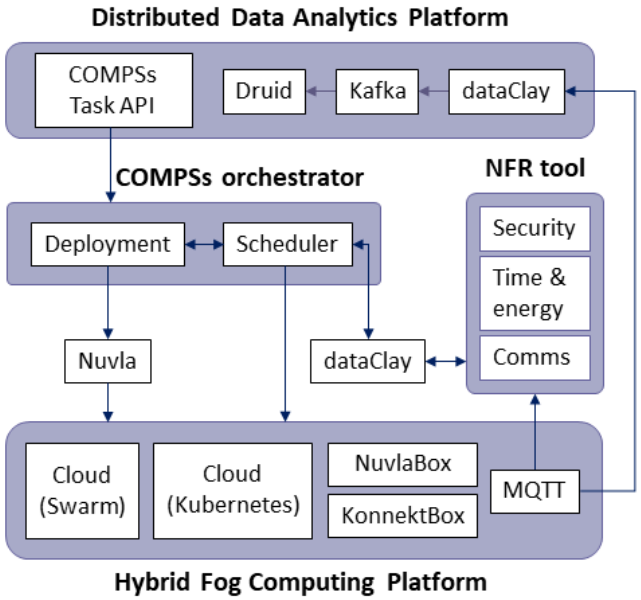

Figure 2.7: Global view of the ELASTIC Software Architecture
Source [2]

examples of the different data values sources and locality are the sensors and cameras installed on the trams and trams stops, respectively. The applications based on real-time values typically will be executed in the Edge, e.g., object detection in the tramway. In contrast, the offline analysis will run in the Cloud, based on Edge's historical values. A conclusion from the offline study could be the readjustment of traffic lights because of the detection of several people crossing the tramway at a particular time. In summary, the intention is to provide access to all the distributed data regardless of its source and function, without interfering between the different applications/tasks;

- **COMPSs orchestrator**, as the name implies, it is responsible for configuring and coordinating all of the applications' workflows. It must schedule and deploy the workflows guaranteeing the non-functional requirements. From the NFR Tool suggestions, COMPSs scheduler should dynamically adjust the workflows to fulfill these requirements. COMPSs [46] is an open-source framework developed and maintained by Barcelona Supercomputing Center[5] for the development and execution of applications in distributed environments. It enables parallel execution of workflows, and it is agnostic of the underlying infrastructure, which releases the programmer from caring about the location of tasks implementation and execution;

- **NFR tool** component is responsible for monitoring the execution of the workflows, identifying when a non-functional requirement is not achieved and apply heuristics to resolve the NFR violations (basically, Quality of Service violations), and find approaches for efficient management of resources. This real-time monitoring tool is distributed across different Edge Nodes and monitors several system properties such as the CPU usage, the energy consumption, the quality of the communications link between the Edge devices and the Cloud, and the security;

[5] Henceforth, the acronym BSC will be used

- **Hybrid Fog Computing Platform** provides an amalgamation of computing resources across the compute continuum that the software architect needs, among them:
  - Cloud-based Containers as a Service (CaaS) technologies for resource auto-scaling;
  - IoT cyber-secure communication and network protocols;
  - Advanced highly parallel and energy-efficiency embedded platforms.
- dataClay [47] is a distributed open-source data store developed at BSC; deals with various data structures (lists, graphs, dictionaries), as well as user-defined objects (supports object-oriented programming language, Python and Java); it accesses, traverses, and manipulates data in an optimized way by following references and invoking methods; insensitive of whether the information is persistent or in real-time; irrespective of its specific location, thus simplifying the application development; finally, it's Edge-to-Cloud data storage, which means it can run on a wide variety of devices, in this context, on trams or tram stops (NVIDIA Jetson, a series of embedded computing boards), as well as in data centres, including anything among them, like laptops;
- Nuvla [48] is an online open-source platform that facilitates the management of applications across the compute continuum developed at SixSq. In essence, it provides Edge Computing as a Service and deals with infrastructures and applications. The main features are a) management capabilities to handle application and infrastructure management, public and private app store, among others; b) data management functionalities to couple datasets with application deployments; c) end-to-end security, which includes access to Virtual Private Network (VPN) connection to your edge fleet; and d) edge control to remotely manage all edge devices and infrastructure.

Each of the smaller boxes, inside each previously elucidated components, will not be explained since it goes beyond this dissertation's scope.

Considering what was said earlier, an example of a simplified version of the object detection algorithm that will be used in the ELASTIC use case is presented in Figures 2.8 and 2.9. Also, Figure 2.9 helps to understand how COMPSs and NFR Tool components interact between them.

The flowchart on the top of Figure 2.8 refers to the object detection algorithm. This single application captures images from a camera, and from the frames received, it performs the object detection. It identifies the type (car, pedestrian, etc.) and position (object tracking) for each detected object, and finally, it displays the output on a screen. On the bottom of Figure 2.8 stays the dependency graph created by COMPSs. One of the features of COMPSs is to transform sequential code into a distributed workflow through the previous identification of certain functions/methods as COMPSs tasks and its respective dependencies. The task graph also identifies all the potential parallelism behind this application.

Then, as Figure 2.9 shows, COMPSs Master, who only possesses the knowledge of the configuration application, distributes the COMPSs tasks, who will run in COMPSs Workers, within the compute continuum. In this case, each Node has one Worker, and Node A has the COMPSs Master additionally.

Suppose that Node A and C are NVIDIA Jetson AGX Xavier located in a tram stop and a cabinet with a camera respectively, and Node B is a cloud facility. Also, we can assume the COMPSs has a master and worker architecture.

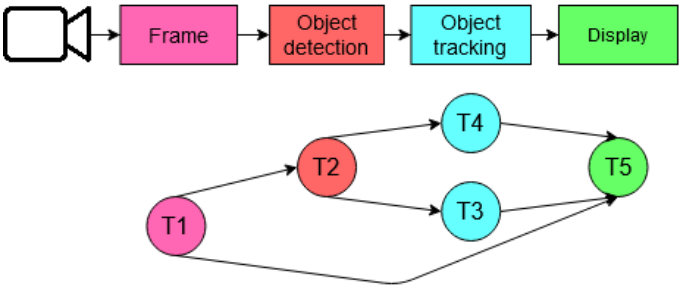

Figure 2.8: Flowchart of the ELASTIC use case and dependency graph of COMPSs tasks

The master has a configuration file with the essential characteristics of only available computer resources, which could be containers, virtual machines and cloud providers. Once this information is available, the master is responsible for deploying the Workers. And after that, the runtime of the COMPSs manages its distribution based on some scheduling policy.

Now, the COMPSs is responsible for transfer data to the Workers where the task will execute, respecting the dependencies.

ELASTIC incorporates a real-time monitoring tool, called NFR Tool, that enhances schedule decision. Every different node will possess this real-time monitoring tool. The NFR tool monitors a particular system's properties, e.g., CPU utilization, energy consumption, the quality of the link between the edge devices and the cloud, and security. The application has other requirements in terms of latency and security, and real-time monitoring tool includes both. So whenever the system availability changes because of specific application necessities, the monitoring tool sends some alerts and recommendations to the COMPSs Scheduler.

Taking advantage of Figure 2.9, bellow are some examples of why and how NFR tool would act:

- if some QoS metric (CPU usage, deadline misses,...) is exceeding some threshold, in Node C, one or more of its tasks would be moved to another Node, for example, Node A or other with more available capacity;
- if some Node/Edge device turns down for some reason, COMPSs will transfer all tasks to a different Worker;
- if some Worker is up again, COMPSs will be notified that there are more available computing resources, and it may transfer some tasks again to this available Worker.

2.3 Research methodology

That said, the starting question that guided this dissertation was: "How to ensure that running applications meet non-functional requirements?". These applications may be running from edge computing to cloud computing. And real-time and energy are the non-functional requirements focused on this dissertation. Another question is, "How to improve the workload distribution to enhance some of the non-functional requirements?" For example, if we move a particular application, will there be energy savings? Or will there be a minimization to the percentage of deadline misses?

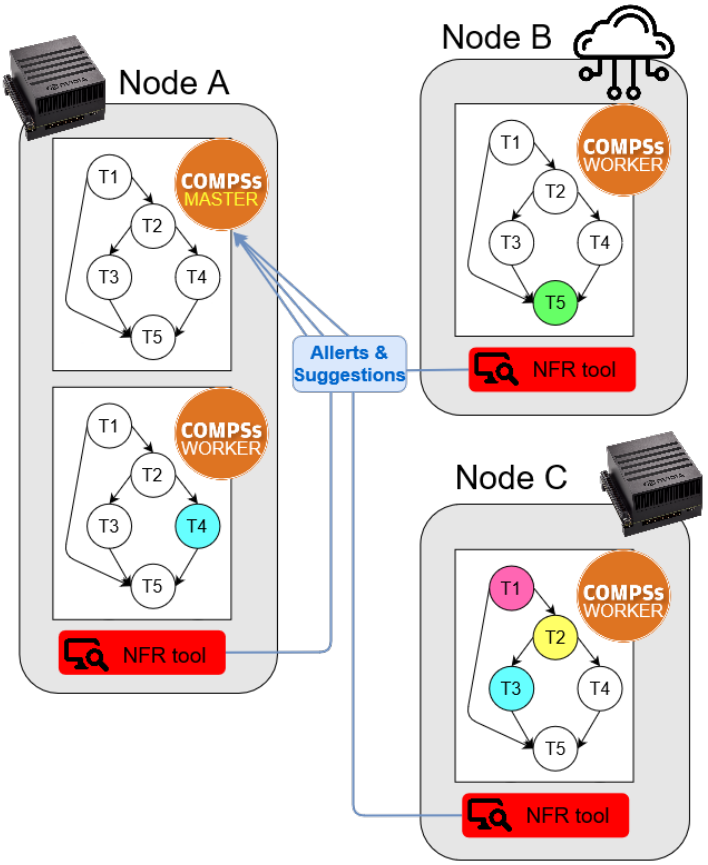

Figure 2.9: Example of the ELASTIC use case

The answer to these questions is to design, implement and experiment with diversified heuristics or algorithms that will ensure non-functional requirements in more unstable situations (with periods of high and low workloads) and improve performance in more stable scenarios.

Given the above and following different methods, we chose a set of relevant sources for the study. The pursued methods were:

- study research - to see approaches already tested;
- controlled experiments - in the sense that we can conclude which are the best metrics to evaluate to make the necessary decisions to balance the workload;
- case studies - attempt to have a real scenario of how the workload will be in a real scenario and how well the heuristic/algorithm will handle more unstable situations in terms of ensuring non-functional requirements.

The validation of the research involves:

- analysis, since it is almost certain we will simulate data (that is, try to vary the resource usage data as much as possible) and compare the various developed heuristics/algorithms;
- experiences, since there is the possibility of being able to validate the heuristics/algorithms in a real scenario (on journeys made by the tram);

- evaluation, since it includes feasibility studies, this dissertation's work possibly will contribute to a pilot project.

An important point to bear in mind is that this dissertation will focus on reactive and preventive modifications/adaptations of the workload distribution. However, the preventive ones are not within the scope of the ELASTIC project, hence the likelihood of having to simulate data.

Finally, the dissemination result in this document and its presentation could be useful for companies which are transitioning their services/applications to microservices or virtualized systems (e.g. Docker). According to a Docker survey from 2016 [49], 71% of the respondents are using or planning to use Docker to containerize a legacy app, 78% to build new microservices applications, and 44% of organizations are adopting microservices. And according to a recent StackOverflow survey from 2020 [50], Docker is the third most popular platform after Linux and Windows and is the first most wanted from developers who are not developing with it.

2.4 State of the Art

Section 2.3 presented an answer to two questions that guided this dissertation's development. The first two sections present an analysis of other important approaches and solutions that helped solve the problem. Section 2.4.1 focuses on resource management and ensure that running applications meet non-functional requirements. Section 2.4.2 targets means of prediction of resource usage for the sake of improving the workload distribution to enhance some of the non-functional requirements.

2.4.1 Resource Management

One of the analyzed documents that helped to understand the state-of-the-art regarding resource management was written by Cheol-Ho Hong and Blesson Varghese [51]. They reported an extensive survey paper with their research on resource management techniques in fog/edge computing and identified and classified the key contributions comprising architectures, infrastructure, and algorithms. A more detailed analysis focused on the algorithms since the architecture is delineated and another component manages the infrastructure (which is not part of this dissertation's scope).

The authors designated four algorithms types:

> *"a) discovery - identifying edge resources within the network that can be used for distributed computation, b) benchmarking - capturing the performance of resources for decision-making to maximize the performance of deployments, c) load-balancing - distributing workloads across resources based on different criteria such as priorities, fairness, and so on, and d) placement - identifying resources appropriate for deploying a workload" [51, p. 24]*

The last three types are interesting and desirable on the NFR Tool. The type of discovery algorithm is supported by the infrastructure component, Hybrid Fog Computing Platform, as explained in Section 2.2.

The benchmarking type would be an important functionality in the NFR Tool. Here, it raises a key challenge of obtaining metrics in near real-time since a global edge benchmarking that measures workloads' diversity is not yet available [51]. If this type of algorithm is applied, the

NFR tool will consider the following concerns. As Edge nodes are resource-constrained, it is needed to evaluate whether adding this algorithm will overload the Node. The edge should also not be benchmarked alone; it should include cloud and edge computing to maximize the applications' overall performance. Although most of the options presented refer to the cloud application like CloudSim [52], there are already tools for edge application like EdgeCloudSim [53] or iFogSim [54].

From load-balancing type, algorithms should distribute a set of tasks across available resources to increase the overall performance or to reach a certain goal, e.g., do not leave a server idle while there are tasks to be performed. One of the simplest load-balancing algorithms is Round-robin. In a scenario where multiple requests are received from clients, these requests are sent sequentially and circularly to the servers, i.e., imagining there are a pull of requests (R) and 3 servers (S); the behaviour will be R1 goes to S1, P2 to S2, P3 to S3, P4 to S1, P5 to S2 and so on [55]. Another technique of load-balancing algorithms is Software Defined Networking (SDN)[6]. In [56], the authors integrate fog computing and SDN to decrease the latency (achieved when using cloud computing) in a Internet of Vehicles (IoV) environment. They proposed an architecture with a SDN controller between the Cloud and Fog computing layers. The controller has a centralized control of the vehicle network and helps to get the information needed for load balancing. To implement the load balancing, they applied Particle Swarm Optimization - Constrained Optimization.

Last but not least, the placement type which is the main goal of this dissertation. Through the NFR Tool's monitoring, it is possible to conclude which is the best Node to allocate Workers, specially in situations with overloaded Nodes. Placement algorithms evaluate the availability of the resources and then place the tasks on satisfactory resources. These algorithms could be applied to all three layers of the compute continuum, but we will focus on fog and edge since there are large computing capabilities in the cloud. The typical challenges in fog or edge devices are high dynamicity, reliability, security and heterogeneity, which increase the complexity for this type of algorithm. Below are four examples to illustrate the type of placement algorithm.

Considering dynamic environments, Shiqiang Wang et al. [57] tried to create an optimal service placement to minimize the average cost over the entire duration that the service runs. To measure the cost, they added up the cost of data transmission/processing with the cost of service migration from actual to the new location. Then, they proposed a method that predicts (within a window of time) the future costs of service hosting and migration (when it exists). That prediction figured out the optimal placement and optimal window of time looking into the future.

Focusing more on computational resources, Mohit Taneja et al. [58] used a Fog-Cloud paradigm to tackle two problems, diminish application latency and provide efficient resources utilization. The authors proposed the ModuleMapping algorithm that, firstly sorts the available resources capacity in ascending order, from the Fog towards the Cloud; secondly, it obtains three requirements (CPU, RAM, and bandwidth) of each application module/task; and lastly, place each module/task on a node that has sufficient resources to receive it. As long as Fog nodes have available resources, no application models/tasks are placed in the Cloud.

In [59], Ehsan Arianyan et al. focused on reducing the enormous energy consumption from cloud data centers through consolidation in virtualized cloud data centers. Although the

[6] "SDN provides flexible centralized control and global knowledge to the network"[56]

authors focused only on Cloud, the paper presents interesting heuristics and approaches to be applied in this dissertation. They give a holistic cloud resource management and heuristics based on multi-criteria decision-making method divided into four parts[7]:

> *"(1) determining when a host is considered as being overloaded; (2) determining when a host is considered as being underloaded; (3) selection of VMs that should be migrated from an overloaded host; and (4) finding a new placement of the VMs selected for migration from the overloaded and underloaded hosts." [59, p. 2]*

They also mentioned that "the basic reason of energy waste in data centers' infrastructure is underutilization" [59, p. 2]. Since energy-efficient is one of the non-functional requirements to fulfill, identifying both states of a Node (overloaded and underloaded) will be considered. Regarding the multi-criteria decision-making method, the considered criteria are[7]:

> *"(1) the selected PM has the least power increase, (2) the selected PM has the most available resources, (3) the selected PM has the least number of VMs, (4) VMs on the selected PM have the least resource correlation with the VM to be allocated, and (5) the migration delay of the VM to be allocated to the selected PM is the least." [59, p. 7]*

In [60], Roman Sosa et al. carry another solution, offloading the applications tasks from edge to cloud resources. They demonstrated how we could offload applications from edge to cloud regardless of the platform on which the applications run. Their goal is to "support the rapid-prototyping of such applications [applications requiring heavy computation] (...) and at the same time make sure that solutions are robust to variations in application load and operation conditions" [60, p. 150]. Examples of addressed use cases to offload the applications are sudden load peaks or overload situations, which corresponds to some of the circumstances that the NFR Tool should deal with. The monitored metrics to decide whether to offload or not are memory utilization, instantaneous CPU utilization, CPU temperature, and the number of computation tasks currently in local execution. In their usage example, job metrics such as completion time and success ratio are measured in parallel. Without offloading, we can observe that when the gateway heats up, reaching limit temperatures, gradually, the CPU performance decreases, and so the job duration increases. While with offloading, some of the its used strategies (they presented four, consisted in limiting i) the number of jobs running locally in parallel; ii) CPU utilization; iii) memory utilization; iv) CPU temperature), had better results. The conclusion is that it is needed a comprehensive study of multi-criteria decisions because of the results that were reached were interesting.

