Institute of Mathematics and Statistics
Bachelor degree in Computer Science

An Edge Computing Platform based on Smart Heaters: Investigating Job Allocation Policies, Describing and Modeling Temperatures

Anderson Andrei DA SILVA

May 15th, 2020

Under the supervision of:
Prof. Alfredo GOLDMAN
Prof. Denis TRYSTRAM
Dr. Danilo CARASTAN

May 2020

Abstract

Fog/Edge computing infrastructures have as one of the most important challenges the allocation of computational jobs with data-sets dependencies in an efficiently way. In this manuscript, we present the usage of a simulated Edge platform and several metrics exemplifying how to develop and compare different scheduling strategies. More precisely, we utilize an on-going project involving academics and a high-tech company that aims at delivering a dedicated tool to evaluate scheduling policies in Edge computing infrastructures. This tool enables the community to simulate various policies and to easily customize researchers/engineers' use-cases, adding new features if needed.

The implementation is built upon the Batsim/SimGrid toolkit, which has been designed to evaluate batch scheduling strategies in various distributed infrastructures. Although the complete validation of the simulation toolkit is still on-going, we demonstrate its relevance by studying different scheduling strategies on top of a simulated version of the Qarnot Computing platform, a production Edge infrastructure based on smart heaters. We were able to develop and simulate the current Qarnot's job allocation strategies, to propose and evaluate different approaches for different metrics.We also propose an analysis of their workloads and data sets dependencies.

Looking on a second perspective, this manuscript also present a descriptive analysis of the real Qarnot Platform logs, in order to better understand its behavior and particularities. Going one step further on a temperature modeling, we built a first attempt of categorization of the Qarnot smart heaters based on their behaviors.

Resumo

Plataformas de *Fog/Edge Computing* tem como um dos mais importantes desafios o escalonamento de trabalhos computacionais com dependências de conjuntos de dados de uma forma eficaz. Neste manuscrito, apresentamos a utilização de uma plataforma Edge simulada e várias métricas exemplificando como desenvolver e comparar diferentes estratégias de escalonamento. Mais precisamente, utilizamos um simulador que tem como objetivo a avaliação de políticas de escalonamento em infra-estruturas de computação em plataformas *Edge*. Esta ferramenta permite a comunidade simular várias políticas de escalonamento e customizá-las facilmente, acrescentando novas funcionalidades se necessário.

A implementação é construída baseada nos simuladores Batsim/SimGrid, que foram concebidos para avaliar estratégias de escalonamento em infra-estruturas distribuídas. Embora a validação completa do simulador utilizador nesse trabalho, fruto da utilização dos últimos dois citados, ainda esteja em curso, demonstramos a sua relevância estudando diferentes estratégias de escalonamento baseadas em uma versão simulada da plataforma da empresa Qarnot Computing, que possui uma infra-estrutura baseada em aquecedores inteligentes. O desenvolvimento e simulação das atuais estratégias de escalonamento da Qarnot, a proposta e avaliação de diferentes abordagens para diferentes métricas e a análise das suas cargas de trabalho e das dependências dos conjuntos de dados serão apresentadas no decorrer deste trabalho.

Numa segunda perspectiva, este manuscrito apresenta também uma análise descritiva dos *logs* reais dos aquecedores inteligentes da plataforma Qarnot, de forma a compreender melhor o seu comportamento e particularidades. Indo um passo além em direção à uma modelagem das temperaturas desses aquecedores, construímos uma primeira tentativa de categorização dos aquecedores inteligentes da Qarnot com base nos suas distribuições de temperatura.

Acknowledgement

This work was supported by the ANR Greco project and by AUSPIN with the International Exchange Program for undergraduate students from University of São Paulo.

I would like to thank my advisors, Professor Denis Trystam for the opportunity, his supporting and teachings, and Professor Alfredo Goldman for indicated me for this Master Program. Also Dr. Yanik Ngoko for having proposed the subject of this thesis, and Dr. Danilo Carastan for the co-supervision.

I would like to thank my family and friends for all the support, my fiancée Andréia, my mom Rosimeri, my father Alex and my brother Leonardo.

Finally, I would like to thank anyone in my work group, colleagues and also ones responsible for the administrative processes from USP and UGA.

Contents

Part I

Introduction, State of the Art and the Qarnot Computing Case Study

— 1 —
Introduction

1.1 Edge Computing and the Internet of Things

The proliferation of Internet of Things (IoT) applications [8] and the advent of new technologies such as Mobile Edge computing [3] and Network Function Virtualization [25] (NFV) have been accelerated the deployment of Cloud Computing like capabilities at the edge of the Internet (i.e, places far from the servers locality). Cloud computing has been targeted of centralized computation, where in general, the data is sent to the Cloud, processed there and delivered to whom requested. However, nowadays, mobile devices have considerable computation power embedded and the usage of these devices has caused new situations, as the partial or total data processing by themselves. So, a paradigm has been emerged in the aspect of data processing, storage and transfer, which is the one to utilize the devices between the users and the Cloud to perform the computations [9]. Referred to as the Fog [10] or the Edge computing [35] paradigms, the main objective is to perform on demand computations close to the place where the data are stored, produced and processed in order to mitigate data exchanges and to avoid too high latency penalties [43]. The main differences from Fog/Edge Computing infrastructures to the Cloud Computing ones are the heterogeneity regarding the network particularities (latency, throughput, etc.) and multiple devices utilization for the same objective. Then, the connectivity between resources is intermittent and the storage/computational resources can join or leave the infrastructure at any time, for an unpredictable duration. At the light of these differences it is necessary a tool to provide a structured Edge platform to allow studies. Among the open questions, one of them is the computation and the data placement problems regarding the Edge Computing scenario, i.e, where to transfer data sets according to their sources and schedule computations to satisfy specific criteria. Although several works have been dealing with questions like that, [41, 11, 38, 40, 26, 13, 5], it is difficult to understand how each proposal behaves in a different context and how they behave among different objectives (scalability, reactivity, etc.). In addition, they have been designed for specific use cases that are evaluated either using *ad hoc* simulators or in limited *in vivo* (i.e., real world) experiments.

1.2 A Simulated Edge Platform

Similarly to what has been proposed for the Cloud Computing paradigm [21], a dedicated simulator toolkit has been developed, Batsim [14], on top of Simgrid [12]. The Simgrid [12] is a scientific instrument to study the behavior of large scale distributed systems such as Grids,

Clouds, HPC (High Performance Computing) or P2P systems [1]. It can be used to evaluate heuristics and prototype applications. The Batsim is a simulator toolkit to analyze batch [2] scheduling strategies, and in particular, it also provides extensions as (i) an Injector, which is an external module to inject any type of event that could occur during the simulation (e.g., a machine became unavailable at a certain time) and (ii) a Storage Controller, to supervise all transfers of data sets within the simulated platform.

In the Part II we will present the Simgrid/Batsim toolkit in more details, its extensions, an Edge simulated platform built on top of them, and the performance analysis of the results from this simulated platform.

1.3 A Case Study

Although the validation of these extensions and the integration of representative edge workloads is still ongoing, the first building blocks implemented enabled the study of an edge infrastructure as complex as the *Qarnot Computing* platform [1]. The *Qarnot Computing* infrastructure is a production platform composed of 3,000 diskless machines distributed across several locations in France and Europe. Each computing resource can be used remotely as traditional Cloud computing capabilities or locally in order to satisfy data processing requirements of IoT devices that have been deployed in the vicinity of the computing resource. Also with computing units and mixed local/global job submissions with data sets dependencies, the *Qarnot* platform is a good example of an Edge infrastructure. As far the simulation of the Qarnot platform was possible utilizing Batsim/SimGrid, it also provided us the possibility to apply different scheduling policies to the Qarnot's jobs. In addition, we had access to some of the Qarnot Platform logs and an extracotr. The Qarnot Platform logs allowed us to perform a descriptive analysis and a categorization procedure, and the Qarnot's Extractor allowed us to extract the real workloads from the Qarnot's platform, providing details of jobs, data sets, its dependencies and so on.

1.4 The Data Analysis Challenge

The novelty since the *Qarnot* foundation is that in addition to provide Cloud and HPC jobs computation, they provide heat services for general environments as houses, apartments or offices. Then, the platform receives two types of user requests: requests for computing and requests for heating, from their machine which are presented as a Smart Heater.

In this chapter, we perform a first investigation of some of the Qarnot Computing's Smart Heater. This investigation is focused in the temperature logs from those machines in order to better understand their temperature and heating mechanisms. We first perform a data cleaning process in these logs to remove noisy and outlier data, then we perform an descriptive analysis in order to have an overview, at first, and then to investigate some details. In the end, we create a first model to analyse the temperature profile of the smart heaters to group the ones with similar temperature characteristics.

[1] Stands for "Peer to Peer", in a P2P network, the "peers" are computer systems which are connected to each other via the Internet

[2] Batch schedulers, or Resource and Jobs Management Systems (RJMS) are systems that manage resources in large-scale computing centers, notably by scheduling and placing jobs.

One of the challenges of this approach is how can we better understand the heating behavior of the smart heaters, since each of these machine are subject to distinct situations (i.e., different processor types, offices with different sizes and thermal isolation efficiency, different heating demands, *etc.*). This better understanding may enable more efficient predictions of the heating characteristics, which in its turn may help to perform a better resource management.

For instance, to illustrate this interesting challenge, let us consider the Smart Heaters utilized at homes and apartments. Let us consider an apartment where a young student lives and other where a senior person lives. Let us imagine that the student gets out home early of the day and comes back late of afternoon. For that day his Smart Heater was turned off for almost half day. Now, let us consider that the senior person hang outs just a little bit, and he stays at home 70% of the day, then his Smart Heater will be tuned on more than the student one. Adding another parameter, maybe the student targets his Smart Heater, in the average 20º C almost all its useful time, and the senior one 24º C. Then, the analyses of both Smart Heater will be completely different from their logs.

Those are very simple details that show us how challenging this context is, and it gets more and more interesting as much as we get close to the reality. As another challenging example, let us consider the interferences of the other appliances, the simple actions of open a window or a door, and so on. A possible way to deal with this kind of heterogeneity of behaviors is to characterize the Smart Heaters by their users. For that, we will look to the temperatures distributions, searching for similarities. But a preliminary step is to understand all the data, to search patterns or particularities that could exist.

In the Part III we present in details the Qarnot Computing's logs, its particularities, a descriptive analysis and a first attempt of categorization of the Qarnot's smart heaters based on their behaviors.

1.5 Main Contributions

Utilizing Batsim, its decision maker component and extensions (Storage Controller and Injector), this work contributed with the development of many scheduling policies which were compared in terms of performance when applied in an use case.

To compare the different scheduling policies from different points of view, this work contributed with the design of experiments that allows with easy modification, execution and visualization of all results provided by the platform. These analyses were conducted by (i) the study of the job's processing time distribution, (ii) the job's dependencies of data sets, (iii) the comparison among the workloads extracted and among the developed policies by several metrics. In addition, beyond the metrics provided by Batsim/Simgrid as waiting time, scheduling time and slowdown this work contributed with the addition of a classical metric utilized in the literature: the bounded slowdown.

Furthermore, this work contributed to the implementation of the remaining components to achieve the full simulation of the Qarnot platform. In order to present details of its implementation, we will depict the whole platform giving an overview of the Edge placement simulator. After presenting the simulated platform built with Batsim/Simgrid, we will describe how the *Qarnot* infrastructure has been instantiated on top of the simulator, how the injector was used to simulate the *Qarnot* workload and, finally, how was developed and evaluated different scheduling strategies for job placement and data movements. Hence, as we applied different policies into an use case, researchers can use it to study whether scheduling algorithms that have been

proposed two decades ago in desktop computing platforms, volunteer computing and computational grids [7, 6, 15] reviewing to cope with edge specifics.