2.4.2 Resource Prediction

This work will support reactive and preventive adaptations, but the latter is the main challenge in this dissertation, that is to turn reactive decisions into proactive ones by forecasting resource usage.

One of the advantage in implementing this idea are the cost reduction to pay for the used resources. In [61], the authors show that by forecasting the resources, they can obtain greater certainty of the amount of resources they intend to reserve (which is not necessary when using on-demand plans) and thus save on the final cost to pay. Another advantage is

[7] For this example, and for the sake of comparison, host, and PM are equivalent to a Node, and VM to Worker

efficiency improvement and flexible resource management. Flexibility provides the ability of changing computing resources according to the demands of the running applications. The idea is to eliminate the allocation of fewer computing resources than it actually needs (to not compromise the running application) and the same is true the other way around, that is, allocate too many resources, which results in increasing resource idle rate and decreasing performance. Countering these two situations will make the system more efficient and therefore, reduce costs [62].

In [63], Vivek Kumar Singh et al. focused on estimating the energy consumption of executing software processes. Their approach consisted of capturing the OS-level energy consumption metric using a watt-meter and the utilization of process-level CPU, disk, network, and memory. After collecting the previous metrics several times varying workloads, they build a model using Support Vector Machine (SVM)-based regression modeling. After that, it is possible to estimate the process-level energy consumption with a periodicity of 1 second. In this way, it is unnecessary to use additional devices (in this case, a watt-meter). Their results showed that "90% of the data points are above 95% accuracy and only about 10% of the estimated energy value data points had more than 10% error" [63, p. 99].

In [64], Florian Schmidt et al. integrated the monitoring of resource usage statistics with kernel and library calls of applications. Then, they trained recurrent neural networks, more specifically Long Short-Term Memory (LSTM), and built a Machine Learning model of a system to forecast resource usage or detect anomalies. They considered the resource usage statistics, like CPU and memory consumption, insufficient to understand the application behavior; for this reason, they created a "dictionary" of system calls, which gives them a deeper understanding of the process behavior. So, they collected resource usage (they only refer to CPU and memory metrics) and system calls from a scientific toolchain (which was consisted of bash and python scripts with heavy I/O and CPU phases interleaved). Next, they incorporated the system calls through the word2vec skip-gram model and trained LSTM with the collected data. They focused on minimizing the Root Mean Square Error (RMSE) of the CPU usage and varying two attributes, the number of seconds to predict into the future (between 1 and 15 seconds), and how much history to take into account for the prediction (1 or 10 seconds). They conclude that: (1) in case of considering 1 second of history, the forecast error tends to increase, varying between 0.125 to 0.2, approximately; (2) in case of considering 10 seconds of history, the forecast error varies between 0.075 and 0.1, approximately;

In [65], Gurleen Kaur et al. proposed an Intelligent Regressive Ensemble Approach for Predicting (REAP) CPU usage to achieve high performance via the integration of two techniques, feature selection, and resource usage prediction. In their solution, one of the drawbacks was tackled by using feature selection. They used Genetic Algorithm[8] to eliminate irrelevant and redundant features, consequently reducing the size of the dataset.

The previous presented works are some examples of how a data-driven solution based on machine learning techniques could help to improve the resource management where past data of similar situations could be opportunistically used.

[8] Genetic Algorithm is an algorithm which essentially selects the best set of inputs/variables to produce the best output.

Chapter 3

Value Analysis

Nick Rick and Matthias Holweg defined Value Analysis as "a process of systematic review that is applied to existing product designs in order to compare the function of the product required by a customer to meet their requirements at the lowest cost consistent with the specified performance and reliability needed" [66, p. 2]. In a simpler definition, value analysis is a process that follows an organized approach to increase the value of the products or services to be or already developed. The primary objective is to study the value analysis of the work presented in this dissertation. Therefore, this chapter presents the opportunity identification and analysis, the value creation and value proposition.

3.1 New Concept Development (NCD)

The New Concept Development (NCD) model was used as the basis for the development of the value analysis chapter. NCD model helps to understand the Front End of Innovation[1] by giving a common language and terminology.

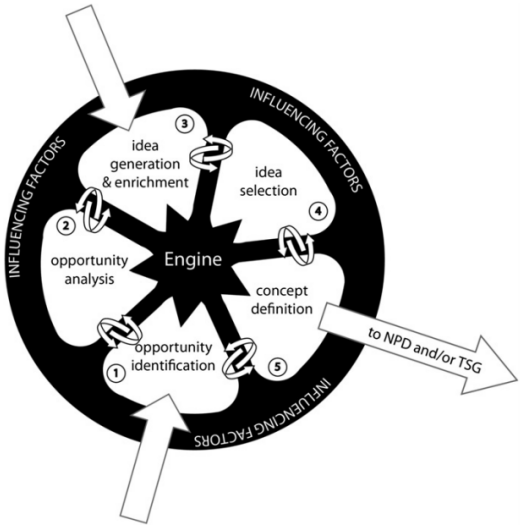

Figure 3.1: New Concept Development Model
Source [67]

As shown in Figure 3.1, NCD is composed of three distinct areas: the engine, the wheel, and the rim. Assenting in this dissertation's scope, only two topics will be covered from the

[1] Front End Innovation, or Fuzzy Front End, is the first area of the innovation process [67]

wheel area, which are opportunity identification and opportunity analysis. According to [68], the principal methods for both opportunity identification and analysis are:

- Roadmapping: captures the business graphically in a way that enhances communication and sharing of wisdom, resources and skills of the entire project team. A person outside of the project can understand the complexity of the business;
- Technology trend analysis and forecasting: analysis of relevant and competitive technological trends for the project. The following questions must be taken into account: "What will be done / where are we going"; "What already exists"; and "How the project will be different / what value will it bring";
- Customer trend analysis: research and analysis of the trends of customers in general;
- Competitive intelligence analysis: allows analyzing and communicating information about competitive trends that occur outside the organization;
- Market research: deeper research, for example, in scientific articles that present analyzes/services/applications already developed and understand how, when and where they were made;
- Scenario planning: developing scenarios to visualize the future, in order to simulate decisions.

On opportunity analysis specifically, there are more methods/techniques that can be used:

- Strategic framing: evaluation of how the opportunity suits the company's market and technology strengths, gaps, and threats (e.g. elaboration of SWOT analysis), and it works as the final framework for the opportunity identification;
- Market segment assessment: evaluation detailed of the market segment (if it is appropriate to the opportunity, if it a business-to-business, a business-to-consumer, etc.);
- Competitor analysis: identifies the competitors in the target market segment, determines the type of products (whether being software, application, algorithms), and its characteristics for the sake of comparison and evaluation of the competitors' strategies and capabilities;
- Customer assessment: identifies customer needs that are not fully met in current products, e.g. verifies and aligns the customer requirements.

3.1.1 Opportunity identification

In the scope of this dissertation, the opportunity identified is the integration of a non-functional requirements tool in a software framework which is suitable to develop, deploy and execute applications based on complex data-analysis. The software architecture incorporates the elasticity concept across the compute continuum in a fog computing environment, also including powerful hardware at the edge computing. Some questions that arose and led to the identification of this problem/opportunity were how to ensure that running applications meet non-functional requirements? How to improve the workload distribution in order to improve some of the non-functional requirements? For example, if we change the location of a certain application, will there be an energy saving? Will there be a decrease in deadline misses? After a team brainstorming, a tool that monitors and analyses software components running whilst dynamically (online) fulfill the non-functional requirements (real-time, energy efficiency, communication quality and security) is needed. One of the methods that was

utilized in the implementation of this tool was the market research in order to enhance the capabilities of the tool's initial purpose, e.g. initially the idea was to monitor the resource usage exclusively; however, after further literature analysis, it was conclude that this same tool could provide recommendations to the scheduler.

3.1.2 Opportunity analysis

The opportunity analysis consists of a) analysing and complementing the information obtained in the process of opportunity identification, b) analysing the available technologies as well as the market demand for the product to be developed/improved, c) as well as ensuring that the product to be developed is attractive to the target audience (it is worth pursuing). Finally, this process can be formal or iterative (as new features and constraints are identified in the concept definition phase).

In short, the base opportunity for the NFR Tool is the insurance of running applications. NFR Tool's goal is to control the satisfaction of the non-functional requirements of a system and dynamically adapt the configuration or deployment of that service when those requirements are not being met. An important part of this dissertation's contribution is the introduction of the system's behaviour prediction, making necessary reconfiguration or redeployments in the system to turn it less reactive and more dynamic (acting in advance). To implement the development of predictive models, it is necessary the application of data mining techniques. And this is the main contribution that, to date, has not been found in projects of the size and applicability of ELASTIC project. Chapter Value Analysis presented several attempts to predict resources in order to improve resource management. However, all the experiments identified in the literature review were used in simulation environments only.

3.1.3 Function Analysis System Technique

Function Analysis System Technique (FAST) is part of the functional analysis of the value analysis process. It is a technique that presents logical relationships between the functions of the solution (project, product, etc.) based on the questions "How", "Why", and possible, "When". The legend of Figure 3.2, helps to understand how to make these questions. From left to right, we ask each function (represented in the figure by a yellow rectangle) "How do you achieve this function?"; from right to left, "Why do you do this function?"; and from top to bottom, "When do you do this function, what other functions must you do?"

The FAST diagram advantage is the support to determine the project's scope coordinated by functions and its logical relationships. As it could be seen in Figure 3.2, the functions between the vertical dashed lines represents the scope of the project and the value/opportunity analysis.

3.2 Value creation

Value for the customer

Value for the Customer can be defined through two other concepts, use value and esteem value. Use value: It is an application that manages the non-functional resources of applications efficiently and intelligently. Although the Operating systems already do it in several ways, depending on the scheduling algorithm you are using (Round-Robin, Shortest-Deadline First), this product will be very useful in compute continuum architectures, that is, that can

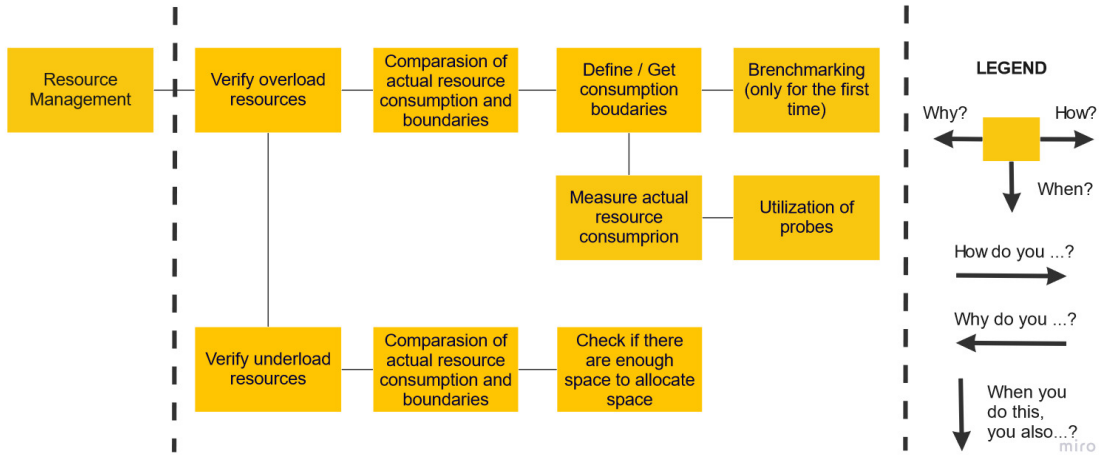

Figure 3.2: FAST diagram

run on edge, fog and cloud computing. Esteem value: From an environmental perspective, this application can reduce the overall energy consumption of the system since energy is one of the non-functional requirements to be monitored, so the possibility of reducing computing power without compromising the system's effectiveness, can be taken account for greater energy efficiency.

Perceived value

Perceived value is, in short, the value that the customer accepts as fair for a service or a product. This value usually differs from the perspective of the producer. For example, while the costumer is more concerned with quality, the producer maybe more concerned with reliability and at the same be cost-efficient.

Benefits

The confidence assurance on the transportation system can be perceived by the costumer as a benefit. The NFR Tool aids by offering better QoS for the running applications (resolving violations in a reactive and predictive way) mainly enhanced by the resources usage forecast. Another benefit is the reduction of maintenance costs, in which the NFR Tool also monitors and manages the use case that addresses this benefit (predictive maintenance of rail track).

Sacrificial

The positioning and obstacle detection use case can help understand how the NFR Tool will help mitigate sacrifices that can be perceived by customers, like helping minimizing injuries to people. Since the NFR Tool knows the resource availability and can manage the positioning and obstacle detection application more precisely (e.g., avoiding application delays or reducing unexpected cyberattacks), this software is unlikely to have flaws that could lead to an accident, such as not detecting a person on the tram line. Another sacrifice than can be perceived by the costumer may be the lack of profitability (Return on Investment (ROI)). At the very least, the NFR Tool will help with energy efficiency at the software level, which probably can reduce some energy costs.

3.3 Value Proposition

Based on the Business Model Canvas, the Value Proposition Canvas was designed to answer the questions "What?" and "To whom?" from the panels Value Proposition and Customer Segments, respectively. This canvas developed by Alexander Osterwalder is shown in Figure 3.3.

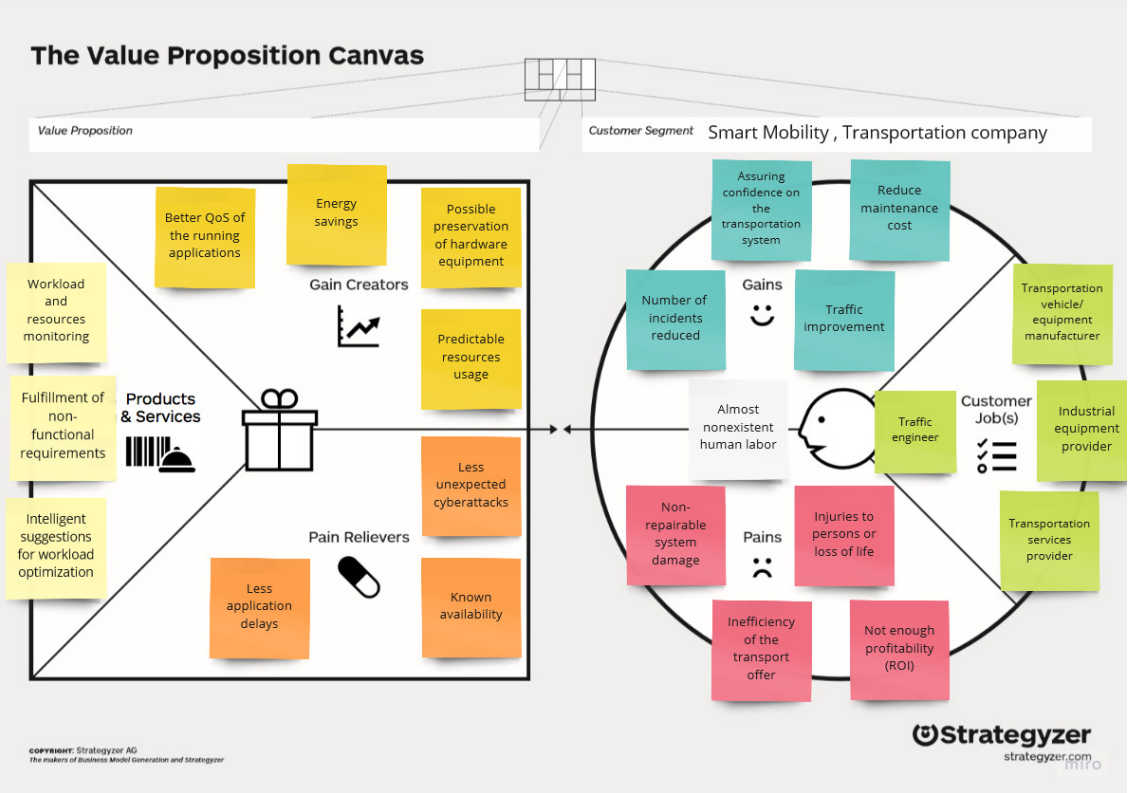

Figure 3.3: Value Proposition Canvas

Some insights for answering the next few questions were acquired from the Value Proposition presented. However, more detail about the ideas presented in the above picture are in the answers of the following questions.

1. What is your product? The product is a tool with the primary objective of fulfilling the non-functional requirements. Monitoring and analysing the resources, and the workload of the running applications is essential to accomplish the first point. In addition, the analysis phase is intelligent, that is, it will use heuristics and algorithms to provide a set of better solutions for the new application configuration, in order to optimize also the workload of applications.

2. Who is your target customer? Some of the customer segments are smart domains such as smart cities and smart mobility, more specifically, transportation companies. Railwails, automotives, avionics and the critical industrial domain are also possible target costumers as well as cloud and data service providers; public sector (city councils, governments, institutions,etc.) with interest on city administration agencies; data analytics and fog computing application developers; public-private partnership (e.g. private operator, public customer)

3. For whom you provide value? We intend to use the ELASTIC project results in cyber-physical systems in general, with particular focus in rail (in collaboration with advisory board member) and automotive (existent collaborations).
4. What value you provide? The principal value provided is the integration of non-functional properties (in particular real-time, energy and predictability) in fog computing architectures including intelligent suggestions for the scheduler.
5. Why is your product unique? To the best of our knowledge, ELASTIC software architecture framework, of which the NFR Tool is a part, is an innovative opportunity since the software framework will develop, deploy and execute applications based on complex data-analytics workflows (including AI and big-data methods) on a fog computing environment and incorporating non-functional requirements inherited from the cyber-physical interactions from smart systems.

3.4 Analytic Hierarchy Process

Phase 1 - Construction of a hierarchical decision making tree

Define the problem and structure it in a hierarchical diagram. This phase consists in a the decomposition of the problem/decision in a hierarchy composed, at least, of a goal, criteria and alternatives.

Phase 2 - Comparison of the alternatives and criteria

The second phase consists in establishing the priorities between the elements for each level of the hierarchy by means of a - The first point that should be considered is the determination of a scale of values for the comparison, that must not exceed a total of nine factors, for the point of maintaining a consistent matrix. - This way, Saaty defined a Fundamental Scale.

Phase 3 - Priority related to each criteria:

To obtain the priority related to each criteria it is needed: a) Priority of the alternatives regarding each criteria; Weight of each criteria in relation with the final goal. b) Normalizing the values of the comparision matrix (matrix A) - the objective of this is to equalize every criteria with the same unit, for this each value on the matrix is divided by the total of its rescpective colunm. c) Obtaining the priority vector - the goal here is to identify the order of importance of each criteria, for this the avarage of the arthmatic values is calculated for each of the lines of the normalized matrix that was obtained in the previous item.

Phase 4 - Evaluate the consistency of the relative priorities

- The next step is to calculate the Consistency Ratio (CR) to measure how consistent the judgments were in relation to the big samples of the completely random judges. - The evaluations of the AHP method are based on the assumption that the decider is reational, this means, if A is prefered to B and B is prefered to C, then A is prefered to C. -If the Cr is superior to 0.1 the judgments are unreliable because they are too close for the comfort of randomness, in this case the values that were obtained did not present consistent values. -To calculate the CR first you need to get the value for λ_{max} which represents the greatest

own value of the matrix A, that can be obtained through the following equation:

$$Ax = \lambda_{max}x$$

- Once λ_{max} has been calculated the Consistency Index (CI) should be calculated in order to be able to calculate the CR.

$$CI = \frac{\lambda_{max} - n}{n - 1}$$

-The CR can be obtained through the following formula:

$$CR = \frac{CI}{RI}$$

in which the random consistency index (RI) referring to a large number of peer-to-peer comparisons. This is a random index calculated for square n matrices of order n by the Oak Ridge National Laboratory, in the USA. The following table defines the RI values according to the number of criteria.

Phase 5 - Construction of the parity comparison matrix for each criterion, considering each of the selected alternatives

All the procedures for the construction of the comparison matrix and for the determination of the relative priority of each criterion must be done again, observing now the relative importance of each one of the alternatives that compose the hierarchical structure of the problem in question.

Phase 6 - Obtain composite priority for alternatives

In this last step, we obtain the priorities composed of the alternatives, multiplying the previous values and those of the relative priorities, obtained at the beginning of the method.

Phase 7 - Choice of alternative.

Chapter 4

Design

At the time the dissertation was developed, the integration of the NFR Tool with other components, specifically with the COMPSs orchestrator, was not ready. And, initially, the dissertation's goal was to build the NFR Tool acting reactively only. Still, because of the delayed integration and the fact that the running tool will be inserted in critical systems, the challenge of acting preventively/proactively appeared pleasing to be tackled by this dissertation. So, it was decided to redesign the NFR Tool's component to incorporate a global and centralized tool, the one with the machine learning application for the resources' forecast. Figure 4.1 illustrates Global Resource Manager (GRM) and some of the ELASTIC SA components already presented in Section 2.2, COMPSs orchestrator, dataClay, NFR Tool and NuvlaBox.

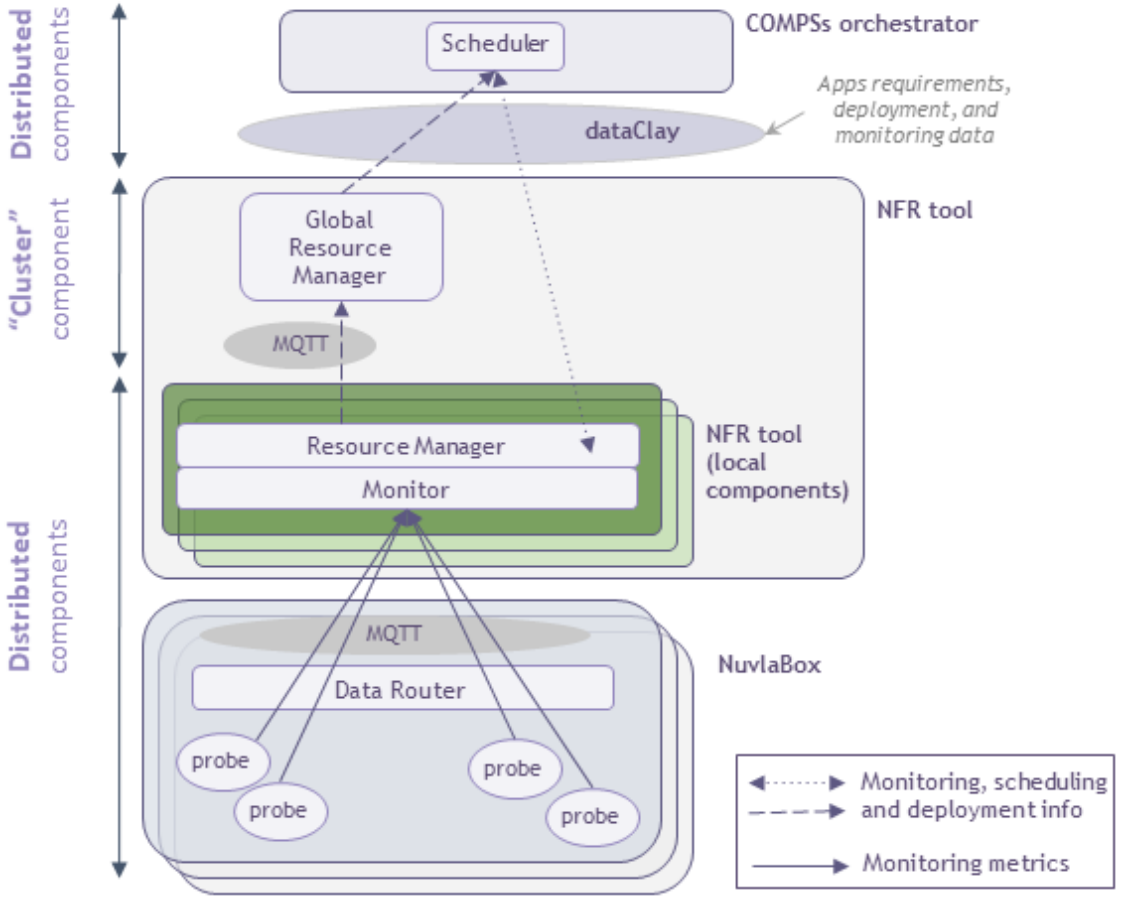

Figure 4.1: ELASTIC SA design

Looking at Figure 4.1 from bottom to top, NuvlaBox is open-source software under Apache license 2.0, which provides several components. But for this dissertation's context, the focus is on Data Router (or under the terms of the NuvlaBox's documentation, Data Gateway). The Data Router abstracts the need to implement an interface between the edge devices and user applications since it is equipped with a communication layer that contains several mechanisms (e.g., Message Queuing Telemetry Transport broker). Via the communication layer, the edge devices can relay raw sensor data, on-demand, to any existing user applications. So, by installing the NuvlaBox in each ELASTIC system Node, the Node resources consumption metrics are captured by the probes and published to the Data Router communication mechanism, MQTT broker in this project. Further up, NFR Tool's Monitor subscribes to a specific topic to obtain the actual resources consumption metrics and monitoring them. Then, NFR Tool's component has several instances (NFR Tool local) installed in every Node. NFR Tool local components include two sub-components: 1) the Monitor, which controls the non-functional requirements and 2) the Resource Manager responsible for informing GRM in case of any NFR violation detected by the Monitor.

The GRM is part of the NFR Tool concept, but it is not distributed by several Nodes., hence the implementation of a clusterized approach. GRM needs a global view of the ELASTIC system, so a completely centralized approach would not be the better approach since GRM must have access to the complete information of the ELASTIC system. A mix of the centralized and distributed approaches seemed the best option since the control of the tram line is separated, i.e., by regions or cities. Finally, the idea of GRM is to influence COMPSs in the tasks' distribution by writing in dataClay.

The dataClay offers the resources' information, e.g., the available CPUs per Worker used by a particular COMPSs application. Then, the COMPSs orchestrator/scheduler will allocate tasks based on its scheduling policy but within the available resources written by GRM. For example, suppose GRM writes in dataClay that COMPSs can use 4 CPUs at Node A. In that case, COMPSs may use up to 4 CPUs in Node A. So, writing in dataClay is almost like a trigger for the COMPSs to reschedule the ELASTIC system tasks, according to the new information given by the GRM. Lastly, COMPSs orchestrator and dataClay are components naturally distributed within the several Nodes available in the ELASTIC system.

Finally, the GRM is outside of the ELASTIC project's goals; initially, the cluster component with a global and holistic view of the ELASTIC system did not exist. Although this addition has been approved, the development of a disaster recovery plan for GRM is outside the dissertation's scope as well as the scheduler/orchestrator development.

So, to summarize, what is in the context of this dissertation?

- Elaboration of distributed tools (NFRs) that monitor locally the best metrics to guarantee non-functional requirements, specifically real-time and energy-efficiency;
- Development of a centralized tool (GRM) that will help the scheduler to configure the distributed applications following reactive and preventive approaches;

And what is not in the dissertation's context:

- Offline analysis that has the purpose of gaining more valuable insights through the applications based on historical data;
- Initial deployment of the Workers in the ELASTIC system;
- Development of a disaster recovery plan for NFR Tools and GRM.

Figure 4.2 shows the Global Resource Manager (GRM) and some of the ELASTIC SA components already presented in Section 2.2, more precisely, the use-case presented in Figure 2.9 with the addition of GRM.

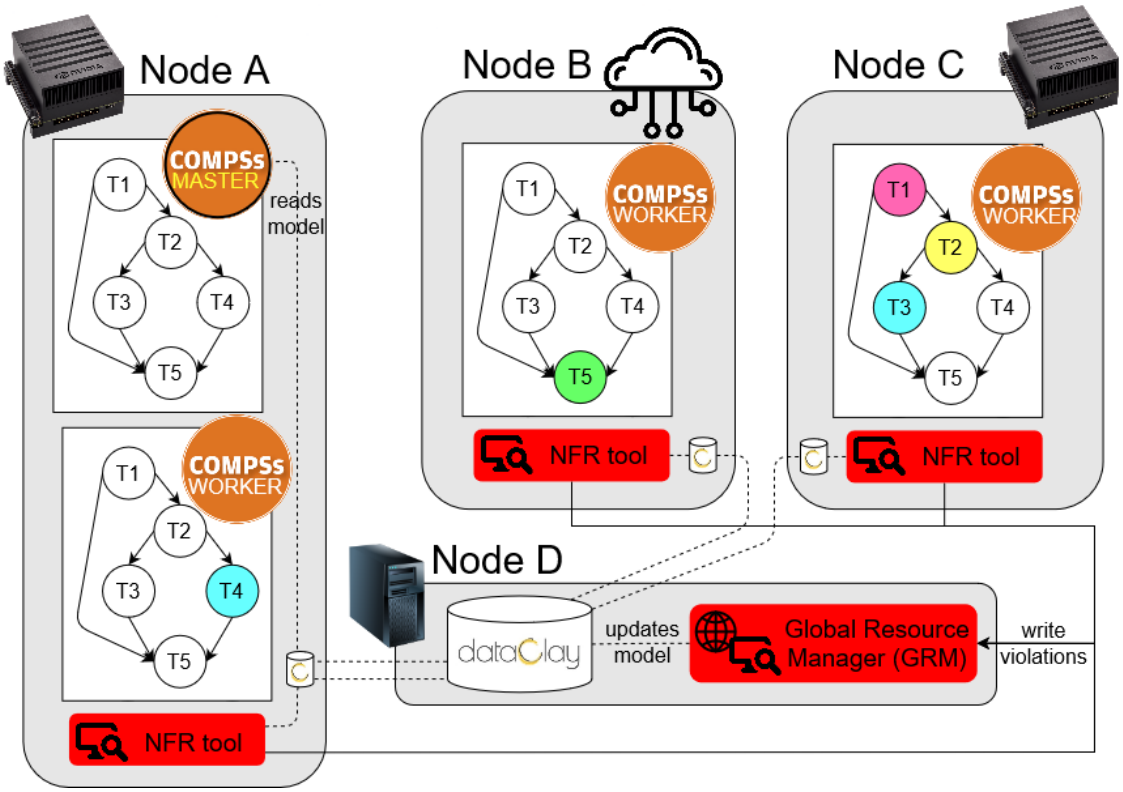

Figure 4.2: Example of the ELASTIC use case with the GRM

Instead of the NFR Tools sending alerts and recommendations to the COMPSs Scheduler, they send to GRM, then the GRM with a holistic view updates the applications' configurations and, finally, the Master redeploys the Workers.

As previously said, GRM will behave in two ways, reactive and proactive. In both ways, GRM must be able to trigger configuration changes when QoS violations are detected.

The first option analysed was Prometheus. Prometheus, similar to Zabbix[1] or Nagios[2], is an open source monitoring tool that offers the necessary features for the NFR tool, such as monitoring metrics and alert notifications on external devices about preprogrammed conditions via Prometheus' alerting rules. In addition, Prometheus collects and stores the metrics values as time series data (with the timestamp on which they were recorded), which seemed an advantageous feature for resource forecasting [69]. Last but not least, Prometheus provides dashboards (usually helpful for production usage, like Grafana[3]) and the Node Exporter that monitors machines; it exposes machine-level metrics on Unix systems such as CPU usage, memory, disk utilization, and network bandwidth. In this point, the security dimension was missing, anyway, some tests were carried out with simulated workloads. However, for the energy dimension, no good metrics were available. Prometheus Pushgateway could be

[1] Outside the scope of the thesis, visit `https://www.zabbix.com/features`

[2] Outside the scope of the thesis, visit `https://www.nagios.org/about/overview/`

[3] Example of a preconfigured dashboard for Prometheus Node Exporter `https://grafana.com/grafana/dashboards/13978?pg=dashboards&plcmt=featured-sub1`

the solution to this problem, as well as for the security dimension. Pushgateway allows the user to push custom metrics to an intermediary job and allow Prometheus to scrape and consume those metrics. But Prometheus Pushgateway is intended for short-lived service-level batch jobs or metrics (which would not be the case) and given that ELASTIC's architecture must support soft real-time requirements, it was decided to try the perf tool.

Perf, originally Performance Counters for Linux (PCL), and also know as perf_events or perf tools, is one of the most popular performance profiling/analyzing tools in Linux [70]. It profiles systems (abstracting the CPU hardware) whether it be kernel or user-made code. Additionally, developers can write highly customised monitors (offers various event types) and profile performance counters from applications [71]. The Perf tool supports an extensive list of predefined events of different sources. Two examples of the types of events are Hardware Events which are basically CPU performance monitoring counters, like cpu-cycles, cache-misses or branch-misses; and Software events, which are low level events based on kernel counters, like CPU migrations or minor/major faults, among others [70].

The second idealised design, which includes the use of perf, is depicted in Figure 4.3.

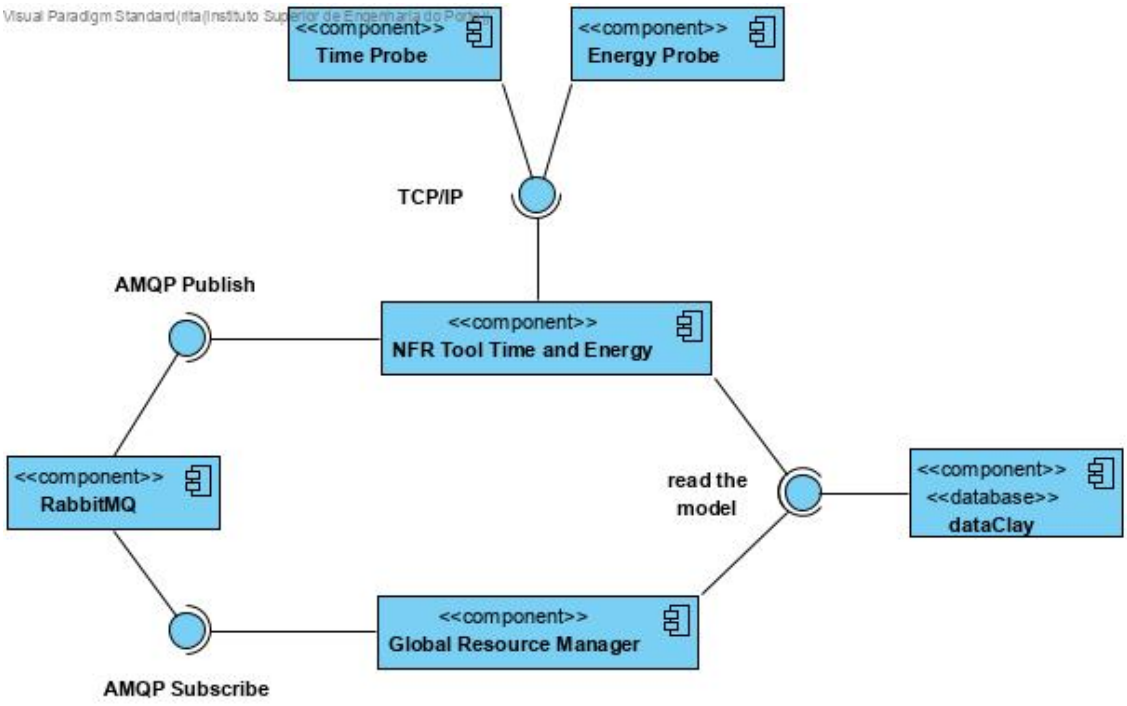

Figure 4.3: Component diagram with the probes

The main difference with the previous and following possibilities is the source of resource consumption collection and its communication protocol. In this case, the sources are the Time and Energy probes which are C scripts that execute the system call exposed by the kernel perf_event_open(2) and the communication protocol is the TCP/IP based network environment.