This work also contributed with a descriptive analysis of the Qarnot Computing' logs, presenting its whole methodology step by step, which can be used as base for another similar studies. Moreover, it contributes with a first attempt of methodology to categorize distributions of temperatures, based on an analysis of variances.

One can find the whole experimental structure to the Part II in the *uga-master-thesis* repository on GitHub [3], and to the Part III in the *usp-tcc* repository on GitLab [4]. In both repositories there are more figures and all scripts utilized for experiments. Also, it was produced a technical [5] reports which describes and summarize the whole implementation of the Edge Simulator and the Qarnot use case.

1.6 Outline

The rest of this work is structured as follows. In the Part I: Chapter 2 presents related works. Chapter 3 presents the *Qarnot Computing* use case. In the Part II: Chapter 1 gives an overview of the Bastim/SimGrid toolkit and the extensions utilized. Chapter 2 describes how we simulated the use case. Chapter 3 presents concepts about job allocation and metrics to evaluate performance in the context of this thesis. Also depicts the algorithms developed based on the use case and its differences. Chapter 4 discusses analyses of the different scheduling strategies for the *Qarnot* platform. We also show investigations regarding jobs processing time and the data sets dependencies based on the simulation and the extracted logs. Finally, Chapter 5 concludes and and remarks future steps for the second part. In the Part III: Chapter 1 presents the methodology of the descriptive analysis and the smart heaters' categorizations. Section 2.1.3 presents the experiments results and discussions. And Chapter 3 concludes and remarks future steps for the third part. Finally, Chapter 1 presents general discussions and the acquires knowledge during this work.

[3] https://github.com/andersonandrei/uga-master-thesis

[4] https://gitlab.com/andersonandrei/usp-tcc

[5] https://hal.inria.fr/hal-02153203v4

— 2 —
State of the Art

This works related to several terms and concepts into the development of an edge simulated platform focused in the scheduling of HPC jobs. Hence we studied the state of the art of these concepts follow as Cluster, Grid and Cloud Computing, as well as IoT and the usage of Mobile devices. Regarding the allocation of HPC jobs, we will present some studies about this kind of job and its allocation process from different possible goals and views. Running these jobs in platforms as Cluster, Grid, Edge or Cloud, is visible the necessity to use metrics to evaluate the performance for such processes and allocation techniques. So, we will present some of these techniques as load balance of HPC jobs and the usage of virtualization and containers to manage this kind of platform. In addition, as the energy consumption is a very important problem related with all of these concepts, we will present some related works that handle it. Next we will present some recent statistics from scientific studies and others from the point of view of companies, where both emphasizes the emergency and the requirement of Edge Computing. Finally, we show show some related simulated platforms providing comparisons with Batsim, which is the simulator that we utilized and improved during this work.

2.1 Computing Platforms, Difference and Evolution

In a survey, Huang et.al [19] define the evolution of computing platforms into three phases that were followed over the years. The mode of the computing has been changed along the time, and Cloud Computing is a very important on in the current state. The following defines those evolution processes three steps:

1. The original mode which for processing, gathered all the tasks to large-scale processors.
2. The distributed tasks processing mode based on the Internet.
3. The Cloud Computing mode for immediate processing.

Hameed Hussain et.al [20], defined as HPC categories: Cluster, Grid and Cloud Computing platforms, which are conceptually similar. However, each of these categories have distinct features, and we emphasize the main distinctions as follows:

- On Clusters, the goal is to design an efficient computing platform that uses a group of computer resources integrated through hardware, networks, and software to improve the performance and availability of a single computer resource, such that:

  - A modern one is made up of a set of commodity computers that are usually restricted to a single switch or group of interconnected switches within a single virtual local-area network (VLAN). In addition to executing compute-intensive applications, cluster systems are also used for replicated storage and backup servers that provide essential fault tolerance and reliability for critical parallel applications.
  - Allows extensions by incorporating load balancing, parallel processing, multi-level system management, and scalability methodologies.

- On Grid, the computing concept is based on using the Internet as a medium for the wide spread availability of powerful computing resources as low-cost commodity components. Computational grid can be thought as a distributed system of logically coupled local clusters with non-interactive workloads that involve a large number of files. Emphasizing that grids tend to be more loosely coupled, heterogeneous, and geographically dispersed, makes grid different from conventional HPC systems, such as cluster.

- On Cloud is found a recent model for Information Technology (IT) services based on the Internet, that typically involves provision of dynamically scalable and often virtualized resources over-the-Internet. Typical Cloud Computing providers deliver common business applications on line that are accessed through web service, and the data and software are stored on the servers. In addition, the Cloud Computing systems are difficult to model with resource contention (competing access to shared resources). Many factors, such as the number of machines, types of applications, and overall workload characteristics, can vary widely and affect the performance of the system.

Focusing on Grid Computing, Qureshi et.al [31], present it as a platform for virtual organizations and computing environments that was introduced in 1990s, which:

- Provides low-cost intelligent methodologies for sharing data and resources such as computers, software applications, sensors, storage space, and network bandwidth due to the necessity of reliable, pervasive, and high computing power.

- Depending on factors as operating system, amount of memory, CPU speed, number of resources, architecture types and so on, Grids can be generally classified as homogeneous or heterogeneous.

Several surveys present the definition of Cloud Computing as composed of three kind of services [29, 20, 31]:

- Cloud Software as a Service (SaaS), which cloud providers offer software running on a cloud infrastructure.

- Cloud Platform as a Service (PaaS), which the cloud platform offers an environment for development and deployment of applications.

- Cloud Infrastructure as a Service (IaaS), which cloud providers manage computing resources such as storing and processing capability.

In addition, different deployment models have been adopted based on their variation in physical location, distribution and services, classifying Clouds among:

- Private, which is restricted for management and usage of predefined users.
- Public or Hosted, which is open to the public, generally usually charged on a pay-per-use.
- Community, which is available to specific group of people or community in order to share resources and services.
- Hybrid, which is a combination of the other three types.

Then, the applicability of Cloud Computing has been studied. Luiz Bittencourt et.al [9] depict how the expansion of Internet of Things (IoT), is affecting the way to use Cloud Computing, storing, processing and producing information and knowledge as a result. It also discuss how in one hand, the wide adoption of Cloud Computing is a consequence of a fast time-to-market for many types of applications due to the paradigm's flexibility and reduced or null initial capital expenditures, on the other hand, this same wide adoption has exposed some limitations of the paradigm in fulfilling all requirements of some classes of applications, such as real-time low latency, and mobile applications. Due to the fact that centralized cloud data centers are often physically and/or logically distant from the cloud client, the communication and data transfers traverses multiple hops, which introduces delays and consumes network bandwidth of edge and core networks. As a combination of the ability of run small, localized applications at the edge with the high-capacity from the cloud, it presents the fog computing as emerged paradigm that can support heterogeneous requirements of small and large applications through multiple layers of a computational infrastructure that combines resources from the edge of the network as well as from the cloud.

In addition, Mao et al. [22] present how Mobile devices tend to grow in terms of usability and processing of data, implicating the decentralization from the Cloud's presence. This survey says that the last decade has seen Cloud Computing emerging as a recent paradigm of computing such that a vision is the centralization of computing, storage and network management in the Clouds, referring to data centers, backbone IP networks and cellular core networks. But, in recent years, it has been changed due to the Clouds being increasingly moving towards the network edges. Mao *et al. [22]* present the estimation that tens of billions of Edge devices will be deployed in the near future, and their processor speeds are growing exponentially, following Moore's Law. Harvesting the vast amount of the idle computation power and storage space distributed at the network edges can yield sufficient capacities for performing computation-intensive and latency-critical tasks at mobile devices. The same survey presents the Mobile Edge Computing (MEC) as a computation provider at mobile devices considering the proximate access, that is widely agreed to be a key technology for realizing various visions for next-generation Internet, such as Tactile Internet (with millisecond-scale reaction time) and Internet of Things (IoT). Also, Y. Mao et.al say :

- It shows implications from the different techniques of implementation of MEC, which are network functions virtualization (NFV), information-centric networks (ICN) and software-defined networks (SDN).
- It presents the Fog Computing as a propose of Cisco as a generalized form of MEC where the definition of edge devices gets broader, Fog Computing and Networking are overlapping the terminologies with MEC.

2.2 Resource Allocation Techniques and Metrics

Some of the works referenced in the previous section also present the challenges for resource allocation from the Cloud, Grid and Edge Computing. Hence, in this section we present some solutions and techniques.

Huang *et al.* [19] affirm that to make appropriate decisions when allocating hardware resources to the tasks and dispatching the computing tasks to resource pool has become the main issue in Cloud Computing. As Cloud Computing has its own features, the resource allocation policies and scheduling algorithms for the other computing technologies are unable to work under these conditions. For that reason there is not an uniform standard for job scheduling in cloud and then it is an important component in this context.

S. M. Parikh [29] points that the management of flexible resources allocation is a problem emerged in this context, due to heterogeneity in hardware capabilities, workload estimation and a variety of services, also as the the maximization of the profit for cloud providers and the minimization of cost for cloud consumers.

According to Hussain *et al.* [20] the resource management mechanism determines the efficiency of the used resources and guarantees the Quality of Service (QoS) provided to the users. Therefore, the resource allocation mechanisms are considered a central theme in HPC. QoS resource management and scheduling algorithms are capable of optimally assigning resources in ideal situation or near-optimally assigning resources in actual situation, taking into account the task characteristics and QoS requirements. In addition it presents common attributes among the HPC categories, such as size, network type, and coupling.

In Grid platforms as Qureshi, Muhammad Bilal et.al [31] present, a Grid resource can be defined as an entity that needs to carry out an operation by an application such that each application in Grid environment competes for various resources according to application needs. This way resource allocation, mechanisms play an important role in allocating the most appropriate resources to applications. The mechanisms perform the allocation of tasks to the resources in order to ensure QoS to the application according to the user requirements. Ressource allocation mechanisms provide two basic Grid services:

- Resource monitoring, which regularly monitors resource performance, capability, usage and future reservations, including processors, disks, memories,and channel bandwidths.
- Resource scheduling, which retrieves the information from a) and decides on the allocation of the application to the underlying resources. There are several goals to conduct the RA process as reduce makespan, power minimization and energy efficiency improvement, reduction of task completion time or the amount of data transfer, among others.

Feitelson, Dror G. [16] says that the root cause for convergence problems is variability in the workloads. Therefore, it characterizes the variability in the runtime and arrivals of workloads observed on different systems, and in models based on them. The first metric dealt with is the response time. It defines "response time" to mean the total wall clock time from the instant at which the job is submitted to the system, until it finishes its run. This can be divided into two components: the running time, denoted by Tr, during which the job is actually running in parallel on multiple processing nodes, and the waiting time, denoted by Tw , in which it is waiting to be scheduled or for some event such as I/O. The waiting time itself can also be used as a metric, based on the assumption that Tr does not depend on the scheduling. Obviously,

a lower bound on the response time of a given job is its running time. As the runtime of jobs have a very large variance, so must the response time.

It was therefore suggested that a better metric may be the slowdown (also called "expansion factor" or stretch), which is the response time normalized by the running time: $slowdown = (Tw + Tr)/Tr$. Thus if a job takes twice as long to run due to system load, it suffers from a slowdown factor of 2, etc. This is expected to reduce the extreme values associated with very long jobs, because even if a week-long job is delayed for a whole year the slowdown is only a factor of 50. Moreover, slowdown is widely perceived as better matching user expectations that a job's response time will be proportional to its running time. It affirms that the slowdown metric is that it over-emphasizes the importance of very short jobs. For example, a job taking 100 ms that is delayed for 10 minutes suffers from a slowdown of 6000, whereas a 10-second job delayed by the same 10 minutes has a slowdown of only 60. To avoid such effects, Feitelson et al. have suggested the "bounded-slowdown" metric. The difference is that for short jobs, this measures the slowdown relative to some "interactive threshold", rather than relative to the actual runtime. Denoting this threshold by , the definition is $bounded - slowdown = max\{(Tw + Tr)/max\{Tr, \}, 1\}$. In addition, Aida, Kento [4] deals with the investigation of the effect of the job size on the scheduling performances, it characterizes and performs experiments showing how does the processor utilization and the bounded slowdown are affected in that context. In details, the processor utilization is the percentage that processors are busy over entire simulation. The slowdown ratio (SR), shows normalized data for mean response time. For instance, Aida, Kento [4] supposes that 10000 jobs were executed in an experiment. The mean response time of these 10000 jobs was 5 hours, and their mean execution time on processors was 2 hours. Then, the slowdown ratio is 2.5.

2.3 Applicability, Load Balance, Energy Consumption Reduction and Other

Different techniques for several goals have been applied in the context of Cloud and Edge Computing. The usage of containers is one technique presented by C. Pahl and B. Lee [28] that introduces the Cloud Computing as a centralized, large-scale data centers to a more distributed multi-cloud setting comprised of a network of larger and smaller virtualized infrastructure runtime nodes, also referred to as edge clouds, Edge Computing or fog computing. It is focused on the virtualization as a form to reach the network and allow Internet-of Things (IoT) infrastructures to be integrated. As a challenge resulting from distribution, it affirms the necessity of more lightweight solutions than the current virtual machine (VM)-based virtualization technology. Virtual machines (VMs) have been at the core of the compute infrastructure layer providing virtualized operating systems. It investigates containers, which are a lightweight virtualization concept, i.e., less resource and time consuming. VMs and containers are both virtualization techniques, but solve different problems. Containers are a solution for more interoperable application packaging in the cloud and should therefore address the PaaS concerns.

Load balancing algorithm is another example, M. Randles et.al [32] identify it as major concern to allow Cloud Computing to scale up to increasing demands. It presents three potentially viable methods for load balancing in large scale Cloud systems. The first one is a nature-inspired algorithm may be used for self-organization, achieving global load balancing via local server actions. The second one by a self-organization that can be engineered based on

random sampling of the system domain, giving a balanced load across all system nodes. The third one, by a restructure to optimize job assignment at the servers.

Since the execution of HPC jobs produce a huge energy consumption, one other target that has been studied is how to reduce this energy consumption. Jie Meng et.al [24] presents that has been reported the worldwide data center electricity consumption increased by 56% from 2005 to 2010, which accounted for 1.3% of the total electricity use. A recent review shows that for every dollar spent on power of data center computing equipments, another dollar is spent on data center cooling infrastructures, which translates to an energy cost reaching up to millions of dollars and cooling costs reaching close to half of the overall energy cost. Thus, it manages simulations in order to study cooling and energy efficiency in this context.

The Batsim/ SimGrid toolkit also includes an energy plugin to keep track of temperature for similar reasons [33, 18], which will be discussed a lit bit more in next sections.

In addition, the Qarnot Computing proposal is direct related with this context, which will be discussed in the next sections, but could be find detailed information in [27].

2.4 Motivation, Companies Usage and Recent Statistics

The work [34] presents in details how Cloud Computing has been used and how it has been requiring Edge Computing as a recent paradigm. It takes into account the aspect and impact commercial from these computing platforms providing a very good panoramic view of the context, it affirms that nascent technologies and applications for mobile computing and the Internet of Things (IoT) are driving computing toward dispersion. For them Edge Computing is a recent paradigm in which substantial computing and storage resources—variability referred to as cloudlets, micro data centers, or fog nodes are placed at the Internet's edge in close proximity to mobile devices or sensors. It shows that industry investment and research interest in Edge Computing have grown dramatically in recent years. Nokia and IBM jointly introduced the Radio Applications Cloud Server (RACS), an Edge Computing platform for 4G/LTE networks, in early 2013. It affirms that the following year, a mobile Edge Computing standardization effort began under the auspices of the European Telecommunications Standards Institute (ETSI). The Open Edge Computing initiative (OEC; [1]) was launched in June 2015 by Vodafone, Intel, and Huawei in partnership with Carnegie Mellon University (CMU) and expanded a year later to include Verizon, Deutsche Telekom, T-Mobile, Nokia, and Crown Castle. This collaboration includes creation of a Living Edge Lab to gain hands-on experience with a live deployment of proof-of-concept cloudlet-based applications. Organized by the telecommunications industry, the first Mobile Edge Computing Congress (tmt.knect365.com/mobile-edge-computing) convened in London in September 2015 and again in Munich a year later. The Open Fog Consortium [2] was created by Cisco, Microsoft, Intel, Dell, and ARM in partnership with Princeton University in November 2015, and has since expanded to include many other companies. The First IEEE/ ACM Symposium on Edge Computing (conferences.computer.org/SEC) was held in October 2016 in Washington, DC.

It is possible to find examples of software as a service (SaaS) instances, such as Google Apps, Twitter, Facebook, and Flickr, that have been widely used in our daily life [36] [37].

[1] openedgecomputing.org

[2] www.openfogconsortium.org

Moreover, scalable infrastructures as well as processing engines developed to support cloud service are also significantly influencing the way of running businesses such as, Google File System, MapReduce, Apache Hadoop, Apache Spark, and so on. In addition, it affirms with recent statistics that with IoT, we will arrive in the post-cloud era, where there will be a large quality of data generated by things that are immersed in our daily life, and a lot of applications will also be deployed at the edge to consume these data. By 2019, data produced by people, machines, and things will reach 500 zettabytes, as estimated by Cisco Global Cloud Index, however, the global data center IP traffic will only reach 10.4 zettabytes by that time. By 2019, 45% of IoT-created data will be stored, processed, analyzed, and acted upon close to, or at the edge of, the network. Finally, they raise the following issues: *Why Do We Need Edge Computing ?*

- Push from Cloud services: putting all the computing tasks on the cloud has been proved to be an efficient way for data processing since the computing power on the cloud outclasses the capability of the things at the edge. However, compared to the fast developing data processing speed, the bandwidth of the network has come to a standstill. With the growing quantity of data generated at the edge, speed of data transportation is becoming the bottleneck for the cloud-based computing paradigm. It examples, about 5 Gigabyte data will be generated by a Boeing 787 every second, but the bandwidth between the airplane and either satellite or base station on the ground is not large enough for data transmission. It considers an autonomous vehicle as another example, one Gigabyte data will be generated by the car every second and it requires real-time processing for the vehicle to make correct decisions. If all the data needs to be sent to the cloud for processing, the response time would be too long. Not to mention that current network bandwidth and reliability would be challenged for its capability of supporting a large number of vehicles in one area. In this case, the data needs to be processed at the edge for shorter response time.

- Pull From IoT many kinds of electrical devices will become part of IoT, and they will play the role of data producers as well as consumers, such as air quality sensors, LED bars, streetlights and even an Internet-connected microwave oven. It is safe to infer that the number of things at the edge of the network will develop to more than billions in a few years. Thus, raw data produced by them will be enormous, making conventional cloud computing not efficient enough to handle all these data. This means most of the data produced by IoT will never be transmitted to the cloud, instead it will be consumed at the edge of the network.

- Change from data consumer to producer: in the Cloud Computing paradigm, the end devices at the edge usually play as data consumer, for example, watching a YouTube video on a your smart phone. However, people are also producing data nowadays from their mobile devices. The change from data consumer to data producer/consumer requires more function placement at the edge. For example, it is very normal that people today take photos or do video recording then share the data through a cloud service such as YouTube, Facebook, Twitter, or Instagram. Moreover, every single minute, YouTube users upload 72 h of new video content; Facebook users share nearly 2.5 million pieces of content; Twitter users tweet nearly 300 000 times; Instagram users post nearly 220 000 new photos. However, the image or video clip could be fairly large and it would occupy

a lot of bandwidth for uploading. In this case, the video clip should be demised and adjusted to suitable resolution at the edge before uploading to cloud. Another example would be wearable health devices. Since the physical data collected by the things at the edge of the network is usually private, processing the data at the edge could protect user privacy better than uploading raw data to cloud.

Also, *what are the benefits of Edge Computing?*

- In Edge Computing we want to put the computing at the proximity of data sources. This have several benefits compared to traditional cloud-based computing paradigm. Here we use several early results from the community to demonstrate the potential benefits. Researchers built a proof-of-concept platform to run face recognition application in, and the response time is reduced from 900 to 169 ms by moving computation from cloud to the edge. Moreover, the energy consumption could also be reduced by 30%–40% by cloudlet offloading. Clone cloud in combine partitioning, migration with merging, and on-demand instantiation of partitioning between mobile and the cloud, and their prototype could reduce 20× running time and energy for tested applications.

2.5 Related Simulation Tools

We described in this work a novel simulation tool for easily designing and testing scheduling strategies on Edge Computing platforms, built on top of Batsim. It was motivated by the huge effort of building a new simulator using adequate tools for modeling the processing and memory units and the network topology.

We discussed briefly below the main competitors and argument for this simulator. Some simulators have constraints that would prevent us to correctly simulate a platform such as the *Qarnot*'s one. For example, EmuFog[23] does not support hierarchical fog infrastructures, whereas *Qarnot* infrastructure is inherently hierarchical. Other simulators such as iFogSim[17], EdgeCloudSim[39] and IOTsim[42], are simulation frameworks that enable to simulate fog computing infrastructures and execute simulated applications on top of them. The two closest simulators to the presented one is a) the CloudSim, widely used to validate algorithms and applications in different scientific publications, however is based on a top-down viewpoint of cloud environments. And b) this one is related to other very close work on the literature [24] where are implemented evaluation models and allocation optimization methods in SST, the Structural Simulation Toolkit. The SST is an architectural simulation framework designed to assist in the design, evaluation, and optimization of HPC architectures and applications. It is developed by Sandia National Laboratories to evaluate the performance of computer systems ranging from small-scale single-chip processors to large-scale parallel computing architectures. It was used for evaluating an optimization algorithm managing real-world parallel workloads, as well as the implementations of job scheduler and allocation algorithms in SST.

The Batsim is built on top of SimGrid, which has been validated in many publications [2] and allows finer-grained simulations, as explained in Section 1.1.

2.6 Future Remarks in Edge and Cloud Computing

According to Shi et al. [36], [37] there will be 50 billion objects connected to the Internet by 2020, as predicted by Cisco Internet Business Solutions Group. Some IoT applications might require very short response time, some might involve private data, and some might produce a large quantity of data which could be a heavy load for networks. They conclude that Cloud computing is not efficient enough to support these applications due to the growth of data production at the edge of the network. Therefore, it would be more efficient to also process the data at the edge of the network, close to where it is generated. And remark that previous work such as micro data center, cloudlet, and fog computing have been introduced to the community because cloud computing is not always efficient for data processing when the data is produced at the edge of the network.

Finally, several possibilities of next steps that could be taken in the future of Edge and Cloud Computing are discussed by Bittencourt et al [9]:

1. Fog and 5G for IoT: while the first 5G deployments are expected in the next couple of years, several challenges remain in how these deployments will support IoT services integrated with cloud and fog computing.

2. Serverless Computing: microservices management throughout the IoT-Fog-Cloud hierarchy presents challenges associated to the movement of services among IoT, fog, and cloud devices. The automatic adaptation of the execution of microservices must consider deployment location and context, but should also not neglect resource constraints that may exist at each level of the fog.