The following reasons lead to the creation of the third and final design. The requirements for the probes execution included certain configurations options compiled in the kernel, the kernel's NMI watchdog and kernel pointer hiding functionality must be disabled, and the perf_event_paranoid file must be configured to specify the access restrictions enforced by the kernel for measurements. Then, the NFR is integrated with other components that

also already offered resource consumption monitoring capabilities, namely Nuvla.io. Finally, the applications were Docker containers images, and in order not to overload it with the installation of more compilers and scripts for the possibility of using the probes previously described, it was decided to choose this last design presented in the Figure 4.4.

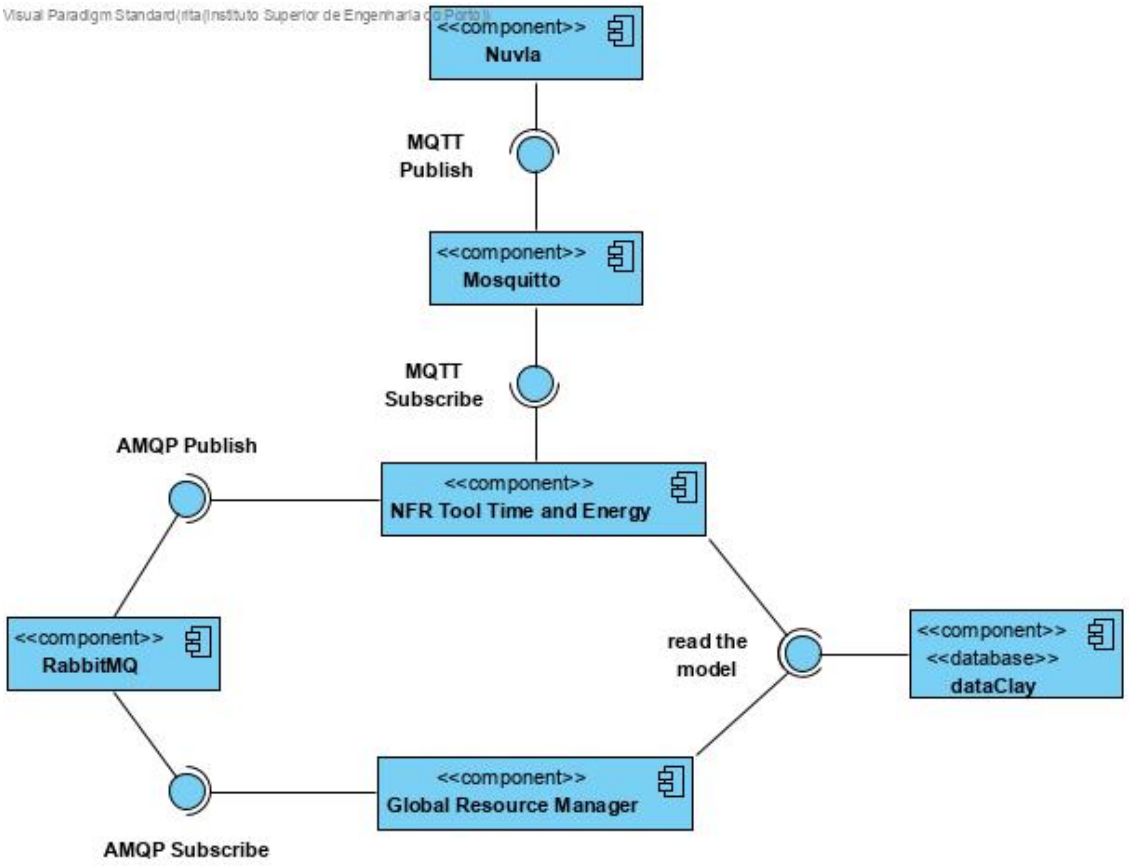

Figure 4.4: Final component diagram

The activity diagrams of the Figures 4.5 and 4.6 help to understand the flow of the last presented component diagram. A typical production flow would start with the execution of the NFR Tool in the several Edge devices (Nodes), and the GRM in a the central Node of its cluster, which can stays at Edge or Cloud.

Then, following the Figure 4.5, GRM searches the ELASTIC system already created and, in parallel, executes two tasks. Once it has found the ELASTIC system, it subscribes to a broker (RabbitMQ [4]) which will notify it of all violations occurring on all the Nodes of the ELASTIC system to which it belongs. Whenever it receives a notification, it adds that violation to its violation queue. And in another thread, it periodically checks its violation queue. If it has no violations to resolve, it sleeps for a certain period and checks again if it has no violations to resolve. If it has violations in the queue, it runs the implemented heuristics, either reactive or predictive, and depending on the result of the heuristics, it sets new configurations for the tasks of the applications running in the ELASTIC system.

[4]The main reasons for this choice was the possibility to use a combination of different messaging protocols, specifically, MQTT and AMQP, and from AMQP, it is possible to use almost any form of messaging pattern, including classic message queues, which is useful for GRM.

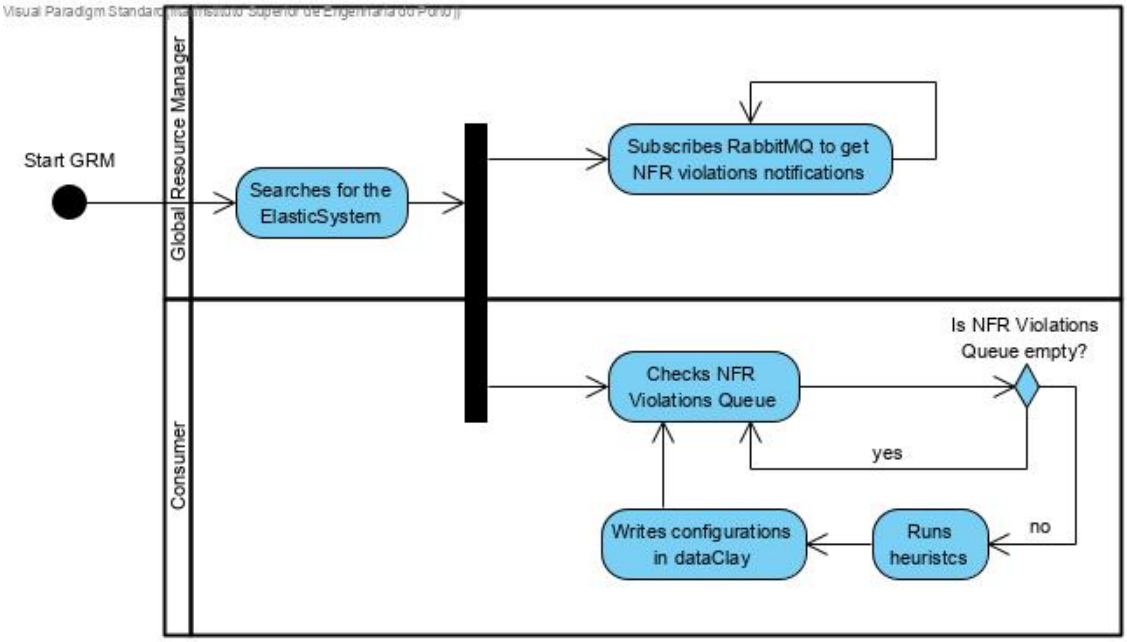

Figure 4.5: GRM's activity diagram

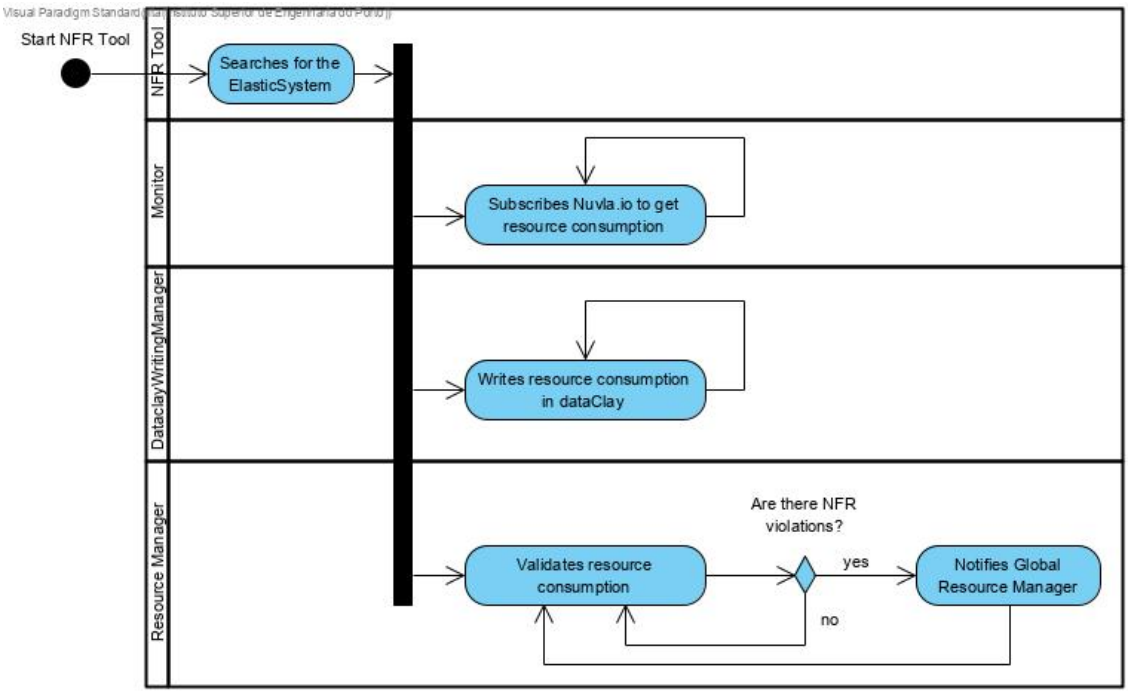

Figure 4.6: NFR's activity diagram

Following Figure 4.6, NFR also searches for an ELASTIC system through an alias, and once found, it subscribes certain topics to receive the resource consumption (metrics) of the different Nodes and Workers of the ELASTIC system. Then, another thread is responsible for periodically writing certain metrics to the dataClay for COMPSs and other components to also have access to them. Finally, in another thread, the received metric values are validated, for example, if the actual resource consumption is exceeding its defined threshold, then in case of committing any NFR violation, NFR publishes a message to the broker that

the GRM is subscribed.

The big question raised for the proactive behaviour was "How to prepare and develop models to make predictions". As the data was expected to be collected from a production environment, and given the works surveyed and presented in Section 2.4.2, the aim of the proactive GRM was to attempt to predict CPU consumption. In this way, the GRM configuration would not only act when something already exceeded the defined threshold. So, current and historical data would be needed to make predictions using the techniques of statistics, data mining, machine learning, and artificial intelligence. For example, in [63], the authors used Support Vector Machine (SVM)-based regression model, also known as Support Vector Regression (a regression algorithm) and in [64], Long Short-Term Memory (an improvement over Recurrent Neural Networks). In [44], Jason Brownlee refers that "Time series generally focus on the prediction of real values, called regression problems". Since the data contains the desirable output, supervised learning algorithms of the regression type should be used.

For the evaluation of the CPU workload prediction, the visualisation of the model predictions versus the actual data of the test set is a good starting point. However, sometimes it is not easy to understand if the model is a good match only by comparing the real and predicted values, and this is where performance measure indicators usually used for evaluating time series forecasts, like Mean Absolute Error (MAE), should be used.

The most basic measures of forecast performance are: a) Forecast Error, or Residual Forecast Error or Prediction's Residual Error, is the difference between the expected value with the predicted value;

$$forecast\_error = expected\_value - predicted\_value \tag{4.1}$$

and b) Mean Forecast Error (or Forecast Bias) is calculated as the average of the Forecast Error values.

$$mean\_forecast\_error = \overline{forecast\_error} \tag{4.2}$$

The reason for being called Forecast Bias is because if the error value is other than zero suggests a model's tendency to over/under forecast (negative/positive error). So, an unbiased model would have a forecast bias of zero, or near zero.

The measure with the same units as the expected outcomes is the Mean Absolute Error (MAE). It is calculated as the average of the forecast error values,

$$mean\_absolute\_error = \overline{|forecast\_error|} \tag{4.3}$$

where all of the forecast values are forced to be positive by the absolute function abs()[5]. The closer to zero, the less error there is in the model; a MAE of zero indicates no error.

Finally, the widely used error calculations that punish large errors, like Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). The MSE is the average of the squared Forecast Error values.

$$mean\_squared\_error = \overline{|forecast\_error|^2} \tag{4.4}$$

[5]Python function that return the absolute value of a number.

Squaring the Forecast Error values turns the outliers[6] highly sensitive, making errors larger/heavier. The RMSE, as the name suggests, is the square root of MSE.

$$root\_mean\_squared\_error = \sqrt{mean\_squared\_error} \tag{4.5}$$

The closer to zero, the more accurate the model is considered. As with the MAE and MSE, an RMSE of zero indicates no error.

In other words, MAE measures the average of the residuals[7] in the dataset, MSE measures the variance of the residuals and RMSE the standard deviation of residuals.

Additionally, accuracy classification score was also calculated, which is the fraction of predictions the model got right.

$$accuracy = \frac{number\_correct\_predictions}{total\_number\_predictions} \tag{4.6}$$

Accuracy is a commonly used metric to evaluate classification models, but since it is an easily understood metric and the values of the ground truth (correct) labels and the predicted labels returned by the model were known, it was also decided to calculate it.

Lastly, this chapter will end with the Requirements Engineering and Analysis made for this project.

4.1 Requirements Engineering

As referred previously, the dimensions focused in this dissertation are time and energy. The requirements defined until the present are listed bellow with its name, description and type (type of requirement according to the MoSCoW Model)[8]. Table 4.1 refers to Time's dimension requirements and Table 4.2 to Energy's dimension.

[6] Dots outside the whiskers (in a Box and Whisker Plots) or extents of the data.

[7] The difference between the actual value and the predicted value from the model.

[8] The MoSCoW Model is a prioritization technique with 4 different categories for managing requirements. The categories are divided into: 1) Must have (M), 2) Should have (S), 3) Could have (C) and 4) Will not have (W) which are self-explanatory.

Table 4.1: Time Requirements

Name	Description	Type
Real-time requirements	The ELASTIC architecture must support applications with hard, firm and soft real-time requirements.	M
Scope of real-time requirements	The ELASTIC architecture must support local and end-to-end real-time requirements.	M
Measurement of soft timing requirements	The ELASTIC architecture must support soft real-time requirements: • ratio of accepted failure • maximum admissible number of consecutive failures, and/or • average response time	M
Response time	The ELASTIC architecture must support end-to-end response time in the order of milliseconds (msec).	M
Real-time scheduling	The ELASTIC architecture must support schedulers with real--time constraints.	M
Mixed-criticality scheduling	The ELASTIC architecture should support schedulers with mixed-criticality constraints.	S
Multi-core scheduling	The ELASTIC architecture must support schedulers with multi-core constraints.	M
Worst-case execution time measurements	The ELASTIC architecture must support worst-case execution time (WCET) measurements.	M
Worst-case execution time analysis	The ELASTIC architecture should support worst-case execution time (WCET) analysis.	S
Execution time on multi-core / many-core platforms	The ELASTIC architecture must support applications executing in nodes with multi-core/many-core processors that require execution time measurements/analysis.	M
Multiple modes of operation	The ELASTIC architecture must support applications with multiple modes of operation. Modes of operation depend on: • internal conditions (e.g. system self checks, etc), • external conditions (e.g. position/speed of the train, track conditions, etc).	M
Allocation of components to nodes	The ELASTIC architecture must support components to be allocated to nodes: • dynamically at run-time, • statically at deployment time.	M

Table 4.2: Energy Requirements

Name	Description	Type
Energy monitoring on the edge	The ELASTIC architecture must support energy monitoring on edge nodes.	M
Energy monitoring on the cloud	The ELASTIC architecture should support energy monitoring on cloud nodes.	S
Energy-aware scheduling	The ELASTIC architecture should support scheduling techniques designed to ensure energy requirements.	S
Hardware speed scaling	The ELASTIC architecture should support speed scaling techniques as means to ensure energy requirements.	S
Multi-core energy mechanisms	The ELASTIC architecture should support on-line task-to-core allocation techniques that ensure energy requirements.	S
Multiple modes of operation	The ELASTIC architecture should support modes of operation designed for specific energy utilisation profiles. ELASTIC components switch between modes depending on predefined conditions.	S

Chapter 5

Proposed approach and experimentations

Commonly, schedulers/orchestrators follow scheduling policies. SCHED_DEADLINE, for example, is based on the Earliest Deadline First (EDF) and Constant Bandwidth Server (CBS) algorithms [72]. Kubernetes uses Completely Fair Scheduler (CFS) quota for CPU management [73]. COMPSs orchestrator provides schedulers for the most part based on First In First Out (FIFO) algorithm [74]. In these situations, the settings can be changed to improve system performance. Specifically on the ELASTIC project, some examples of the situations of why and how NFR tool would act are:

- if some QoS metric (CPU usage, deadline misses, etc.) is exceeding some threshold, one or more of its tasks would be moved to another Node with more available capacity;
- if some Node/Edge device turns down for some reason, COMPSs will transfer all tasks to a different Worker;
- if some Worker is up again, COMPSs will be notified that there are more available computing resources, and it may transfer some tasks again to this available Worker.

Kubernetes also orchestrates containers as the latter examples e.g., restricting resource consumption. However, there are a few examples where these decisions are not only reactive but also proactive/preventive. [75] is the most recent work that uses Machine Learning forecast methods to optimize the management of cloud based applications using Kubernetes. This dissertation presents the research/work done applied to COMPSs orchestrator and to be tested in a realistic scenario in the Florence tramway. The way to influence the scheduler/orchestrator is by changing the dataClay model parameters. As said previously, dataClay deals with various data structures (lists, graphs, dictionaries), as well as user defined objects (supports object-oriented programming language, Python and Java). Figure 5.1 shows part of the dataClay model currently used in the ELASTIC project[1]. Tables 5.1 and 5.2 present a description of each object/class presented in Figure 5.1 and a description of each attribute of each object/class. In the Appendix A, it is attached the figure with the complete dataClay model as well as the description of the remaining dataClay objects and their attributes.