3. Resource Allocation and Optimization: The composition of devices in the IoT-Fog-Cloud continuum brings novelties as the heterogeneity of devices and applications reach unprecedented levels, then optimization in resource allocation becomes more challenging.

4. Energy Consumption: The proliferation of IoT devices and the ever increasing rate of data produced are increasing pressures on energy consumption. One should expect that such pressures will have to be addressed at both hardware and software levels as well as their interplay.

5. Data Management and Locality: There are several open issues related to data management and locality in IoT-Fog-Cloud computing systems. First and foremost, these systems are typically composed of a broad set of heterogeneous communication technologies such as cellular, wireless, wired, and radio frequency. This means that the systems orchestrator has to be able to handle distinct underlying networks as well as different addressing schemes.

6. Applying Federation Concepts to Fog Computing and IoT: Federations will be widely used in many different application domains. The outstanding challenge here is how can federation capabilities be best applied in fog and IoT environments? The easiest answer is to simplify the deployment and governance models to be used.

7. Trust Models to Support Federation in Fog and IoT Environments: Identity and trust are the cornerstones of federation management. While a number of methods exist for establishing identity and trust, the only feasible methods are based on cryptographic

methods. An inherent property of IoT environments, though, is that the closer to the edge one gets, the more resource-constrained the devices will become.

8. Orchestration in Fog for IoT: Despite recent developments in the area of fog orchestration for the Internet of Things, there are still several open issues that need to be addressed. First and foremost, privacy must be tackled in accordance to the European Union General Data Protection Regulation as well as similar regulations being enforced worldwide.

9. Business and Service Models: While cloud computing has been offering a variety of business and service models through the years, it is not clear yet if fog computing can simply incorporate the cloud models or if new business or service models would be feasible.

10. Mobility: Efficiently allocating resources for mobile users is a challenge in fog computing. Users and devices mobility patterns are an important aspect to provide proper service when offloading to the fog occurs. Dealing with a large set of mobile users with diverse applications and requirements is a highly dynamic scenario, which makes resource management challenging.

11. Urban Computing: Although several research efforts related to urban computing have been performed recently, it is possible to find open issues and opportunities for studying cities and societies using location-based social networks (LBSN) data.

12. The Industrial Internet of Things: Designing software that exploits the Industrial Internet of Things constitutes a "system of systems" challenge. Taking into account the whole Iot-Fog-Cloud continuum, addressing the complexity of this challenge will require frameworks that enable interoperability but are also able to cope with varying and possibly conflicting user and system requirements.

— 3 —
Case study: the Qarnot Computing platform

We present in this section the platform of the *Qarnot Computing* company, which serves as our case study.

3.1 Infrastructure Overview

Qarnot Computing has been incorporated in 2010 to develop a disruptive solution able to turn IT heat waste into a viable heating solution for buildings. The infrastructure is distributed in housing buildings, offices and warehouses across several geographical areas in France and Europe, in each situation directing the heat waste produced by computations to heat air and water for the building. As of writing this manuscript, the whole platform is composed of about 1,000 computing devices(*QRads*, the Qarnot Radiators, or Smart heaters) hosting about 3,000 diskless machines. The diskless machines have access to some storage area present on the deployment site (*QBox*), shared as NFS (Network File Systems) through a LAN (Local Area Network). From now one, we will use machines, computing devices, Smart Heaters and QRads as synonymous.

In a typical configuration a computing machine has a 1 Gbps uplink to a common switch, which then has up to 40 Gbps uplink to the *QBox*. The latency between a computing machine and its storage area is of the order of 1 ms. The various deployment sites are connected to the Internet using either a public or enterprise ISP, with characteristics varying from 100 Mbps to 1 Gbps symmetric bandwidth to the Internet, and about 10 ms latency to French data centers used by *Qarnot* to host control and monitoring infrastructure, central storage services, and gateways to its distributed infrastructure.

Qarnot deploys high performance computing hardware and storage capacity to buildings, which makes it a fitting infrastructure to locally gather and process data that is generated at the edge of the network (for instance on smart buildings). One objective of the edge simulator is to evaluate evolution's of the *Qarnot* architecture to handle such local use-cases. It will allow investigating the edge infrastructure dimensioning as well as the optimization of the local data and processes placement with regard to the global ones. This can reduce global data movements, enable buildings to be autonomous in terms of IT and to handle Internet connectivity loss gracefully.

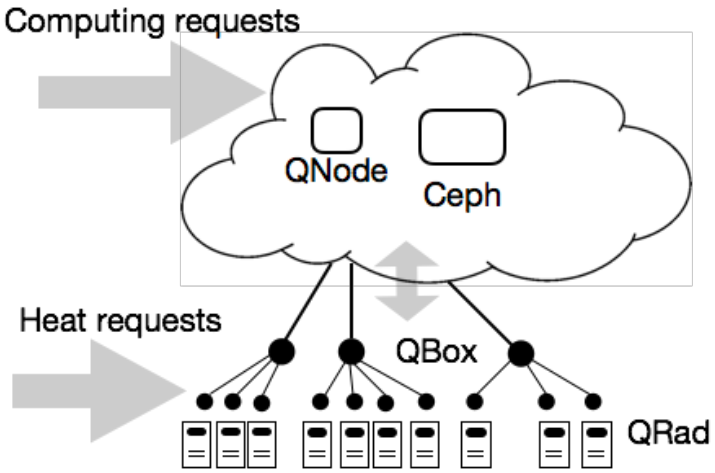


Figure 3.1 – Scheme of the Qarnot platform.

3.2 Platform Terminology

The job and resource manager of the *Qarnot* platform, named *Q.ware*, is based on a hierarchy of 3 levels as shown in Figure 3.1: the *QNode*-, the *QBox*- and the *QRad*-level.

The QNode is the root node, a "global" server that takes placement decisions for the whole platform. It can be viewed as a load balancer for the platform. Connected to this QNode there are the QBoxes, which are "local" servers in smart buildings that take scheduling decisions locally on their own computing nodes. Each QBox is in charge of a set of computing nodes, the QRads, which are composed of one or several motherboards, denoted by *QMobos*.

Moreover, a centralized storage server, the *CEPH*, is present at the QNode-level while each QBox has its own local storage disk. From a physical point of view, the QNode and CEPH are on the cloud while QBoxes are distributed over smart buildings of several cities. QRads among a building are distributed in different rooms.

The *Qarnot* platform receives two types of user requests: requests for computing and requests for heating. The computing requests describe the workload to be executed on the platform. They are made by users that first upload input data needed to execute their jobs (named Docker [1] image either to the centralized server or the Docker Hub. Then, they submit the QTasks to the QNode. A QTask can be decomposed in a bag of several *instances* that share the same Docker image and data dependencies, but with different command arguments. This can be used for example to process each frame of a given movie, with one frame or a range of frames per instance.

The heating requests are made by inhabitants that can turn on and off the smart heaters in their homes, or set a target temperature for rooms to be reached as soon as possible. Since the computing units in a smart heater are unavailable when cooling is necessary, and are available otherwise, such changes increase the heterogeneity challenges of an edge infrastructure: the machines does not simply appear or disappear but also varies according to the heating needs.

[1] Docker is a platform for developers and sysadmins to build, run, and share applications with containers. The use of containers to deploy applications is called containerization. Containers are not new, but their use for easily deploying applications is. See https://docs.docker.com/get-started/.

3.3 Current Workflow

Whenever QTasks are submitted on the platform all the data dependencies should be uploaded to the CEPH. To be executed, these QTasks have to be scheduled to the QBoxes and then scheduled onto QMobos through two scheduling steps.

The first step takes place at the QNode-level. The QNode greedily dispatches as many instances of the QTasks ordered by priority on QBoxes, depending on the number of QMobos available for computation on each QBox.

The second step takes place at the QBox-level. Upon receiving instances of a QTask, the QBox will select and reserve a QMobo for each instance and fetch from the CEPH each missing data dependency before starting instances.

Notice that, at all times, a *FrequencyRegulator* runs on each QRad to ensure that the ambient air is close to the target temperature set by the inhabitant, by regulating the frequencies of the QMobos and completely turning off a QRad if it is too warm. Moreover, whenever there is no computation performed on the QMobos while heating is required, some "dummy" compute-intensive programs are executed to keep the QRad warm. Figure 3.2 summarizes the execution flow of a QTask within the *Qarnot* platform.

As mentioned, QTasks to be executed on this platform are submitted to the QNode and all the data dependencies are uploaded to the CEPH (steps 01 and 02). To be executed, these QTasks, along with their data dependencies, have to be sent to the QBoxes and then scheduled onto QMobos. Every 30 seconds, the QBoxes send information about the state of their disk, QRads and QMobos to the QNode (steps 03 through 05), in particular, the number of QMobos available for computation and the free storage space are reported.

A first scheduling process is made at the QNode-level to dispatch instances of the QTasks to the QBoxes (steps 06 and 07). The QNode tries to dispatch, for each QTask taken by priority, as many instances as possible onto QBoxes, with respect to the number of available QMobos and storage space left on the QBox disks.

Upon receiving instances of a QTask, the QBox will reserve for each instance a QMobo from the warmest QRad for the case of low priority QTask, and a QMobo from the coolest QRad in the case of high priority (step 08). This distinction is made to keep more QMobos available in case high priority QTasks are sent to the QBox in the near future. The QBox then checks whether the Docker image and other data dependencies for these instances are on disk and fetches any missing data from the CEPH (steps 09 and 10).

Once all data transfers are completed for this QTask, the reserved QMobos are rebooted and the instances are started (steps 11 through 14). When the instance completes, its output is uploaded to the CEPH and the QNode is notified of the instance completion (steps 15 through 17). Finally, the queue of QTasks is updated and if an instance of the same QTask can be directly dispatched, it is sent to the QBox and the execution starts immediately, without rebooting the QMobo (steps 18 and 19).