So, changing the attributes of any of the objects will trigger COMPSs to reallocate tasks within these attributes. If GRM concludes that only 1 core can be used by a specific Node (i.e. updates numCores=1 in dataClay), then COMPSs may use up to 1 CPU in that Node.

[1] To understand how the dataClay influences the scheduler, it is enough to show the three objects presented in the Figure 5.1 (COMPSs Application, Worker and Node), since the other objects of the dataClay model do not have the attributes that directly influence the scheduler

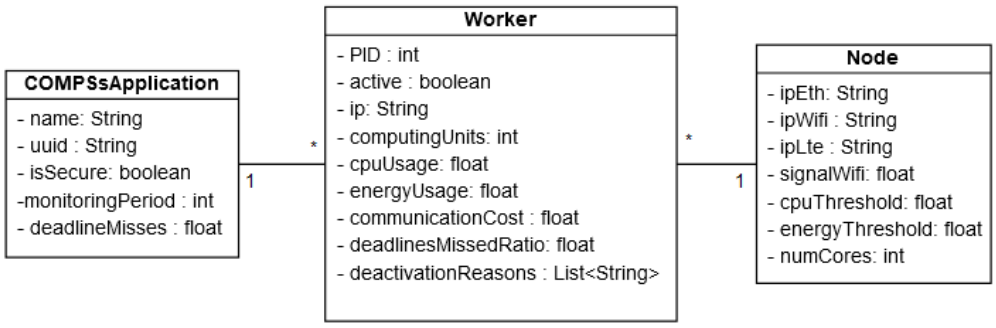

Figure 5.1: Partial dataClay model

Table 5.1: dataClay classes definitions

Class	Definition
COMPSs Application	represents a data analytics workflow distributed with COMPSs
Worker	represents each of the COMPSs worker processes of a given data analytics workflow and is executed in a given node
Node	represents a distributed computing node of the fog computing platform, with the following attributes

If GRM reduce the computing units to 5 in a specific Worker, the Worker will run at most in 5 computing units. Once the functioning of the system is presented, it is time to understand which heuristics are used for the reactive and proactive behavior of the system.

Table 5.3 shows the used devices for the experimental setup with some technical specifications of the machines. Briefly, three NVIDIA Jetson Modules and a laptop were used.

The NVIDIA Jetson devices will run NFR Tools and the latter the GRM. The stress-ng, a Docker image made from scratch with stress-ng tool statically linked [76] and another Docker image that offers three different levels of CPU consumption, were used to simulate the applications workload. As said previously, it was impossible to test the system in a production environment, and the COMPSs orchestrator/scheduler integration was not ready. So, in order to test the GRM reactive and proactive behaviors, some methods were added to the GRM and NFR Tool to simulate the COMPSs functionalities.

According to the Docker documentation [77], for the time dimension, there are two ways to set constraints in a given container's resource access to the host machine, also known as Node. The options are configuring the default CFS scheduler or the real-time scheduler. The first provides several runtime flags to configure the amount of access to CPU resources the container has, being the most relevant: a) "–cpus=<value>", that guarantees the container will use at most the specified CPUs every second, which is what it is needed, i.e., to reduce the computing units, and b) "–cpuset-cpus", specify which CPUs or cores the container can use. If the Worker is moved from one Node to another, the procedure is to stop the container in the overloaded Node and start in the recommended Node. Lastly, the real-time in ELASTIC is soft real-time and not hard real-time, so the provided features by the real-time scheduler (second option) were not considered.

Table 5.2: dataClay classes' attributes definitions

Class	**Attributes**	**Definition**
COMPSs Application	uuid	unique identifier for the COMPSs application since there may be application names that are the same
	name	application name
	isSecure	a flag that denotes whether the application should fulfil the security property (i.e., run only in secure nodes)
	infoNature	a string denoting the type of the application (i.e., video stream, etc.)
	monitoringPeriod	the reporting interval from the NFR tools to the dataClay model
	deadlinesMissed	the ratio of missed application deadlines
Worker	PID	PID of the process
	IP	the IP of the node where the worker is deployed, corresponding to the employed networking interface (i.e, WiFi, Ethernet of LTE)
	active	a flag denoting whether the worker should be active. The active flag is set to false by the NFR tool when a worker should not be used by COMPSs due to severe NFR violations (e.g., security property is not met)
	cpuUsage	current CPU usage of a worker
	energyUsage	current energy usage of a worker
	computingUnits	available computing units that can be used by the worker (must be a subset of the number of cores of the node)
	communicationCost	value of the estimated communication cost
	deadlinesMissed	ratio of missed deadlines for the specific worker
	deactivationReasons	a list with the NFR violations that led to the deactivation of the worker (i.e., violations of time, energy, communications, security)
Node	*ipWifi*	the IP of the WiFi networking interface (if available)
	ipEth	the IP of the Ethernet networking interface (if available)
	ipLTE	the IP of the LTE networking interface (if available)
	signalWiFi	the level of the WiFi signal
	cpuThreshold	the threshold for the time property (CPU utilization)
	energyThreshold	the threshold of the energy property
	numCores	number of CPU cores of the node

Table 5.3: Used devices for the experiments

ID	Machine's Name	CPU	RAM
xavier-rit	NVIDIA Jetson AGX Xavier	8-core ARM v8.2	32 GB
xavier-ric	NVIDIA Jetson AGX Xavier	8-core ARM v8.2	32 GB
nano09	NVIDIA Jetson Nano Developer Kit	Quad-core ARM A57	4 GB
cister	Lenovo ThinkPad T440s	Dual-core Intel(R) Core (TM) i5-4200U	8 GB

Regarding energy dimension, there are no straightforward options to limit the energy consumption of a specific process/Worker, only at a Node level. Two options that would lead to decreased power are limiting the CPU cores or turning off the fan. However, these examples have an impact at a Node level and not at a Worker level. So, when some specific Worker has a significant amount of energy, the possible action of GRM to eliminate that NFR Violation will be almost the same for the time dimension since CPU consumption is correlated with the energy spent. In addition, energy will be an evaluation factor for changes in the system in situations at the Node level. Therefore, one of the approaches is also to modify at a Node level, through the nvpmodel functionally, which will be later explained in more detail. Docker yields a command to update dynamically one or more containers, docker update. Windows containers do not support this command, but this is not a problem since the project is UNIX based.

So, to limit the resources that a Worker/container is consuming from their Docker host, and through the recommendation given by the GRM, some resource constraints will be updated. With a single command, you can place limits on a single container or many. However, GRM and NFR Tools will not support the execution of docker commands inside it, and two other mechanisms were tested, pseudo-files in sysfs (/sys/fs/cgroup) or cgroup fs (/cgroup) (system dependent) and Docker API. Through the pseudo-files, we cannot limit the number of CPUs, but we can modify cpuset.cpus file to use a specific set of CPUs or cores (comma-separated list i.e., 1,3 or hyphen-separated range i.e., 0-3) that a container can use.

```
echo 1,3 > /sys/fs/cgroup/cpuset/docker/CONTAINER_ID/cpuset.cpus
```

is an example that will obligate the container to use core 2, 3 and 4. Through Docker API [78], the same restriction remains; in the JSON body, the CpuCount parameter takes an integer, the number of usable CPUs, but it is available on Windows only. Besides specifying the set of CPUs or cores for setting CPU period constraints, one workaround is to use –cpu-period and –cpu-quota e.g., setting –cpu-period=50000 and –cpu-quota=25000 is the same as setting –cpus=0.5 (50% CPU). If there is 1 CPU, the container can get 50% CPU worth of run-time every 50ms. It was also implemented a way to affect the energy consumption at a Node level. The solution was not straightforward since the NFR Tool is a Docker container, and when it is needed to change anything in the host machine from a container, it usually leads to security risks. The final implementation to simulate the scheduler in these terms was to use inotify cron daemon (incrond). In the edge devices, incrond was installed, the user of the machine/Node was added in /etc/incron.allow, the incrontab -e was executed to create incron jobs, and the following job was added with the format <path> <mask> <command>, /opt/nfr/script.sh IN_MODIFY /opt/nfr/script.sh. So, every time the script is modified, the script is executed. One of the scripts executes nvpmodel, which

is used to change and display the power mode of a Jetson device, and each mode allows several configurations with various CPU frequencies and numbers of cores online. Finally, tail /var/log/syslog was several times executed to check the status of an incron job, if it was triggered, if it succeeded or if there were errors, and what the actual command was that it executed.

5.1 Reactive behaviour

The GRM is a central cluster tool that suggests/recommends the best configurations for orchestrator/scheduler. It will maintain a list of all Elastic System Nodes sorted by the highest resources' availability to receive tasks, hereafter called List of Sorted Nodes (LSN). This reordering will happen periodically so that the list is always up to date, and thus the GRM's suggestion will be faster. In the GRM's reactive behavior, whenever a violation oc-

Algorithm 5.1 Act on NFR Violation(s)

```
Require: list of all sorted Nodes of the Elastic System
  NodeViolated ← object of Node with NFR Violation(s)
  Law ← list of all active Workers of the NodeViolated
  ListSize ← size of Law
  if ListSize ≥ 1 then
      WorkerMinCpu ← get Worker with less CPU usage of the NodeViolated
      if application's WorkerMinCpu is secure then
          if deadlineMissedRatio's WorkerMinCpu ≥ deadlinesThreshold then
              deactivate WorkerMinCpu
              BestNode ← get Node with higher resources' availability
              suggest activation of WorkerMinCpu in BestNode
          else
              CompUnitsDiff ← number of the dimensions with violations
              if CompUnitsDiff ≤ computing units of WorkerMinCpu then
                  computing units of WorkerMinCpu −= CompUnitsDiff
              else
                  deactivate WorkerMinCpu
                  BestNode ← get Node with higher resources' availability
                  suggest activation of WorkerMinCpu in BestNode
              end if
          end if
      else
          deactivate WorkerMinCpu
      end if
  else
      warns that NodeViolated does not have active Workers          ▷ energy waste
  end if
```

curs, Algorithm 5.1 is executed. This algorithm is basically a load-balancing heuristic. One thread of the GRM maintains the LSN. Then, after getting the $NodeViolated$, Law and $ListSize$, it checks if there are still Workers active in the overloaded Node. If there aren't any, an alert is issued about that situation (which could mean that energy is being wasted). If there are, it gets the Worker with the lowest CPU consumption ($WorkerMinCpu$) of

the $NodeViolated$. If the application (that the Worker belongs to) is suffering from cyberattacks, the Worker is immediately deactivated. If security dimension is stable, before reducing the execution capacity ($CompUnits$) of the Worker, it is checked whether it is already exceeding a certain number of failed deadlines ($deadlinesThreshold$). If it is, it is immediately deactivated. If not, then its execution capacity is reduced, depending on the violated dimensions $CompUnitsDiff$. If the calculation of this reduction results in a null or negative execution capacity, then the Worker is deactivated. If not, the computing units will be reduced. Every time a Worker is deactivated, the GRM informs which Node is more available to receive Workers $BestNode$ by choosing the first Node of LSN to place the Worker recently deactivated. Beyond that, the reason (dimension) why it was deactivated is saved. Later, if it is needed to reactivate this Worker, the dimension must be already stable. Additionally, and not evidenced in the pseudo-code of Algorithm 5.1, some of the reasons to not consider specific Nodes to activate Workers are:

- Time | Energy: The difference between the CPU | Energy Node threshold and the total Node CPU | Energy usage is lesser than the mean of the Worker CPU | Energy usage (if it is known) to be reactivated;
- Communication: Impossible to establish communication;
- Security: isSecure parameter is false, meaning that the Node is cybersecurity threatened.

5.2 Predictive/Proactive behaviour

For the GRM predictive/proactive behavior, the main objective is to introduce some way of predicting the system's behavior and act in advance and not just reactively. For that, and because there is no exact definition of what type of ML algorithm works best for each problem type, the only way to know if an algorithm can accurately provide predictions for a problem is by testing the developed models [79].

Firstly, the creation of a dataset. Some data was collected from the production environment, but only 50% of the code that will be running when the project is completed was implemented; at the time of collection, it was just recording data for the use cases and processing them a bit, so it was not possible to collect meaningful data on actual use cases running on Edge. For this reason, new data was simulated where several scripts with different workloads were executed randomly to simulate the use cases resources' consumption. The collected data from the real environment (hereafter referred to as CDRE) was measured every 5 seconds, since it is being collected from a third-party tool. It included the following metrics: CPU consumption and capacity, RAM used and capacity, power consumption, disk usage statistics (amount of data read from and written), network statistics (total data received and transferred from all interfaces) and the timestamp. The simulated data (hereafter referred to as SD) was collected every second. The metrics were almost identical; instead of the first four mentioned, the CPU and RAM percentage was collected. In addition, other scripts were created to measure the consumption of the simulated workloads. Each metric (CPU, RAM, disk, net and power) was measured on a different thread to collect the resources' consumption at the same instant. In both cases, the data was sent to a NoSQL database, more specifically MongoDB.

After the data collection, the next step was data preparation or data cleaning. A little processing was needed in the CDRE. This data was collected from the Nuvla API, and not

Table 5.4: Metrics results for CDRE dataset and regression algorithms

Model	RMSE		R2		MAE		MSE		Accuracy	
MLR	0.37	2.0^3	0.80	-1.5^7	0.29	1.4^3	0.14	4.1^6	0.86	-
PR	0.14	3.9^6	0.97	-5.7^13	0.11	2.5^6	1.9^-2	1.6^13	0.95	-
RT	1.1^-4	0.73	1.00	-0.95	0.4^-5	0.53	1.2^-8	0.53	0.99	0.79
RF	1.1^-4	0.73	1.00	-0.95	1.2^-5	0.53	1.2^-8	0.53	0.99	0.79
BT	1.6^-4	0.73	1.00	-0.95	1.7^-5	0.53	2.7^-8	0.53	0.99	0.79
SVR	5.3^-2	1.15	0.99	-3.9	4.5^-2	1.08	2.8^-3	1.32	0.98	0.67
kNN	2.23	2.35	-6.3	-19.4	2.08	2.30	4.97	5.54	0.33	0.29

all data had the desired format. The SD had the extra step of combining the data from multiple collections because of how it was collected, which led to some missing values. For both datasets, the head and tail observations that have missing values were dropped. In the rest of the observations, the average of the previous and subsequent values of the missing value was inserted as well as the treatment of duplicate observations and outliers. The SD dataset will not be mentioned hereafter for having insufficient data volume. After the data preprocessing, the data was split into training and testing sets (70/30) and modeling. Among the metrics collected, CPU consumption was chosen as the metric to predict in order to provide in advance the necessary resources for the applications that will be running.

In the first attempt to predict CPU consumption, a subset of the CDRE dataset (see Figure 5.2) was used to create the following models (abbreviated in Table 5.4 first column): Multiple Linear Regression (MLR), Polynomial Regression (PR), Regression Trees (RT), Random Forest (RF), Bagged Trees (BT), Support Vector Regression (SVR), K-Nearest Neighbor (kNN) regression and later tested with a similar subset of the CDRE dataset (see Figure 5.3).

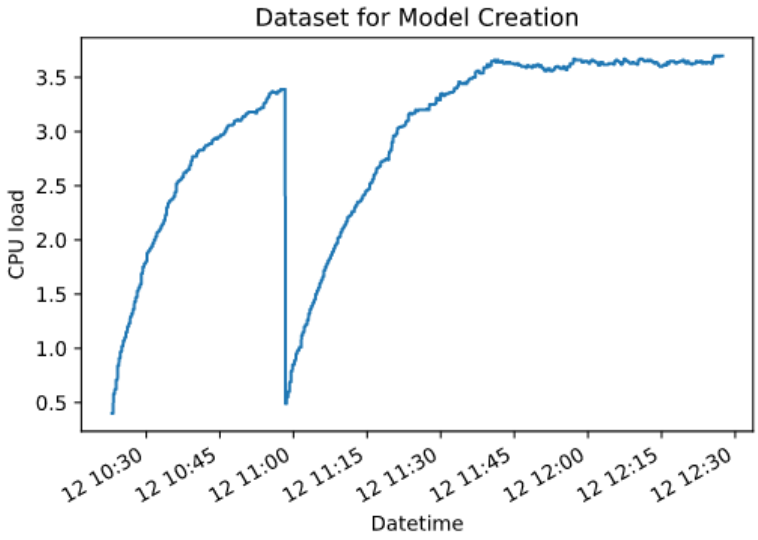

Figure 5.2: CDRE for model creation

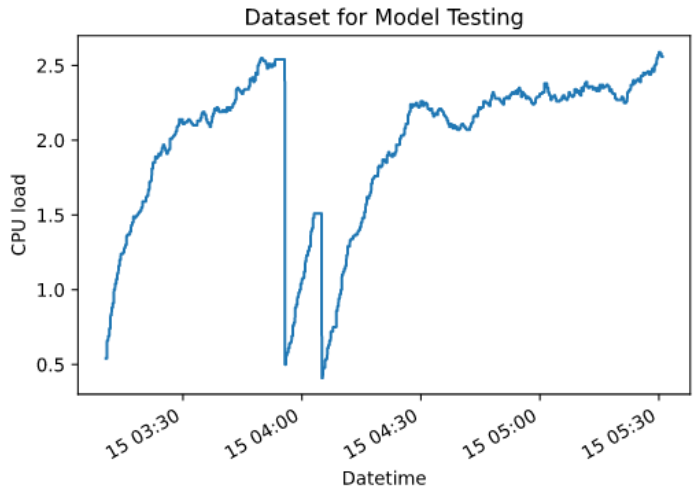

Figure 5.3: CDRE for model testing

The usual metrics to evaluate and compare regression models were collected, such as RMSE, R2, MAE, MSE and Accuracy, already explained in 4. Table 5.4 presents the results of all models for the two subsets. The metrics columns (from RMSE to Accuracy) are divided into two subcolumns, the first being the results of the train set of the dataset in Figure 5.2, and the second being the results of the complete dataset in Figure 5.3 (all values approximated to one or two decimal cases).