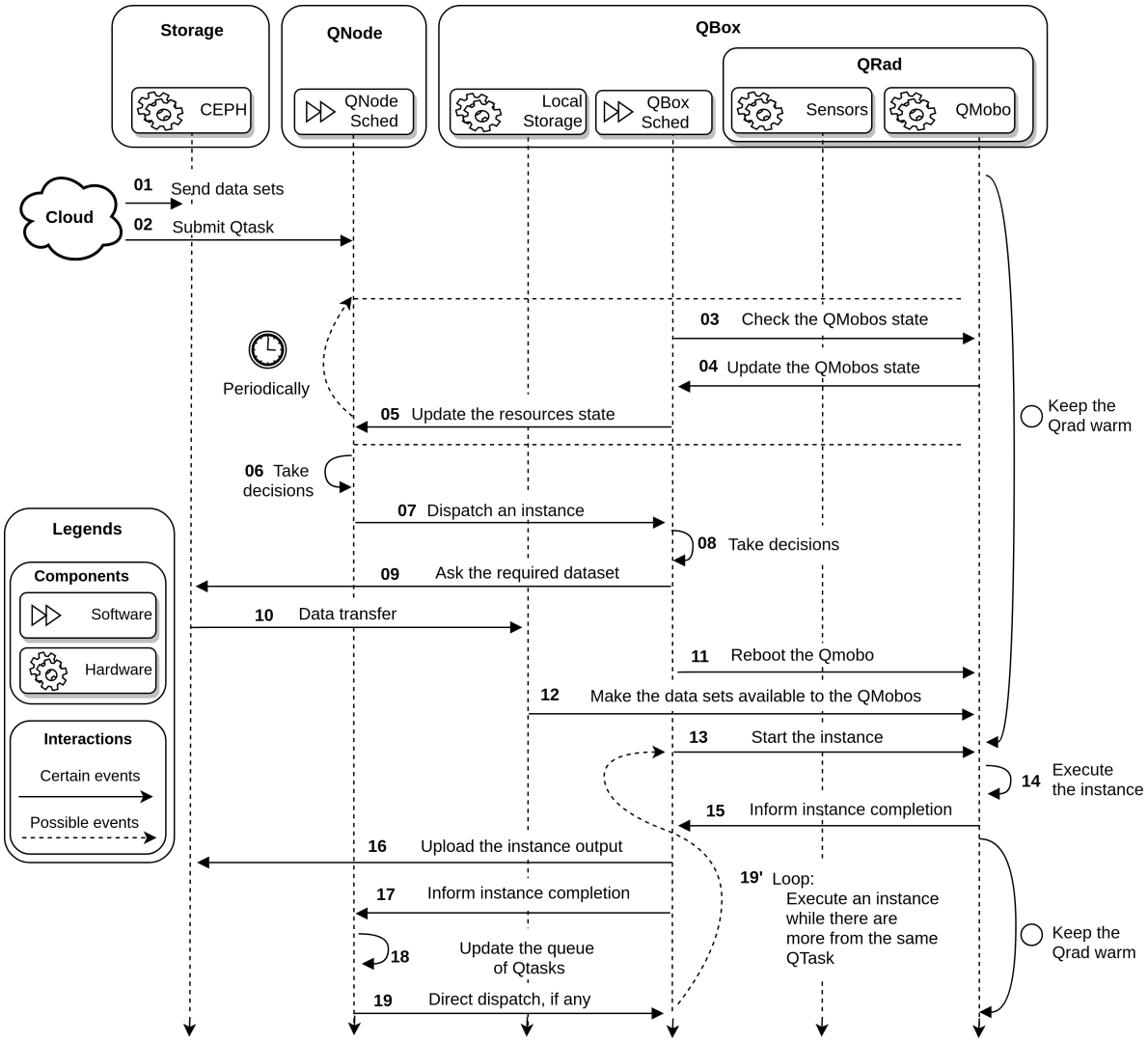


Figure 3.2 – Execution flow of a QTask on the Qarnot platform.

Part II

A Simulator Edge Platforms

— 1 —

A Dedicated Scheduling Simulator for Edge Platforms

We chose to Batsim/SimGrid instead of available fog/edge simulators [17, 39, 42] for several reasons:

- Batsim has been specially designed to test and compare batch scheduling policies in distributed infrastructures. In other words, the design of Batsim enforces researchers to use the same abstractions and thus, favor straightforward comparisons of different strategies, even if they have been implemented by different research groups.
- The accuracy of the internal models (computation and network) of SimGrid has been already validated.
- Batsim provides a Python API that makes the development of a scheduling strategy simple.

Released in 2017, Batsim [14] delivers a high-level API to facilitate the development of batch scheduling algorithms that are simulated on top of SimGrid [12], the well-proven simulator toolkit for distributed infrastructures. Thus, it is possible to rely on high-level tools that have been proposed and already validated. Some parts have been customized to reflect the edge specifics, especially the decision processes, which is the main goal of this work, and the others (Storage Controller, Injector and Qarnot Extractor) have been developed by the workgroup at the same time. In this chapter is described all of them.

1.1 Operational Components

In this section we discuss the role of the different components, namely SimGrid, Batsim, the *decision process* and their interactions.

1.1.1 SimGrid

SimGrid [12] is a generic simulator framework that enables simulation of distributed systems. Performing simulations with SimGrid requires (i) writing a platform specification, (ii) formatting workload input data, and (iii) interfacing the program to evaluate.

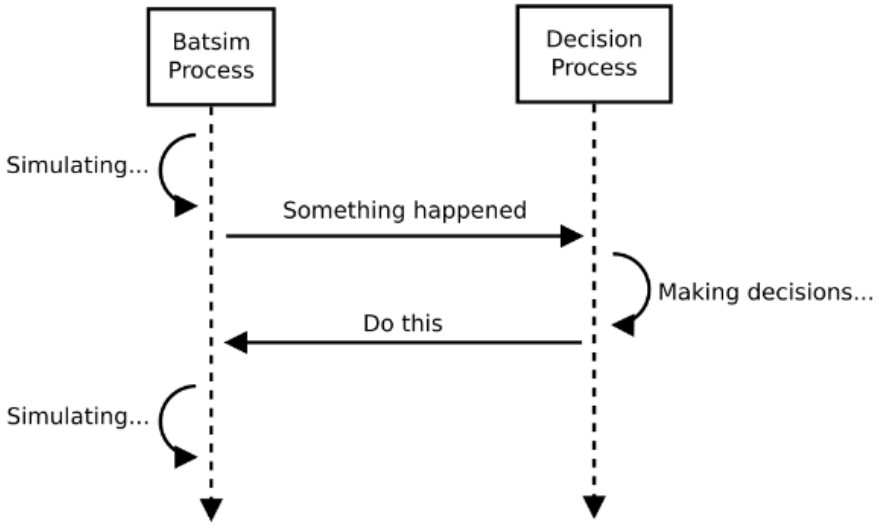


Figure 1.1 – Batsim and decisions maker interaction.

The choice of using SimGrid as the main engine for Batsim is mainly due to its relevance in terms of performance, as well as its validity that has been backed-up by many publications [2]. Moreover, it enables the description of complex infrastructures, such as hierarchical ones, that are composed of many interconnected devices with possibly highly heterogeneous profiles. Finally, the injection of external events on demand, such as machines connecting or disconnecting on the network, has allowed the easy simulation of complex systems such as fog/edge infrastructures.

1.1.2 Batsim and the Decision Process

Batsim [14] is an infrastructure simulator for jobs and I/O scheduling, built on top of SimGrid, to help the design and analysis of batch schedulers. Batch schedulers, or Resource and Jobs Management Systems, are systems in charge of managing resources in large-scale computing centers, notably by scheduling and placing jobs. Batsim allows researchers to simulate the behavior of a computational platform on which a workload is executed according to the rules of a decision process. It uses a simple event-based communication interface. An event here represents some action (i.e, a job submission/conclusion, an user request, etc.). As soon as an event occurs, Batsim stops the simulation and reports what happened to the *decision process*.

The decision process embeds the actual scheduling code to be evaluated. In other words, in order to simulate a given scheduling algorithm, an experimenter has to implement this decision process. Comparing different algorithms consists in switching between different decision processes, which is straightforward.

Figure 1.1 illustrates the interaction between Batsim and the decision process. The decisions process reacts to the simulation events received from Batsim, takes decisions according to a given scheduling algorithm, and drives the simulated platform by sending back its decisions to Batsim. In this work, we used Batsim's Python API to implement the decision process, which provides functions to ease the communication with Batsim.

More details on Batsim and SimGrid mechanisms can be found on Chapter 4 of Millian Poquet's manuscript [30].

1.2 Extensions

There are a couple of extensions that have been developed to deal with edge challenges, such as machines getting unavailable during a job execution, on networks failures. In this section, we present the ones that are already available, the events injector and the storage controller. Modifications made in Batsim[1] and its Python API[2] for this work are available in a separate branch of their main repository.

1.2.1 External Events Injector

To simulate the execution of an edge infrastructure, which by essence is subject to very frequent unexpected or unpredictable changes, Batsim offers the opportunity to inject external events on demand. Those events impact the behavior of the platform during the simulation and thus the choices of the scheduling strategy. For example, one would be interested in studying the behavior and resilience of a scheduling policy when a range of machines may become unexpectedly unavailable for a period of time, due to a failure or action occurring at the edge (from a local user).

The mechanism we implemented replays external events that occurred at a given time. When an event occurs it is handled by the main process of Batsim, that updates the state of the platform and the simulation, and then forwarded to the decision process.

An event is represented as a JSON object that contains two mandatory fields: a *timestamp*, which indicates when the event should occur, and *type*, the type of the event. Then, depending on the type of event, other fields can complement the event description, such as the name of the unavailable resource for example, the new value of an environment parameter such as the network bandwidth, or anything that is of interest to the decision process. External events are injected in Batsim by one of its internal processes, which reads the list of events from an input file containing one of the above described JSON objects per line.

This event injection mechanism is generic by the concept: users can define their own types of events and associated fields, which will be forwarded to the decision process without any modification in the code of Batsim.

1.2.2 Storage Controller

The Storage Controller is a Python module that exposes multiple functions to the scheduler in order to manage the storage entities as well as the data transfers. In order to give the scheduler reliable information, it keeps track in real-time of the platform status, the on-going data transfers, the available resources, etc. It also manages all aspects related to caching policies, while offering advanced features such as speculation.

[1] https://gitlab.inria.fr/batsim/batsim/tree/temperature
[2] https://gitlab.inria.fr/batsim/pybatsim/tree/temperature

— 2 —
Simulated Platform

In this chapter we present how the *Qarnot* platform was modeled and we explain how the required inputs were instantiated to be used on the Qarnot simulated platform. Then, it is described the content of each file given as input and how they are generated.

2.1 Qarnot to Batsim/SimGrid Abstractions

Figure 2.1 depicts the real and the simulated platform. Each QMobo of the platform is simulated as a SimGrid host (a machine which performs computation) as they are the only computing units of the platform. QMobos belonging to the same QRad are aggregated in the same SimGrid zone (a group of components managed by the SimGrid simulations), as well as QRads of the same QBox, and all QBoxes of the QNode. The management of storage spaces is done by adding special hosts that handle the *storage* role. Thus, in each QBox zone there is one additional storage host for the QBox disk. Similarly, there is one storage host in the QNode zone to represent the CEPH. For the computing requests, each instance of a given QTask can run independently of the others, so we transcribed each instance as one Batsim job, with the same data-set dependencies and submission time for instances belonging to the same QTask. Regarding the heating requests, each change of the target temperature of a QRad is simulated as an external event injected in the simulation, as well as when a QRad was turned off for being too warm.

2.2 Workflow

In order to simulate the behavior presented in Section 3.1 the schedulers of the QNode- and QBox-level were implemented in Python and both live in the same process, along with the Storage Controller. Through the platform translation shown in Figure 2.1, the Qarnot platform has been simulated as shown in Figure 2.2, which illustrates in detail how the abstraction between the Qarnot and the simulated platform is performed and it is possible to see fewer components for the same workflow since Batsim is in charge of all physical and execution process.

The simulated platform considers 3 main classes: the QNodeSched, QBoxSched and StorageController. Each one can communicate with the Batsim process delivering or receiving information by messages.