Although the algorithms based in trees (RT, RF and BT) present an almost perfect accuracy and R squared with the dataset used to create the model, when new similar data was injected, the accuracy decreased, and the errors got worse. And even though the kNN had bad results, we tried to get the best k and weights through GridSearchCV (from the library sklearn), but bad results were still obtained. Since realistic data collection was not yet possible, the second attempt to predict CPU consumption was to resort to time series prediction. The used dataset was the same to create the regression models presented in Figure 5.2. First, it was checked whether the time series is white noise because if true, it means that it is a sequence of random numbers and cannot be predicted [44]. And given that the time series has a non-zero mean (≈ 2.96), the variance changes over time (e.g., from 0.622 to 0.015 after splitting the series in half), and the values correlate with lag values (see Figure 5.4), we could then proceed because it would only take one of these criteria to be true for the time series not to be white noise.

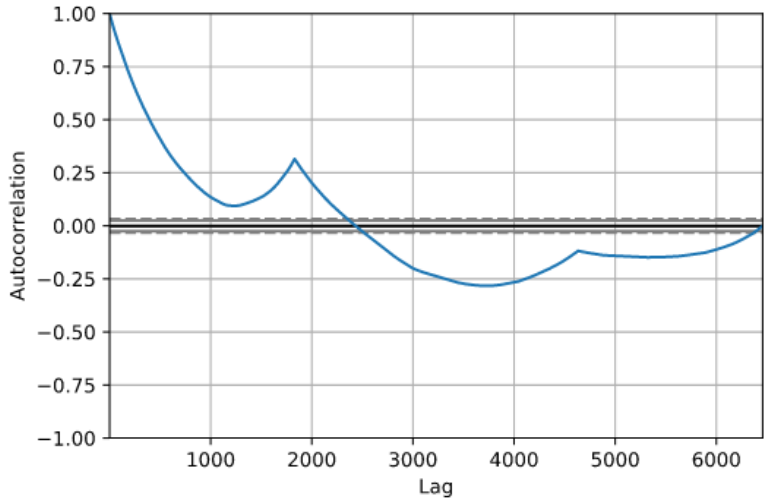

Figure 5.4: Autocorrelation plot

The plot provides the lag number along the x-axis and the correlation coefficient value between -1 and 1 on the y-axis. The plot also includes solid and dashed lines that indicate the 95% and 99% confidence interval for the correlation values. Correlation values above these lines are more significant than those below, providing a threshold or cutoff for selecting more relevant lag values. Then, the time series was decomposed to check its level and noise and verify if it had a trend or seasonality, and as Figure 5.5 shows, the entire series was taken as the trend and that there was no seasonality.

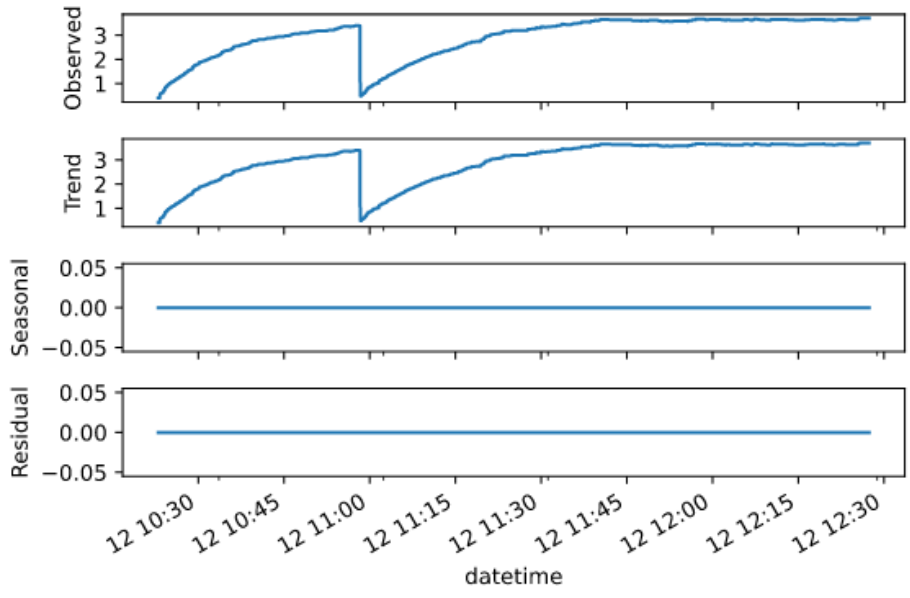

Figure 5.5: Time series decomposed

After this initial analysis, we start to evaluate the models. In this case, train-test splits or k-fold cross-validation are not an option since they ignore the temporal order inherent in the problem. Backtesting was used instead since when the data is split, it respects the temporal order. The used performance measures to evaluate the forecast model were the forecast

error or residual error (expected value minus predicted value), mean forecast error or the forecast bias (average of the forecast error), MAE, MSE and RMSE. Then, a test harness was developed, i.e. we develop a persistence model to determine a baseline performance following the next steps:

- transform the time series into a supervised learning problem;
- establish the train and test datasets for the test harness;
- define the persistence model, make a forecast and establish a baseline performance;
- and finally, review the complete example and plot the output. And from this naive forecast, we achieved "good" results, as shown in Figure 5.6 and Figure 5.7, where the values of RMSE were 0.006 and 0.033, respectively.

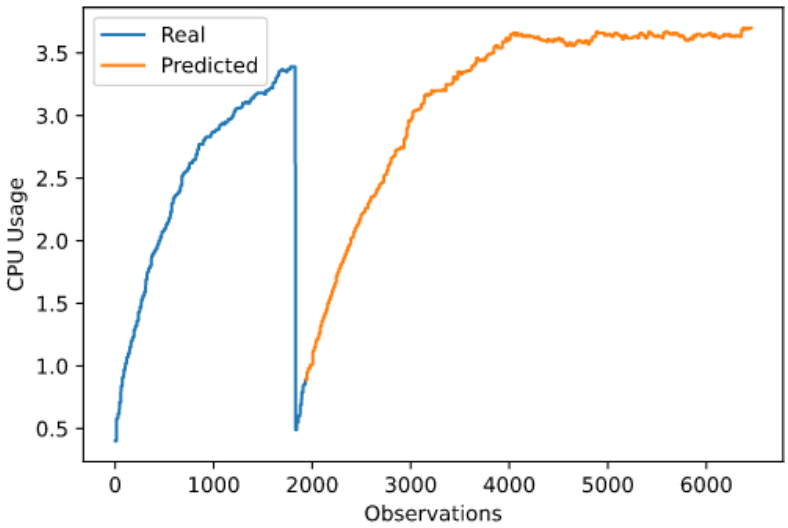

Figure 5.6: CDRE experiment 1

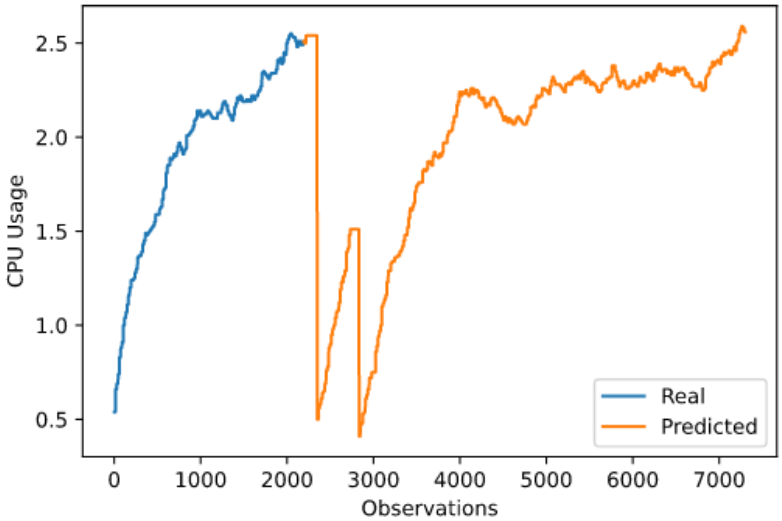

Figure 5.7: CDRE experiment 2

These results are questionably good since the objective was to establish a baseline performance and later apply more numerical methods to minimize a loss or error term.

Chapter 6

Evaluation

In this chapter it will be included the performed test and the addressed requirements for the both dimensions.

Figure 6.1 shows one of the Florence tram line with drafts for the tests, focusing on the southernmost part of the T1 line. Three possible test scenarios were elaborated and for the demonstration of the operation of the tools, it is sufficient to have an ElasticSystem[1] with a GRM and one or more NFRs; so, in essence, any tram stop can be a test scenario. The tram line is composed of at least three intersections Resistenza, Arcipressi and Batoni. Intersections are one of the most complex situations to deal in the tram line since it involves the points of contact between the traffic trajectory of the road vehicles with the tram lines, as shown in Figure 6.2, the Resistenza intersection.

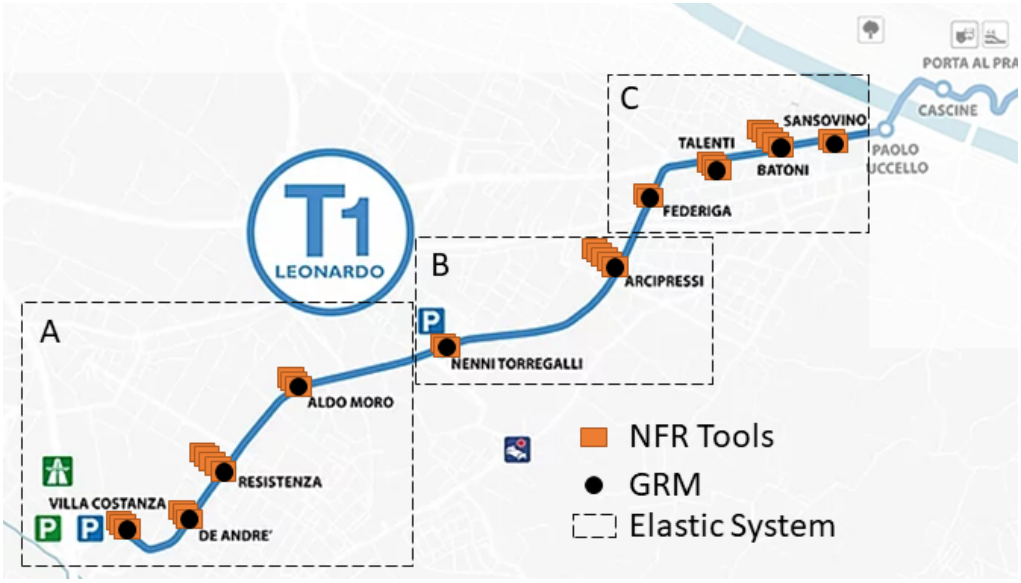

Figure 6.1: Scenario tests based on the real environment

Resistenza intersection was chosen because of the equipment/hardware available in the laboratory and, more specifically, because of the complexity of the included intersection. Each tram stop is composed of at least a cabinet and requires two cameras. Then, in the cabinets of each stop and close to each camera, it is needed one edge device. So, in total, the chosen tram stop needs three edge Nodes and another Node that will execute the GRM.

[1] From now on the concepts will be referenced with their object names in the code for better understanding/comparison with the rest of the text and images. In this case, instead of ELASTIC system, it becomes ElasticSystem.

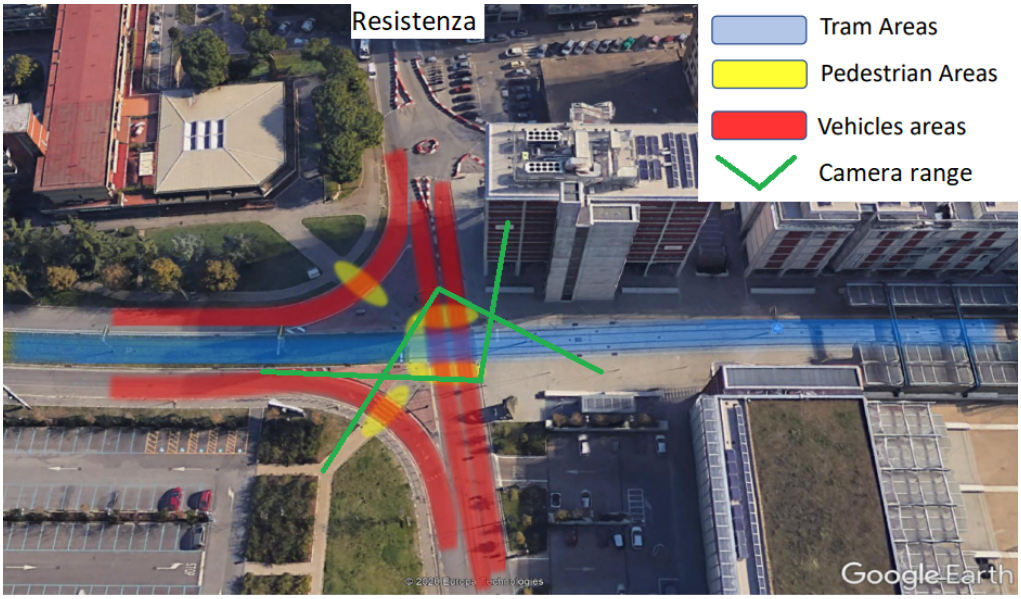

Figure 6.2: Resistenza intersection
Source [2]

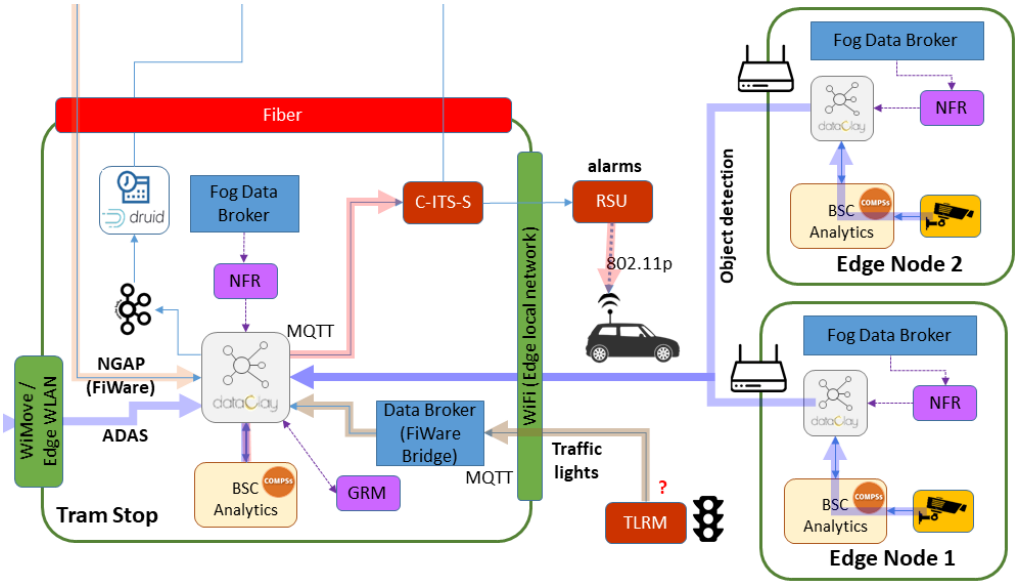

Figure 6.3: Part of the ELASTIC Use Cases diagram
Source [2]

Figure 6.3 helps to understand what each location/situation needs in terms of devices, tools and communication protocols essentially. In the Appendix B, it is attached the figure with all ELASTIC SA components.

As shown in the Figure 6.3, Edge Node 1 and Edge Node 2 have a camera (rectangle with yellow background) as the source of the data. Then, the data is analysed by BSC Analytics (rectangle with orange background) while the Fog Data Broker (rectangle with blue background) is publishing the resource consumption metrics, the NFR (rectangle with purple background) performs an analysis on the metrics, updates dataClay and communicates with the GRM whenever there is the need to do so (commitment of NFR Violations). The

Trams Stops have other software running that doesn't matter for this context (as druid, NGAP(FiWare), etc.), but it is important to point out that it is the only place where the GRM will be running.

With this in mind, the next two sections describe the attributes being evaluated and the different threshold violation being reported to the GRM regarding Time and Energy dimensions.

6.1 Time dimension

Decision making policies CPU and RAM mean usage are the metrics being monitored to evaluate the time dimension in the system, at Node and Worker levels. NFR tool, or more specifically, the Monitor gets from Nuvla API the time metrics and alerts when NuvlaBox is offline. Resource Manager verifies (periodically) if the CPU and RAM consumption do not exceed their thresholds (CPU threshold and RAM capacity), and if they exceed, the Resource Manager publishes a message to GRM notifying the committed violation with the following information: location (Node IP address) and metric (CPU or RAM) or dimension (time).

6.1.1 Tests for the violations of requirements

At least two tests were performed to evaluate the NFR tool Time and Energy, focusing on the time dimension. Some scripts were developed to perform the first test in order to facilitate the generation of NFR Violations and thus be able to prove the proper functioning of the NFR Tool. For the second test, it was necessary to integrate COMPSs with the NFR Tool and define other limits besides CPU and RAM. Table 6.1 presents the ID, name and addressed requirements for the two described tests.

Table 6.1: Time tests

Test ID	Name	Addressed requirements
TIM-1	Monitoring and management of time dimension	Real-time requirements
		Scope of real-time requirements
		Real-time scheduling
		Multi-core scheduling
TIM-2	Monitoring of metrics provided by COMPSs	Measurement of soft time requirements
		Worst-case execution time measurements
		Execution time on multi-core/many-core platforms
		Allocation of components to nodes

The next two subsections show how a user might reproduce the tests that are available in [80].

TIM-1 Monitoring and management of time dimension

In the proper class of the GRM (FakeElasticSystem), define some Workers and put a low threshold in the docker-compose file. Then, start GRM and NFR Tools. The NFR Tool will search for a fake Elastic System with an alias equals to "system". Then, it pulls data from Nuvla API telemetry (for Node level metrics) and Docker API (for Worker level metrics), evaluates the resources' consumption and it will publish violations to GRM. Additionally, another thread writes the resource's consumption in dataClay.

TIM-2 Monitoring of metrics provided by COMPSs

It is only necessary to have at least one Worker running because without the latter, there will be no deadlines misses to evaluate. Then, start GRM and NFR Tools. The NFR Tool will search for a fake Elastic System with an alias equals to "system" and then will read the *deadlinesMissedRatio* captured and shared by COMPSs on the dataClay and evaluate it.

6.1.2 Evaluation results

Table 6.2 presents the expected results for the time tests and above it is detailed additional comments.

Table 6.2: NFR tool (time) evaluation results in lab

ID	Result
TIM-1	[NFR] Resource Manager sent violation '{"nodeIP":<>,"dimension":"cpu"}' [GRM] Received violation '{"nodeIP":<>,"dimension":"cpu"}' [GRM] Suggestion for COMPSs scheduler: [GRM] Worker X should run in Y computing units
TIM-2	[NFR] Resource deadlineMisses is fine... OR [NFR] Resource Manager sent violation '{"nodeIP":<>,"dimension":"cpu"}'

For TIM-1, in case you get a message like "Worker PID X is a static Worker" check if Workers are being activated or ignore it since not all tests need real workload.

For TIM-2, in case you don't get any of the results, possibly you're using a version without COMPSs integration.

6.2 Energy dimension

Several metrics related with energy/power and temperature are monitored to evaluate the energy dimension at a Node level. To calculate the energy consumption at a Worker level, the Worker's CPU usage is multiplied by the Node's energy consumption and only then it is stored in the dataClay. Similarly to time dimension, the Monitor gets the metrics from Nuvla API and alerts when NuvlaBox is offline; the Resource Manager checks if the actual energy's consumption is surpassing the Energy's threshold and if so, it sends the NFR Violation to GRM with the same information's structure: location (Node IP address) and metric (power and/or temperature) or dimension (energy). Lastly, at this moment, the temperature is being monitored and utilized to control the fan of the device, i.e. if it exceeds a specific threshold, the fan is switched on and when the temperature falls below another limit, it is switched off.

6.2.1 Tests for the violations of requirements

Two tests were also carried out to evaluate the NFR tool Time and Energy, now focusing on the energy dimension. For the first test, methods have been added to be able to change the energy threshold in order to simplify the origination of NFR Violations and thus be able

to prove the proper functioning of the NFR Tool. For the second test, some scripts were developed to trigger the machine's fan. Table 6.3 presents the ID, name and addressed requirements of the described tests above.

Table 6.3: Energy tests

Test ID	Name	Addressed requirements
NRG-1	Energy-aware scheduling	Energy monitoring on the edge
		Energy monitoring on the cloud
		Energy-aware scheduling
		Multi-core energy mechanisms
NRG-2	Temperature monitoring	Energy monitoring on the edge
		Energy monitoring on the cloud

The next two subsections show how a user might reproduce the tests that are available in [80].

NRG-1 Energy-aware scheduling

In the GRM repository [81], there are scripts that are available to setup the machines in order to generate violations as soon as the tools start running/monitoring. Then, start GRM and NFR Tools. As the system runs, the user can change the threshold stop and/or restart the generation of violations in the following format:

```
mosquitto_pub -h <IP address of GRM> -p 1884 -t violations -m
'{"IP":"<IP address of Node under test>","energyThreshold":0}'
```

NRG-2 Temperature monitoring

Similarly to the previous test, the GRM repository [81] has one script that is available to setup the machines in order to generate violations as soon as the tools start running/monitoring. This script basically shrinks the range of the thresholds for turning the fan on or off. Then, start GRM and NFR Tools.

6.2.2 Evaluation results

Table 6.4 presents the expected results for the energy tests and above it is detailed additional comments.

Table 6.4: NFR tool (energy) evaluation results in lab

ID	Result
NRG-1	[NFR] Resource Manager sent violation '{"nodeIP":<>,"dimension":"energy"}' [GRM] Received violation '{"nodeIP":<>,"dimension":"energy"}' [GRM] Suggestion for COMPSs scheduler: [GRM] Worker X should run in Y computing units
NRG-2	[NFR] Resource temperature is fine... OR [NFR] Resource Manager sent violation '{"nodeIP":<>,"dimension":"temperature"}'

For NRG-1, you can ignore the initial zero values of the energy consumption.

For NRG-2, in case you don't get the expected result, it can mean that the board does not have a fan or it would be necessary to the correct configurations to the NFR Tool to support the device.

Finally, the addressed requirements of time and energy are presented in Tables 6.5 and 6.6 respectively.

Table 6.5: Addressed Time Requirements

Name	Evaluation	Requirement Status
Real-time requirements	Test	Partial. ELASTIC supports applications with soft-real time requirements.
Scope of real-time requirements	Test	The ELASTIC SA supports applications both local to node and distributed in the compute continuum. Scheduling to cope with real-time requirements is done at the application level.
Measurement of soft timing requirements	Test	Partial. The NFR Tool supports detection of several different types of timing property failure, currently it implements CPU utilization threshold.
Real-time scheduling	Test	Real-time constraints are supported both in the application to node mapping, as well as application task scheduling.
Multi-core scheduling	Test	The COMPSs scheduler supports scheduling of parallel computation in multicore platforms.
Worst-case execution time measurements	Test	The NFT Tool monitors execution time through online measurement.
Execution time on multi-core /many-core platforms	Test	The NFT Tool monitors execution time through online measurement.
Allocation of components to nodes	Test	COMPSs allocates application tasks both statically at deployment time as well as dynamically during execution.

All in all, almost all requirements have been fully completed. Some of the exceptions are due to changing the requirement above or redefining the person responsible for that requirement. But the biggest problem was really the COVID pandemic because it delayed the implementation of the code by all partners due to the new reality that everyone was subjected to (e.g. working from home). The reactive behaviour of the GRM can be said to be 95% complete given that the integration with COMPSs is not 100% integrated. The predictive behaviour suffered the most since until the current date (October 2021), the complete software that would run on the ELASTIC system was not completely ready. Although it was still possible to collect data from the real environment, these did not represent the real (or approximate) workload, which prevents us from saying that the aforementioned techniques could be applied and the GRM would be able to act in a reactive way in unexpected cases and the rest of the time in a predictive/proactive way.

Table 6.6: Addressed Energy Requirements

Name	**Evaluation**	**Requirement Status**
Energy monitoring on the edge	Test	The NFR Tool monitors the energy consumption in all nodes.
Energy monitoring on the cloud	Test	The NFR Tool monitors the energy consumption in all nodes.
Energy-aware scheduling	Test	Upon detection of energy violations, the NFR Tool changes mapping of application workers, enabling COMPSs scheduler to make energy-aware scheduling decisions.
Multi-core energy mechanisms	Test	Upon detection of energy violations, the NFR Tool changes mapping of application workers, enabling COMPSs scheduler to change task-to-core allocation.

Chapter 7

Conclusions

This chapter presents the conclusions about the project developed, namely the objectives achieved, the limitations encountered and the improvements that can be made in the future.

7.1 Results

This dissertation presented the research and implementation of two tools to help schedulers or orchestrators distributing applications that may run on devices from the Edge to the Cloud. The NFR tool focuses on pulling information about resource consumption, writes it to the dataClay, and alerts the GRM in case of NFR violations. The GRM listens to various NFR tools, makes decisions on NFR violations to optimize the system workload, and communicates with the dataClay so that COMPSs can read the proposed decisions. This work is being used in the ELASTIC project, which will bring the possibility to improve the GRM heuristics, mainly in the proactive behaviour, since soon more sensors will be installed on the tram, which will enable the collection of more realistic data. The data collected from the production environment at the time of the dissertation development was very stable, which is one of the reasons to achieve "good" results. For now, to handle the unpredictable system loads, the GRM takes a reactive behaviour.

7.2 Limitations

The delays in the ELASTIC project was the most significant limitation encountered. The fundamental contribution expected from this dissertation was the study of how to influence schedulers and orchestrators in critical and real environments. However, the latter negatively impacted achieving the dissertation's objectives, which, most certainly, wouldn't happen if it was in a simulated environment. The COVID-19 pandemic was the main reason the project suffered several delays, firstly in the implementation of the sensors and, consequently, in the software deployment and respective software tests. Furthermore, the work for the project was divided among several partners, which inherently sometimes delayed the development due to time/availability incompatibilities.

7.3 Future Work

As future work, we intend to continue studying and improving resource prediction in continuum computing environments after the final installation of the sensors.

Specifically, on the resource management of the reactive behaviour, the load averages (reported in 1-minute, 5-minute, and 15-minute) could be used to understand if the load is

increasing or decreasing (one possible way to understand/predict CPU consumption without resorting to Machine Learning techniques). For example, if the 1-minute average is higher/lower than the 5 or 15-minute averages, then the load is increasing/decreasing [82].

Regarding preventive/proactive behaviour, the first and most crucial step focuses on collecting data from the completed ELASTIC system; or, at least, one of its systems, namely the positioning and obstacle detection, the predictive maintenance and energy consumption and the application to manage the coexistence of public and private means of transport.

There are other metrics that have been added to the dataClay model, but are not yet well explored. The *deactivationReasons* list in the Worker object is one example. This list currently helps to decide the amount of computing units to reduce (the more dimensions have violations, the more computing units are reduced). By doing an evaluation at the Node level, we can come to conclusions that a Node has some specific problem, e.g. if it has constant energy (temperature) violations, it may be because it is overexposed to unfavourable weather conditions. At the ELASTIC system level, we can verify which dimensions normally cause more NFR violations, e.g. if time violations are higher only in one area of the tram line, it may be necessary to add computing capacity in that area or improve the algorithms to better distribute the tasks to be executed. Whether we apply Machine Learning techniques or simpler statistics like absolute and relative frequencies to these unexplored metrics, it would be possible to add, or even improve, reactive and preventative heuristics,always with the goal of reducing the amount of violations.

Bibliography

[1] P. Berrone and J. E. Ricart, "IESE Cities in Motion Index 2020," IESE Business School, Navarra, Tech. Rep., 2020, pp. 1–112. doi: https://dx.doi.org/10.15581/018.ST-542.

[2] *Developing a novel software architecture for extreme-scale analytics*, https://elastic-project.eu/, Accessed December 28, 2020.

[3] C. Insights, *What is edge computing?* https://www.cbinsights.com/research/what-is-edge-computing/, Accessed February 13, 2021.

[4] N. Choudhury, "World Wide Web and Its Journey from Web 1.0 to Web 4.0," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 6, pp. 8096–8100, 2014, issn: 14712105. doi: 10.1186/1471-2105-9-82.

[5] W. Tian and Y. Zhao, "An Introduction to Cloud Computing," in *Optimized Cloud Resource Management and Scheduling*, Elsevier, 2015, pp. 1–15. doi: 10.1016/b978-0-12-801476-9.00001-x.

[6] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-degree compared," *Grid Computing Environments Workshop, GCE 2008*, 2008. doi: 10.1109/GCE.2008.4738445. arXiv: 0901.0131.

[7] Armbrust, A. and Fox, R. Griffith, M., "Above the clouds: A Berkeley view of cloud computing," *University of California, Berkeley, Tech. Rep. UCB*, pp. 07–013, 2009, issn: 00010782. doi: 10.1145/1721654.1721672. arXiv: 0521865719978052186571.

[8] L. M. Vaquero, L. Rodero-merino, J. Caceres, and M. Lindner, "A Break in the Clouds : Towards a Cloud Definition," vol. 39, no. 1, pp. 50–55, 2009.

[9] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *Lecture Notes in Electrical Engineering*, 2011, issn: 18761119. doi: 10.1007/978-981-13-1056-0_66. [Online]. Available: https://nvlpubs.nist.gov/%7B%5C%%7D0Anistpubs/Legacy/SP/nistspecialpublication800-145.pdf.

[10] Eugene Gorelik, "Cloud Computing Models," no. January, pp. 1–81, 2013, issn: 03029743. arXiv: 1003.4074.

[11] A. Sunyaev, *Internet computing*, 1st ed., 6. Springer, Cham, 2020, vol. 9, pp. 4–6, isbn: 9783030349561. doi: 10.1109/MIC.2005.125.

[12] P. Levine, *The End of Cloud Computing*, 2016. [Online]. Available: https://a16z.com/2016/12/16/the-end-of-cloud-computing/.

[13] M. Aazam and E. N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT," *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, vol. 2015-April, pp. 687–694, 2015, issn: 1550445X. doi: 10.1109/AINA.2015.254.

[14] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014, issn: 15513203. doi: 10.1109/TII.2014.2300753.

[15] F. Wortmann and K. Flüchter, "Internet of Things: Technology and Value Added," *Business and Information Systems Engineering*, vol. 57, no. 3, pp. 221–224, 2015, issn: 18670202. doi: 10.1007/s12599-015-0383-3.

[16] Dave Evans, "The Internet of Things," *CISCO white paper*, no. April, pp. 1–78, 2011, issn: 09598138. arXiv: 1011.1669v3. [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

[17] H. Minn, M. Zeng, and V. Bhargava, "Towards a definition of the {Internet of Things (IoT)}," *IEEE Internet Initiative*, pp. 1–86, 2015.

[18] R. Ande, B. Adebisi, M. Hammoudeh, and J. Saleem, "Internet of Things: Evolution and technologies from a security perspective," *Sustainable Cities and Society*, vol. 54, no. February 2019, pp. 1–15, 2019, issn: 22106707. doi: 10.1016/j.scs.2019.101728. [Online]. Available: https://doi.org/10.1016/j.scs.2019.101728.

[19] A. Miles, A. Zaslavsky, and C. Browne, "IoT-based decision support system for monitoring and mitigating atmospheric pollution in smart cities," *Journal of Decision Systems*, vol. 27, no. May, pp. 56–67, 2018, issn: 21167052. doi: 10.1080/12460125.2018.1468696. [Online]. Available: https://doi.org/10.1080/12460125.2018.1468696.

[20] B. Diène, J. J. Rodrigues, O. Diallo, E. H. M. Ndoye, and V. V. Korotaev, "Data management techniques for Internet of Things," *Mechanical Systems and Signal Processing*, vol. 138, 2020, issn: 10961216. doi: 10.1016/j.ymssp.2019.106564.

[21] Domo, Inc., *Connecting Your Data, Systems & People | Domo*. [Online]. Available: https://www.domo.com/.

[22] ——, *Domo Resource - Data Never Sleeps 5.0*, Accessed February 27, 2021, 2017. [Online]. Available: https://www.domo.com/learn/data-never-sleeps-5.

[23] ——, *Domo Resource - Data Never Sleeps 8.0*, Accessed February 27, 2021, 2020. [Online]. Available: https://www.domo.com/learn/data-never-sleeps-8.

[24] ——, *Domo Resource - Data Never Sleeps 7.0*, Accessed February 27, 2021, 2019. [Online]. Available: https://www.domo.com/learn/data-never-sleeps-7.

[25] International Data Corporation (IDC), *Idc - about - home*, IDC: The premier global market intelligence company, 2019. [Online]. Available: https://www.idc.com/about (visited on 02/27/2021).

[26] I. D. G. (IDC), *The growth in connected iot devices is expected to generate 79.4zb of data in 2025, according to a new idc forecast*, web.archive.org, Jun. 2019. [Online]. Available: https://web.archive.org/web/20210102101203/https://www.idc.com/getdoc.jsp?containerId=prUS45213219 (visited on 02/27/2021).

[27] P. Russom, "Big Data Analytics," Tech. Rep., 2011, pp. 1–34. doi: 10.1017/9781108566506.005.

[28] V. Shobana and N. Kumar, "Big data - A review," *International Journal of Applied Engineering Research*, vol. 10, no. 55, pp. 1294–1298, 2015, issn: 09739769. doi: 10.26634/jit.6.1.13507.

[29] S. Yin and O. Kaynak, "Big Data for Modern Industry: Challenges and Trends," *Proceedings of the IEEE*, vol. 103, no. 2, pp. 143–146, 2015, issn: 15582256. doi: 10.1109/JPROC.2015.2388958.

[30] B. Mirza, T. Q. Syed, B. Khan, and Y. Malik, "Potential Deep Learning Solutions to Persistent and Emerging Big Data Challenges—A Practitioners' Cookbook," *ACM Computing Surveys*, vol. 54, no. 1, pp. 1–39, 2021, issn: 0360-0300. doi: 10.1145/3427476.

[31] R. H. Hariri, E. M. Fredericks, and K. M. Bowers, "Uncertainty in big data analytics: survey, opportunities, and challenges," *Journal of Big Data*, vol. 6, no. 44, pp. 1–16, 2019. doi: 10.1186/s40537-019-0206-3. [Online]. Available: https://doi.org/10.1186/s40537-019-0206-3.

[32] J. Zenkert, M. Dornhofer, C. Weber, C. Ngoukam, and M. Fathi, "Big data analytics in smart mobility: Modeling and analysis of the Aarhus smart city dataset," in *Proceedings - 2018 IEEE Industrial Cyber-Physical Systems, ICPS 2018*, Institute of Electrical and Electronics Engineers Inc., 2018, pp. 363–368, isbn: 9781538665312. doi: 10.1109/ICPHYS.2018.8387685.

[33] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959, issn: 0018-8646. doi: 10.1147/rd.33.0210. [Online]. Available: http://ieeexplore.ieee.org/document/5392560/.

[34] T. W. Ruger, P. T. Kim, A. D. Martin, and K. M. Quinn, *The supreme court forecasting project: Legal and political science approaches to predicting supreme court decisionmaking*, 2004. doi: 10.2307/4099370. [Online]. Available: https://www.jstor.org/stable/4099370?origin=crossref.

[35] S. Russel and P. Norvig, *Artificial Intelligence A Modern Approach*, isbn: 9781626239777.

[36] I. E. Naqa, M. J. Murphy, I. E. Naqa, and M. J. Murphy, "What is machine learning?," 2015. doi: 10.1007/978-3-319-18305-3_1.