We illustrate the simulated platform workflow in Figure 2.2. First, Batsim receives as input and loads the events, platform, workload and the data sets description (steps 01 and 02). Then,

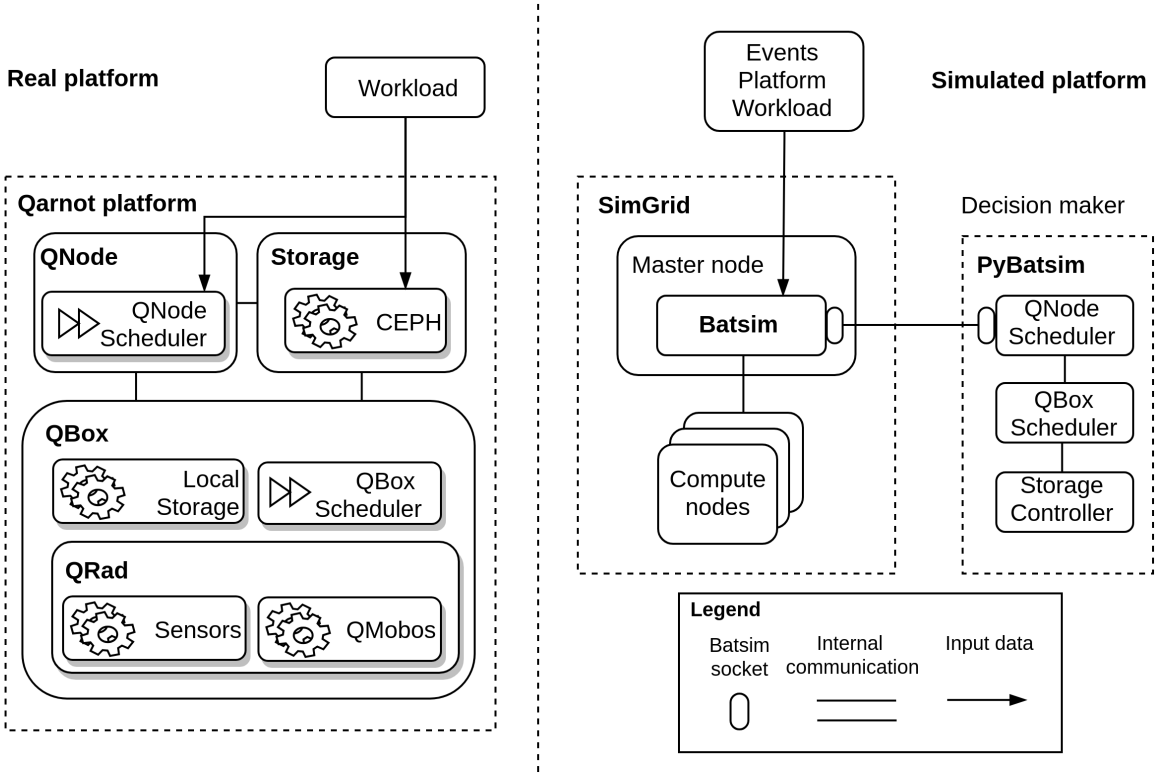


Figure 2.1 – Comparison between the real and simulated *Qarnot* platform.

the QNodeSched will be notified to take some decision (step 03). First, it should know which resources are available. For that, every 30 seconds (defined as default), the QNodeSched ask information about the state of the disk, QRads and QMobos of the QBoxes (steps 03 and 04), in particular, the number of QMobos available for computation and the free storage space are reported.

A first scheduling process is made at the QNode-level to dispatch instances of the QTasks to the QBoxes (steps 05 and 06). The QNode tries to dispatch, for each QTask taken by priority, as many instances as possible onto QBoxes, with respect to the number of available QMobos and storage space left on the QBox disks.

Upon receiving instances of a QTask, the QBox will reserve for each instance a QMobo from the warmest QRad for the case of low priority QTask, and a QMobo from the coolest QRad in the case of high priority (step 08). This distinction is made to keep more QMobos available in case that high priority QTasks are sent to the QBox in the near future. The QBox then checks whether the Docker image and other data dependencies for these instances are on disk and fetches any missing data from the CEPH (steps 08 and 09).

Once all data transfers are completed for this QTask, the QBoxSched will be informed by the StorageController and then will allow Batsim to simulate the instance execution, which will notify the QNodeSched whenever the instance completes (steps 11 through 14).

Finally, the queue of QTasks is updated and if an instance of the same QTask can be directly dispatched, it is sent to the QBox and the execution starts immediately, without rebooting the QMobo (steps 15 through 16).

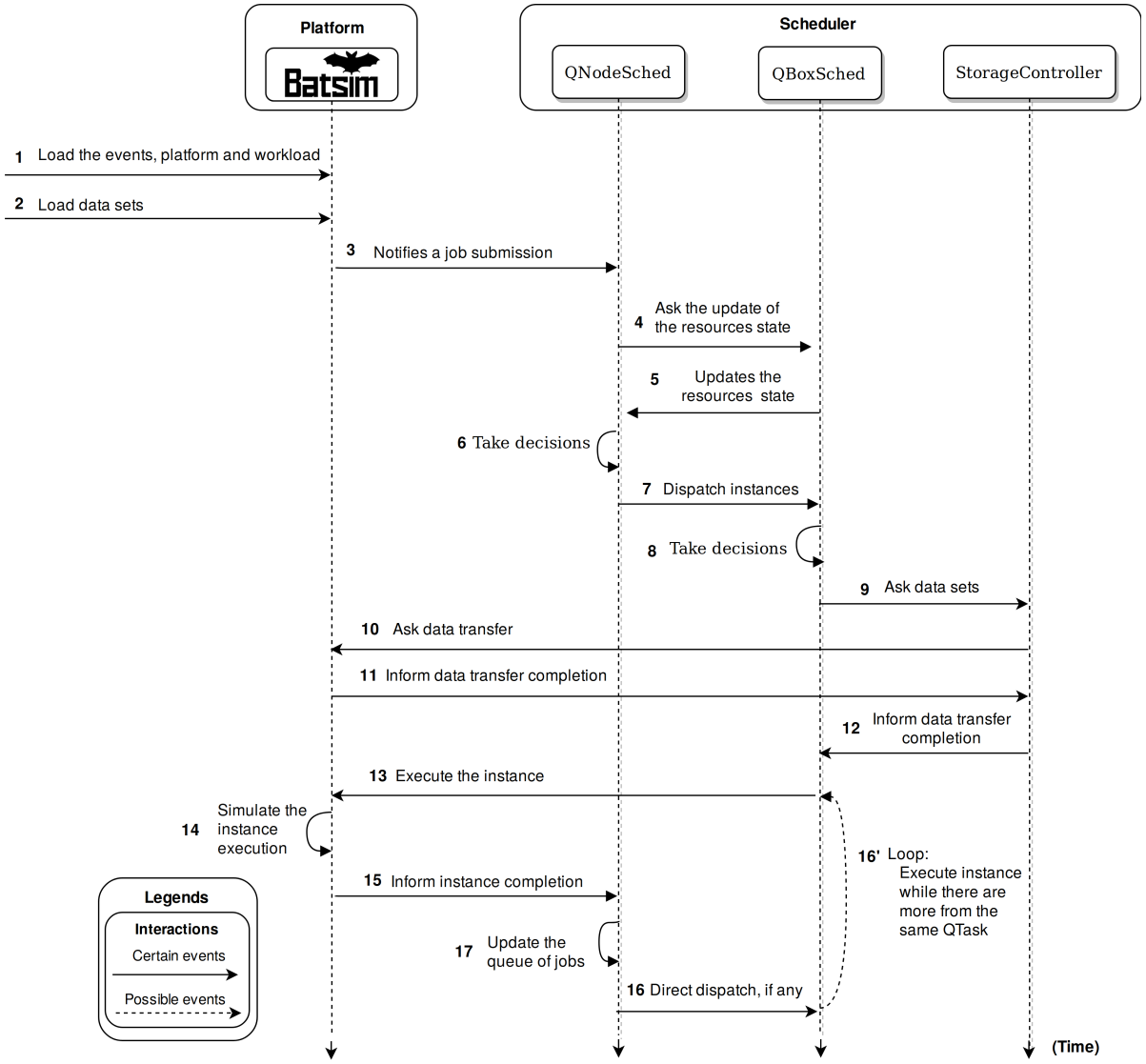


Figure 2.2 – Simulated platform on SimGrid/ Batsim.

2.3 Extracting Qarnot Traces

A log extractor was built to generate all the input files to feed Batsim and the decision process from real logs of the *Qarnot* platform, for a given time period. These files describe the platform, the workloads and their data-dependencies, the list of data-sets and all events that are mandatory to simulate the *Qarnot* system.

2.3.1 Platform Description

The definition of the platform is an XML file readable by SimGrid. The file describes the whole platform to simulate, within details:

A list of QBoxes with for each:

- The id,

- The network bandwidth and latency to the CEPH and to its QBoxes,
- The storage disk host and its size,
- The localization,
- A list of QRads with for each:
  - The id
  - A list of QMobos with for each:
    * The id,
    * The list of speeds and corresponding power usage,
    * The coefficients required for the temperature plug-in.

2.3.2 Workload Description

The workload is represented by a JSON file containing a list of job descriptions and a list of profile descriptions.

Job descriptions are defined by the user requests and contain:

- The id,
- The submission time,
- The job profile to use.

Profile descriptions represent how a job should be simulated, plus other specific information, and contain:

- The type of job to simulate,
- The number of flops to compute,
- The job priority,
- The list of data-sets required as inputs.

In addition, each *Qarnot* instance is represented by a Batsim job with a specific job profile and requiring a single computing resource. Also, each job extracted provides the following information, not used by Batsim, but useful for some analysis that will be presented in further sections:

- The time when the job was started in the real platform,
- The time when the job finished in the real platform,
- The id of the real resource where the QTask was executed,
- The speed of execution in the real platform.

2.3.3 Data Sets Description

The list of data-sets is also described as a list of JSON objects, with one per line in the file is read by the *decision process* and fed to the Storage Controller. Each data-set object is represented by two fields:

- The id
- The size in bytes of the data-set.

2.3.4 External Events Description

As discussed in Section 1.2.1, each event is timestamped and is described as a JSON object:

- *qrad_set_target_temperature*: containing the id of the QRad and its new target temperature. This event informs the associated QBox that the temperature target of a QRad has changed.
- *machine_available* and *machine_unavailable*: containing a list of resources impacted by this event.
- *site_set_outside_temperature*: containing the location and its new outside temperature. This event is directly forwarded to the temperature plug-in(see **??**).

In addition there is a *stop_simulation* event to ask the scheduler to kill all executing jobs and reject waiting jobs to strictly stop the simulation after a given time. It is useful if one intends to simulate a specific period of time, for example, the behavior of a platform for one week.

— 3 —
Job Allocation

As presented in Chapter 2, the HPC job allocation problem has been studied since a long time ago, and the main idea is to fit, by some convenient metric, HPC jobs to be executed as soon as possible onto resources. The metrics and weights taken into account for the decision processes could change as the proposals of the system, but in general, from a queue of jobs, allocation decisions among resources should be taken.

As one of the goals of this work, several scheduling policies were implemented in order to compare their performances. In this section will be described the challenges present in its use-case and all algorithms for the implemented policies.

3.1 Scheduling Challenges

The Qarnot proposal was described in Section 3.1, which is naturally a multi-objective problem, because there are at least three viewpoints. The first one from customers that want to compute (HPC customers), the second one from customers that want to be heated (hosts) and the third one from the middleware.

The viewpoint of the HPC customers is what is found in classical distributed scheduling systems: the goal is to get the results of submitted jobs as soon as possible. The viewpoint of the hosts completely differs from what is found in classical scheduling theory since the Qarnot foundation. Finally, the middleware viewpoint is specifically related to a goal of the Qarnot computing business model, which is the one of reducing the energy consumption in the processing of the jobs, related to the Qarnot business model since the company re-funds the electricity bill of the hosts.

Following the Q.Ware infrastructure, the decision processes are taken in two-levels, through the QNodes and the QBoxes. The scheduling at QNode-level can be viewed as an assignment step. The idea is to dispatch, in priority, tasks to QBoxes having QRads which need to heat the most. The scheduling at the QBox-level is in charge to select the best QRad based on its QMobos, also download the required data-sets and report periodically the status of its resources.

From a QNode point of view, the notion of temperature and heating needs is unknown and hidden behind other information sent periodically by the QBoxes, as detailed below. Upon the arrival of a task, the QNode knows the following information:

- The number n of instances composing the task.
- The priority w of the task.

- The list of data sets $\{D_1 \cdots D_k\}$ which the task depends on.

The priorities are defined as high, low and background, such that QTasks with high priority can preempt low and background others, QTasks with low priority can preempt background others and QTasks with background priority can not preempt any other. This last one is not a real HPC or IoT job submitted on the platform. Named *burn_job*, this Qtask is a fake job created locally when some user requires heating and there are no jobs to be sent to that QRad. This way, this job is useful just to provide heating, been possible to be preempted anytime whenever a real job arrives on that QRad.

Moreover, from periodic reports from the QBoxes, the QNode knows the number of available QRads for each type of priority, which is critical to achieving a good quality of service regarding the priority of the tasks. For example, if there are not enough available QRads to start a high priority task, some lower priority tasks being executed must be preempted to free resources for that high priority task. Hence is necessary to have different values for the available QRads, one for each different class of task priority.

The last source of information to help the QNode to dispatch tasks to QBoxes is provided by the storage controller. This one is in charge of managing the data sets available in the centralized storage and the QBox disks, as well as their movements. It also provides the list of QBoxes already having the data sets required by that task.

Taking all this informations in account, the Algorithm 1, 2, 3, 4, 5 and 6 were developed as follows.

3.2 Standard Schedulers

The standard scheduler for both levels was based on the current Qarnot QNodes and QBoxes policies.

3.2.1 QNode Scheduler

The QNode scheduler is in charge of manage the queue of QTasks and dispatch to the QBoxes that require heating, taking into account the priority of the QTasks. It is the central decision maker and has a global view of the process, receiving information about the available resources from the QBoxes and QTasks from submissions of HPC jobs or from IoT devices. This scheduler was implemented as Algorithm 1.

3.2.2 QBox Scheduler

The QBox scheduler is in charge to require the data sets to the Storage Controller, dispatch jobs to the QRads and start the execution of the QTasks in the QMobos whenever the data sets are ready.

Algorithm 2 aims to execute an instance of high priority on the coolest QRad available, if possible without preempting low instances. Then, it executes an instance with background/low priority on the warmest QRad available.

Algorithm 1 QNode scheduler: dispatching instances onto QBoxes - Standard version

available_mobos_list ← List of available QMobos among all QBoxes
qtask_queue ← The list of QTasks to be dispatched
if *qtask_queue*$\emptyset$ **then**
    return
else
    Sort the *qtask_queue* by 1) decreasing *priority*; 2) increasing $nb\_of\_running\_instances$
    for $qtask \in qtask\_queue$ **do**
        $nb\_instances\_left$ ← Number of instances of *qtask* waiting to be dispatched
        if $nb\_instances\_left > 0$ **then**
            mobos_list ← List of QMobos from *available_mobos_list* with BKGD priority
            Dispatch as many instances as possible on QMobos from *mobos_list*
            if $nb\_instances\_left > 0$ **and** $qtask.priority\_group >$ BKGD **then**
                # There are more instances to dispatch and the qtask is either LOW or HIGH priority.
                mobos_list ← List of QMobos from *available_mobos_list* with LOW priority
                Dispatch as many instances as possible on QMobos from *mobos_list*
                if $nb\_instances\_left > 0$ **and** $qtask.priority\_group >$ LOW **then**
                    # There are more instances to dispatch and the qtask has HIGH priority.
                    mobos_list ← List of QMobos from *available_mobos_list* with HIGH priority
                    Dispatch as many instances as possible on QMobos from *mobos_list*

Algorithm 2 QBox scheduler: dispatching instances onto QRads

waiting_instances ← List of instances waiting to be scheduled on this QBox, sorted by priority
for $qtask \in waiting\_instances$ **do**
    Ask for the transfer of data-sets from the *CEPH* to the *QBox* disk.
    if $qtask.priority$ HIGH **then**
        # Find coolest QRad which is not running LOW instance
        qmobo_list ← List of QMobos which is possible to run HIGH priority Qtasks
        qrad_list ← List of QRads by decreasing temperature
        Run as many instances as possible in $Qmobos \in qmobos\_list$ which are from $Qrads \in qrad\_list$
    else
        # Find warmest QRad among the availLow and availBkgd
        qmobo_list ← List of QMobos which is possible to run LOW priority Qtasks
        qrad_list ← List of QRads by decreasing temperature
        Run as many instances as possible in $Qmobos \in qmobos\_list$ which are from $Qrads \in qrad\_list$
if $waiting\_instances \neq 0$ **then**
    # Some *qtaks* could not be executed.
    Reject the *waiting_instances* to the QNode Scheduler

3.3 QNode Schedulers Variants

Based on the standard QNode scheduler we built other four variants. This way we compared its performances in order to describe which one fits better for the use-cases.

3.3.1 Locality Based Scheduler

The *LocalityBased* scheduler gives priority to the QBoxes already having in disk all data dependencies of the QTask to be dispatched. This first variant, Algorithm 3, aims at avoiding useless data transfers if some QBoxes already have the required data dependencies of a given QTask.

Algorithm 3 QNode scheduler: dispatching instances onto QBoxes - Locality based version

```
    available_mobos_list ← List of available QMobos among all QBoxes
 2: qtask_queue ← The list of QTasks to be dispatched
    if qtask_queue∅ then
 4:   return
    else
 6:   Sort the qtask_queue by 1) decreasing priority; 2) increasing nb_of_running_instances
      for qtask ∈ qtask_queue do
 8:     nb_instances_left ← Number of instances to be dispatched by the qtask
        if nb_instances_left > 0 then
10:       list_qboxes ← List of QBoxes with the data sets required by the qtask.
          mobos_list ← List of QMobos from available_mobos_list with BKGD priority.
12:       mobos_list ← mobos_list filtered by QMobos from QBoxes in list_qboxes.
          for mobo ∈ mobos_list do
14:         Dispatch as many instances as possible on mobo.
          if nb_instances_left > 0 and qtask.priority_group > BKGD then
16:         # There are more instances to dispatch and the qtask is either LOW or HIGH
            priority.
            mobos_list ← List of QMobos from available_mobos_list with LOW priority
18:         mobos_list ← mobos_list filtered by QMobos from QBoxes in list_qboxes.
            for mobo ∈ mobos_list do
20:           Dispatch as many instances as possible on mobo.
            if nb_instances_left > 0 and qtask.priority_group > LOW then
22:           # There are more instances to dispatch and the qtask has HIGH priority.
              mobos_list ← List of QMobos from available_mobos_list with HIGH priority
24:           mobos_list ← mobos_list filtered by QMobos from QBoxes in list_qboxes.
              for mobo ∈ mobos_list do
26:             Dispatch as many instances as possible on mobo.
    Apply Algorithm Algorithm 1
```

3.3.2 Full Replicate Scheduler

The third scheduler, namely *FullReplicate*, replicates all data dependencies of a QTask on all QBox disks before this QTask arrives in the system. This variant, described in Algorithm 4,

aims at visualizing the behaviors of the scheduling policy without any impact of the data movements.

Algorithm 4 QNode scheduler: dispatching instances onto QBoxes - Full Replicate version

Replicate all data sets to all QBoxes.
Apply Algorithm Algorithm 1

3.3.3 3-Replicated and 10-Replicated Schedulers

Finally, the schedulers *Replicate3* (Algorithm 5) and *Replicate10* (Algorithm 6), replicate all data dependencies of a QTask on the 3 and the 10 least loaded QBox disks, respectively. These schedulers will behave as the LocalityBased one, whenever the datasets replications happen. The replications are done upon the submission of a QTask.

As the $LocalityBasedScheduler$ transfers new data sets whenever it is required by some QBox and the $FullReplicateScheduler$ transfers all data sets to all QBoxes whenever a qtask arrives in the QNode, the 3-10$Replicate$ schedulers are two trade-offs between 0% and 100% of data replication before dispatching. They aim at reducing the waiting time of QTasks instances by providing more QBox candidates for the LocalityBased dispatcher.

Algorithm 5 QNode scheduler: dispatching instances onto QBoxes - Replicate3 version

Whenever a QTask is submitted:
  $qbox\_list \leftarrow$ The list of the 3 QBoxes with most empty disks.
3: Replicate all data sets on QBoxes from $qbox\_list$.
Apply Algorithm Algorithm 3

Algorithm 6 QNode scheduler: dispatching instances onto QBoxes - Replicate10 version

Whenever a QTask is submitted:
  $qbox\_list \leftarrow$ The list of the 10 QBoxes with most empty disks.
  Replicate all data sets on QBoxes from $qbox\_list$.
4: Apply Algorithm Algorithm 3

— 4 —
Experiments, Results and Discussions

By the logs extracted detailed in Section 2.3, we were able to obtain workloads for specific periods of time. To compare the results of simulations we utilized workloads of one and three days, one and two weeks. In this chapter we present the results regarding workloads with the size of one week. Due to recent modifications in the Qarnot extractor, we had less than two months of data available. To characterize one full month, we will present four workloads, with each one started from 03, 10, 17 and 24 of May, denoted respectively as 1w_03, 1w_10, 1w_17 and 1w_24. In addition, the platform simulated was composed by approximately 3390 QMobos, from 669 QRads, managed by 20 QBoxes.

This chapter will present analyses and discussions done from two different sources of data, the logs extracted and the simulations results. The logs extracted were used in part to compose the inputs to the simulator and in part to validate the results and to process general information from the real platforms. Then, Section 4.1 and Section 4.2 will present analyses from the real logs extracted. And Section 4.3, Section 4.3.1, Section 4.3.2, Section 4.3.3 will describe the analyses from the simulations results. Finally, Section 4.4 summaries all analyses done.

4.1 Job's Processing Time

By the extracted logs, the processing time distribution for each workload was computed, such that, for each QTask in each workload: $processing\_time = real\_finish\_time - real\_start\_time$. In other words, it was computed a list of processing times, for each instance of a workload, then its distribution is presented in Table 4.1. To facilitate and use the terminology as the common one in the literature, during this chapter we used QTasks and jobs as synonymous.

The Table 4.1 shows that all workloads have 25% of jobs with the processing time bigger than the others 75%. Then, we characterize this instances as *long jobs*. Looking for each workload in details:

- 1w_03: 75% of the jobs are processed in less than 635s, and 25% are processed up to 35372s, which is 55 times the maximal processing time of short jobs.
- 1w_10: 75% of the jobs are processed in less than 425s, and 25% are processed up to 27121s, which is 63 times the maximal processing time of short jobs.
- 1w_17: 75% of the jobs are processed in less than 425s, and 25% are processed up to 29700s, which is 143 times the maximal processing time of short jobs.

Table 4.1 – Processing time distribution for different weeks of workload

Statistics	1w_03	1w_10	1w_17	1w_24
Count	7350	5989	5497	8850
Mean (s)	465.96	582.25	480.21	403.93
Std (s)	817.18	2400.22	2268.20	1723.62
Min (s)	1.0	1.0	1.0	1.0
25% (s)	132.0	77.0	48.0	34.0
50% (s)	235.0	151.0	106.0	117.0
75% (s)	635.0	425.0	207.0	291.0
Max (s)	35372.0	27121.0	29700.0	28952.0

- 1w_24: 75% of the jobs are processed in less than 291s, and 25% are processed up to 28952s, which is 99 times the maximal processing time of short jobs.

Because of this distribution, we decided to split, for each workload, the results of the simulations in two others, one composed of the jobs from the 75% of the distribution, and the other one composed of the jobs from the 25% of the distribution, respectively, denoted by short_jobs and long_jobs. In the same idea will be denoted as all_jobs the original workload.

First, we will present in the next sections the analysis of results of all_jobs, then we will compare them with the analyses of short_ and long_ jobs aiming to point out some possible effect caused by the size of the jobs.

4.2 Data Sets Dependencies

In order to investigate if the data sets somehow affect the scheduling policies, we discussed in this section an analysis regarding the data sets dependencies. We figured out that several QTasks depend on the same data sets, which could cause different results if considered at the scheduling decision phase, for example, the Algorithm 1 does not consider this information.