[37] N. Dutta, S. Umashankar, V. K. A. Shankar, S. Padmanaban, Z. Leonowicz, and P. Wheeler, "Centrifugal pump cavitation detection using machine learning algorithm technique," *Proceedings - 2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe, EEEIC/I and CPS Europe 2018*, Oct. 2018. doi: 10.1109/EEEIC.2018.8494594.

[38] *Linear regression | machine learning tutorial*. [Online]. Available: https://sci2lab.github.io/ml_tutorial/linear_regression/.

[39] *Lecture 10: Regression Trees*, 2006. [Online]. Available: http://www.stat.cmu.edu/$%5Csim$cshalizi/350-2006/lecture-10.pdf.

[40] G. Vasconcelos, *How Random Forests improve simple Regression Trees? | insightR*, 2017. [Online]. Available: https://insightr.wordpress.com/2017/09/23/how-random-forests-improve-simple-regression-trees/ (visited on 10/12/2021).

[41] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. [Online]. Available: https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12_toc.pdf.

[42] B. M. Achsan, *Support vector machine: Regression*. [Online]. Available: https://medium.com/it-paragon/support-vector-machine-regression-cf65348b6345.

[43] *Chapter 7 regression i: K-nearest neighbours | data science: A first introduction*. [Online]. Available: https://ubc-dsci.github.io/introduction-to-datascience/regression1.html.

[44] J. Brownlee, *Introduction to Time Series Forecasting with Python*.

[45] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 2nd. OTexts, 2018. [Online]. Available: https://otexts.com/fpp2/.

[46] F. Lordan, E. Tejedor, J. Ejarque, R. Rafanell, J. Álvarez, F. Marozzo, D. Lezzi, R. Sirvent, D. Talia, and R. M. Badia, "Servicess: An interoperable programming framework for the cloud," *Journal of Grid Computing*, vol. 12, pp. 67–91, Sep. 2013. doi: 10.1007/s10723-013-9272-5. (visited on 02/10/2021).

[47] J. Martí, A. Queralt, D. Gasull, A. Barceló, J. José Costa, and T. Cortes, "Dataclay: A distributed data store for effective inter-player data sharing," *Journal of Systems and Software*, vol. 131, pp. 129–145, Sep. 2017. doi: 10.1016/j.jss.2017.05.080. (visited on 02/10/2021).

[48] *Enable your edge with our management platform as a service*, https://nuvla.io/, Accessed February 10, 2021.

[49] M. Holmes, *The modern software supply chain runs on docker*, https://www.docker.com/blog/the-modern-software-supply-chain-runs-on-docker/, Accessed December 13, 2020.

[50] *2020 developer survey*, https://insights.stackoverflow.com/survey/2020, Accessed February 18, 2021.

[51] C.-H. Hong and B. Varghese, "Resource Management in Fog/Edge Computing," *ACM Computing Surveys*, vol. 52, no. 5, pp. 1–37, 2019, issn: 0360-0300. doi: 10.1145/3326066.

[52] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," *Softw. Pract. Exper.*, vol. 41, no. 1, pp. 23–50, 2011, issn: 0038-0644. doi: 10.1002/spe.995. [Online]. Available: https://doi.org/10.1002/spe.995.

[53] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, pp. 39–44, 2018, issn: 21613915. doi: 10.1002/ett.3493.

[54] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Software - Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017, issn: 1097024X. doi: 10.1002/spe.2509. arXiv: 1606.02007.

[55] I. R. Management, "Resource Management and Quality of Service," *Quality*, pp. 1–43, 2009.

[56] X. He, Z. Ren, C. Shi, and J. Fang, "A novel load balancing strategy of software-defined cloud/fog networking in the Internet of Vehicles," *China Communications*, vol. 13, pp. 140–149, 2016, issn: 16735447. doi: 10.1109/CC.2016.7833468.

[57] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic Service Placement for Mobile Micro-Clouds with Predicted Future Costs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002–1016, 2017, issn: 10459219. doi: 10.1109/TPDS.2016.2604814. arXiv: 1503.02735.

[58] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm," *Proceedings of the IM 2017 - 2017 IFIP/IEEE International Symposium on Integrated Network and Service Management*, no. May, pp. 1222–1228, 2017. doi: 10.23919/INM.2017.7987464.

[59] E. Arianyan, H. Taheri, and S. Sharifian, "Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers," *Computers and Electrical Engineering*, vol. 47, pp. 222–240, 2015, issn: 00457906. doi: 10.1016/j.compeleceng.2015.05.006. [Online]. Available: http://dx.doi.org/10.1016/j.compeleceng.2015.05.006.

[60] R. Sosa, C. Kiraly, and J. D. Parra Rodriguez, "Offloading execution from edge to cloud: A dynamic node-red based approach," *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom*, vol. 2018-Decem, pp. 149–152, 2018, issn: 23302186. doi: 10.1109/CloudCom2018.2018.00039.

[61] J. Huang, C. Li, and J. Yu, "Resource prediction based on double exponential smoothing in cloud computing," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks, CECNet 2012 - Proceedings*, 2012, pp. 2056–2060, isbn: 9781457714153. doi: 10.1109/CECNet.2012.6201461.

[62] J. F. Martínez and E. Ipek, "Dynamic Multicore Resource Management: A Machine Learning Approach," pp. 9–17, 2009.

[63] V. K. Singh and D. Vandermeer, "Estimating the Energy Consumption of Executing Software Processes," 2013. doi: 10.1109/GreenCom-iThings-CPSCom.2013.40.

[64] F. Schmidt, M. Niepert, and F. Huici, "Representation Learning for Resource Usage Prediction," pp. 1–3, 2018. eprint: 1802.00673. [Online]. Available: http://arxiv.org/abs/1802.00673.

[65] G. Kaur, A. Bala, and I. Chana, "An intelligent regressive ensemble approach for predicting resource usage in cloud computing," *Journal of Parallel and Distributed Computing*, vol. 123, pp. 1–12, 2019, issn: 07437315. doi: 10.1016/j.jpdc.2018.08.008.

[66] N. Rich and M. Holweg, "Value Analysis, Value Engineering," *INNOREGIO: dissemination of innovation and knowledge management techniques*, pp. 1–31, 2000.

[67] P. A. Koen, H. M. J. Bertels, and E. J. Kleinschmidt, "Managing the front end of innovation—part ii," doi: 10.5437/08956308X5703199.

[68] P. A. Koen, G. M. Ajamian, S. Boyce, A. Clamen, E. Fisher, S. Fountoulakis, A. Johnson, P. Puri, and R. Seibert, "Fuzzy Front End : and Techniques," *Industrial Research*, pp. 5–35, 1996. [Online]. Available: http://www.stevens.edu/cce/NEW/PDFs/FuzzyFrontEnd%7B%5C_%7DOld.pdfNEW/PDFs/FuzzyFrontEnd%7B%5C_%7DOld.pdf.

[69] Prometheus Authors, *Overview | Prometheus*. [Online]. Available: https://prometheus.io/docs/introduction/overview/ (visited on 09/22/2021).

[70] J. Kukunas, *Power and Performance: Software Analysis and Optimization*. 2015, pp. 1–284, isbn: 9780128008140. doi: 10.1016/C2013-0-18946-5.

[71] Z. Yongchang, *Kernel perf tool user guide*, 2017.

[72] J. Lelli, C. Scordino, L. Abeni, and D. Faggioli, "Deadline scheduling in the Linux kernel," *Software: Practice and Experience*, vol. 46, no. 6, pp. 821–839, Jun. 2016, issn: 1097-024X. doi: 10.1002/SPE.2335. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/spe.2335%20https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2335%20https://onlinelibrary.wiley.com/doi/10.1002/spe.2335.

[73] Cloud Native Computing Foundation, *Control CPU Management Policies on the Node | Kubernetes*. [Online]. Available: https://kubernetes.io/docs/tasks/administer-cluster/cpu-management-policies/#cpu-management-policies (visited on 07/15/2021).

[74] Barcelona Supercomputing Center, *Schedulers — COMPSs 2.9 documentation*. [Online]. Available: https://compss-doc.readthedocs.io/en/stable/Sections/03_Execution_Environments/03_Schedulers.html# (visited on 07/15/2021).

[75] "Machine Learning-Based Scaling Management for Kubernetes Edge Clusters," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 958–972, 2021, issn: 19324537. doi: 10.1109/TNSM.2021.3052837.

[76] Alexeiled, *stress-ng - Docker Image*. [Online]. Available: https://hub.docker.com/r/alexeiled/stress-ng (visited on 07/19/2021).

[77] I. Docker, *Runtime options with Memory, CPUs, and GPUs | Docker Documentation*. [Online]. Available: https://docs.docker.com/config/containers/resource_constraints/ (visited on 07/13/2021).

[78] ——, *Docker Engine API v1.29 Reference*. [Online]. Available: https://docs.docker.com/engine/api/v1.29/#operation/ContainerUpdate (visited on 07/14/2021).

[79] D. H. Wolpert, "The Lack of A Priori Distinctions Between Learning Algorithms," *Neural Computation*, vol. 8, no. 7, pp. 1341–1390, Oct. 1996, issn: 0899-7667. doi: 10.1162/NECO.1996.8.7.1341. [Online]. Available: http://direct.mit.edu/neco/article-pdf/8/7/1341/813495/neco.1996.8.7.1341.pdf.

[80] *Elastic-h2020 / elastic-sa / nfr-tool / nfrtool - time and energy · gitlab*. [Online]. Available: https://gitlab.bsc.es/elastic-h2020/elastic-sa/nfr-tool/nfrtool-time-and-energy.

[81] *Elastic-h2020 / elastic-sa / nfr-tool / grm-global$_{r}esource_{m}anagergitlab$*. [Online]. Available: https://gitlab.bsc.es/elastic-h2020/elastic-sa/nfr-tool/grm-global_resource_manager.

[82] Brendan Gregg, *Linux Load Averages: Solving the Mystery*, 2017. [Online]. Available: https://www.brendangregg.com/blog/2017-08-08/linux-load-averages.html (visited on 09/14/2021).

Appendix A

Model of dataClay

Figure B.1 presents the complete class diagram of the dataClay model.

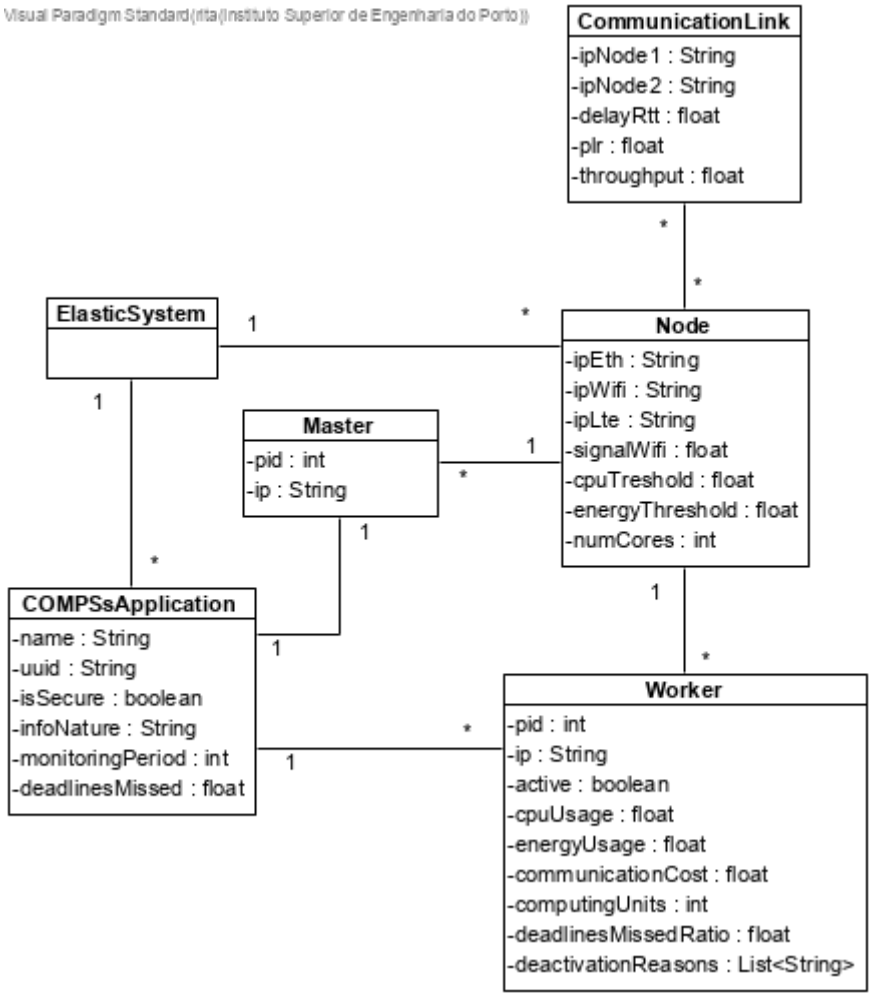

Figure A.1: Complete dataClay model class diagram

Tables A.1 and A.2 presents the complete description of the objects of the dataClay model as well as their attributes.

Table A.1: Complete dataClay classes definitions
Source [2]

Class	**Definition**
ELASTIC System	represents the list of all the data analytics workflows of the ELASTIC infrastructure, and the list of all the distributed computing nodes available in the fog computing platform
COMPSs Application	represents a data analytics workflow distributed with COMPSs
Master	represents the COMPSs master process which is unique for a given data analytics workflow and is deployed in a given node
Worker	represents each of the COMPSs worker processes of a given data analytics workflow and is executed in a given node
Node	represents a distributed computing node of the fog computing platform, with the following attributes
CommunicationLink	represents the communication link between a pair of nodes of the ELASTIC system

Table A.2: Complete dataClay classes' attributes definitions
Source [2]

Class	**Attributes**	**Definition**
COMPSs Application	uuid	unique identifier for the COMPSs application
	name	application name
	isSecure	a flag that denotes whether the application should fulfil the security property (i.e., run only in secure nodes)
	infoNature	a string denoting the type of the application (i.e., video stream, etc.)
	monitoringPeriod	the reporting interval from the NFR tools to the dataClay model
	deadlinesMissed	the ratio of missed application deadlines
Master	PID	PID of the process
	IP	IP of the node where the master is deployed, corresponding to the employed networking interface (i.e, WiFi, Ethernet of LTE)
Worker	PID	PID of the process
	IP	the IP of the node where the worker is deployed, corresponding to the employed networking interface (i.e, WiFi, Ethernet of LTE)
	active	a flag denoting whether the worker should be active. The active flag is set to false by the NFR tool when a worker should not be used by COMPSs due to severe NFR violations (e.g., security property is not met)
	cpuUsage	current CPU usage of a worker
	energyUsage	current energy usage of a worker
	computingUnits	available computing units that can be used by the worker (must be a subset of the number of cores of the node)
	communicationCost	value of the estimated communication cost
	deadlinesMissed	ratio of missed deadlines for the specific worker
	deactivationReasons	a list with the NFR violations that led to the deactivation of the worker (i.e., violations of time, energy, communications, security)
Node	*ipWifi*	the IP of the WiFi networking interface (if available)
	ipEth	the IP of the Ethernet networking interface (if available)
	ipLTE	the IP of the LTE networking interface (if available)
	signalWiFi	the level of the WiFi signal
	cpuThreshold	the threshold for the time property (CPU utilization)
	energyThreshold	the threshold of the energy property
	numCores	number of CPU cores of the node
Communication Link	IPNode1	the IP of the first node of the interconnected pair (the order is interchangeable)
	IPNode2	the IP of the second node of the interconnected pair (the order is interchangeable)
	delayRTT	Round Trip Time (RTT) delay of the link
	PLR	Packet Loss Rate (PLR) of the link
	throughput	The measured throughput of the link

Appendix B

All ELASTIC SA components

Figure B.1 presents the all components of the ELASTIC Software Arquitecture that supports the three use cases of the project.

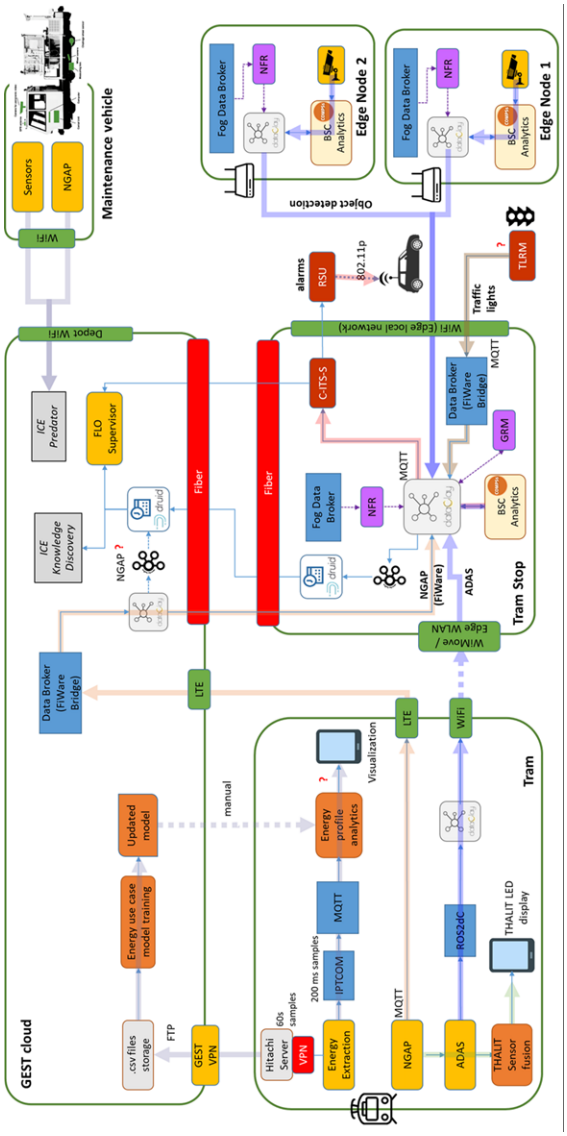

Figure B.1: All ELASTIC SA components supporting all use cases
Source [2]