The figures 4.1, 4.2, 4.3 and 4.4 show in x-axis the id of the data sets and in the y-axis the number of instances that depend on that data set. It is important to emphasize here two points: i) the first one is that a QTask is composed of many instances but, instances from the same QTask could be allocated to different QBoxes, if there are not enough available QMobos on the same QBox at the allocation phase. Then, these instances would require the data set transfer to two or more different QBoxes. ii) The second one, as described in Section 2.3.2, a QTask depends on a list of data sets.

In addition, is important to emphasize here that: i) these figures describe the number of instances requiring the same data sets, which does not mean that the sum of all bars totals the number of instances. For example, an instance could require the data sets with ID: 3, 5, 23 and 30. ii) The data sets with ID 1 represent the *null* data sets, which means that the jobs are not

dependent on any data set.

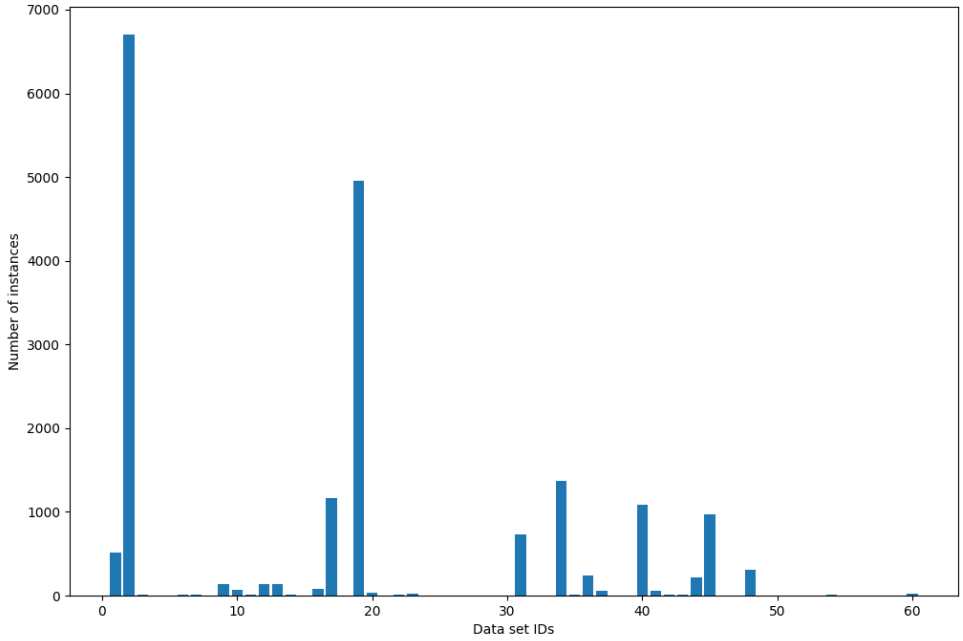


Figure 4.1 – Data sets dependencies for 1w_03 workload.
Number of instances: 7350. **Number of data sets:** 60

From Figure 4.1, one can see that the data set with ID 2 is required by about 6,700 instances, which represents 91% of the total number of instances. It is followed by the data set with ID 19, about 5,000 instances, representing 68% of the total number of instances. In the figure, one can also see other data set IDs reasonably required as 17, 34, 40 and 45, but, not so much as the two emphasized.

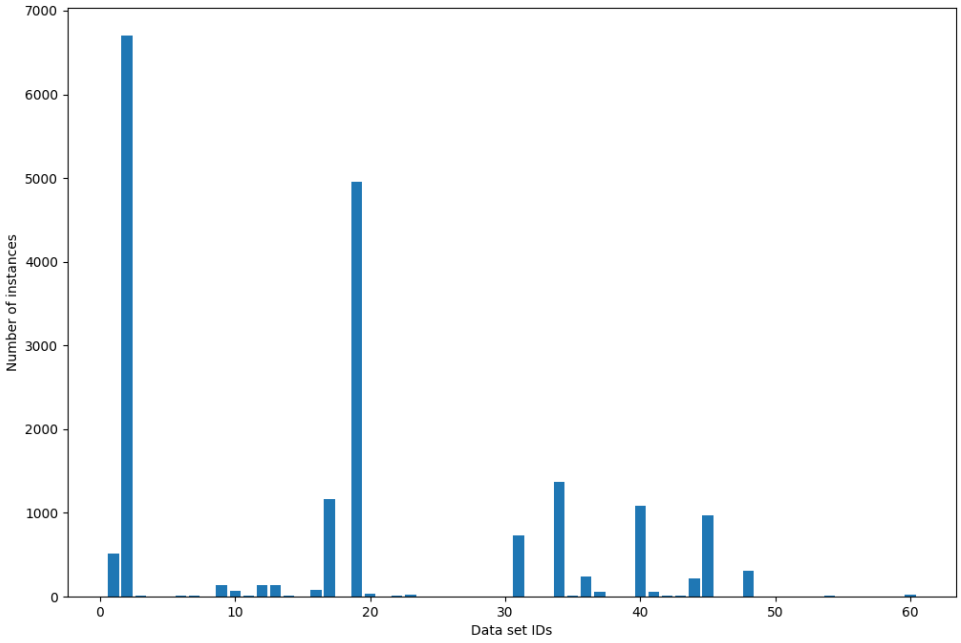


Figure 4.2 – Data sets dependencies for 1w_10 workload.
Number of instances: 5990. **Number of data sets:** 43

From Figure 4.2, one can see that the data set with ID 18 is required by about 2,900 instances, which represents 48% of the total number of instances. It is followed by the data sets with ID 34 and 16, about respectively 2,400 and 1,700 instances, representing 40% and 28% of the total number of instances. In the figure, one can also see other data set IDs reasonably required as 1, 15, 24, 33 and 37, but, not so much as the two emphasized.

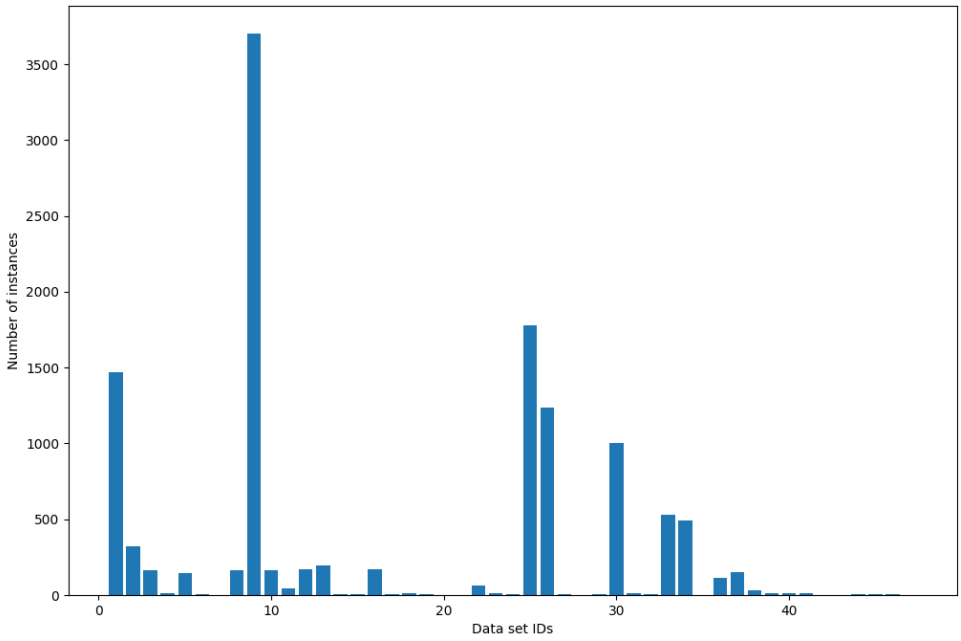


Figure 4.3 – Data sets dependencies for 1w_17 workload.
Number of instances: 5506. **Number of data sets:** 47

From Figure 4.3, one can see that the data set with ID 9 is required by about 3,700 instances, which represents 67% of the total number of instances. It is followed by the data sets with ID 25 about 1,800 instances, representing 33% of the total number of instances. In the figure, on can also see other data set IDs reasonably required as 1, 26 and 30, but, not so much as the two emphasized.

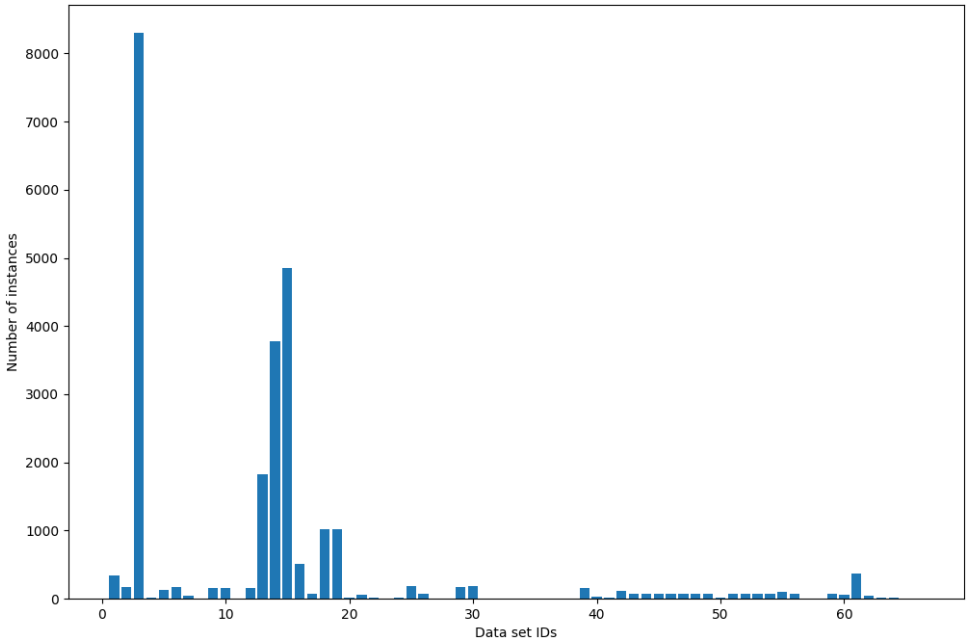


Figure 4.4 – Data sets dependencies for 1w_24 workload.
Number of instances: 8852. **Number of data sets:** 66

From Figure 4.4, one can see that the data set with ID 3 is required by about 8,500 instances, which represents 96% of the total number of instances. It is followed by the data sets with ID 15 about 5,000 instances, representing 56% of the total number of instances. In the figure, one can also see other data set IDs reasonably required as 13 and 14, but, not so much as two the emphasized.

Analyzing the data set dependencies we recognized that the four workloads extracted from Qarnot are not equally distributed in terms of data sets. In other words, it is possible to see from these ones that there are at least two very popular data sets among the instances, which could affect the scheduling policies and will be discussed in the next section.

4.3 Scheduling Metrics

In order to compare various scheduling strategies based on real-world traces of the *Qarnot* platform we discuss in this section the number and the total size of data transfers, along with the bounded slowdown of the instances. The following discussion will be based on plots such that the x-axis represents the workloads, respectively 1w_03, 1w_10, 1w_17 and 1w_24. In addition, it is possible to see in x-axis the 5 scheduling policies implemented as described in

Chapter 3, respectively the FullReplicate, LocalationBased, Replicate10, Replicate3 and the Standard. The y-axis represents the metrics discussed.

4.3.1 Data Transfers

The number of transfers and the total data transferred are two important metrics if, for example, there are huge data sets which would take long time to be transferred, or in the case of limited resources in terms of storage disks, which will not support many data sets in the same machine.

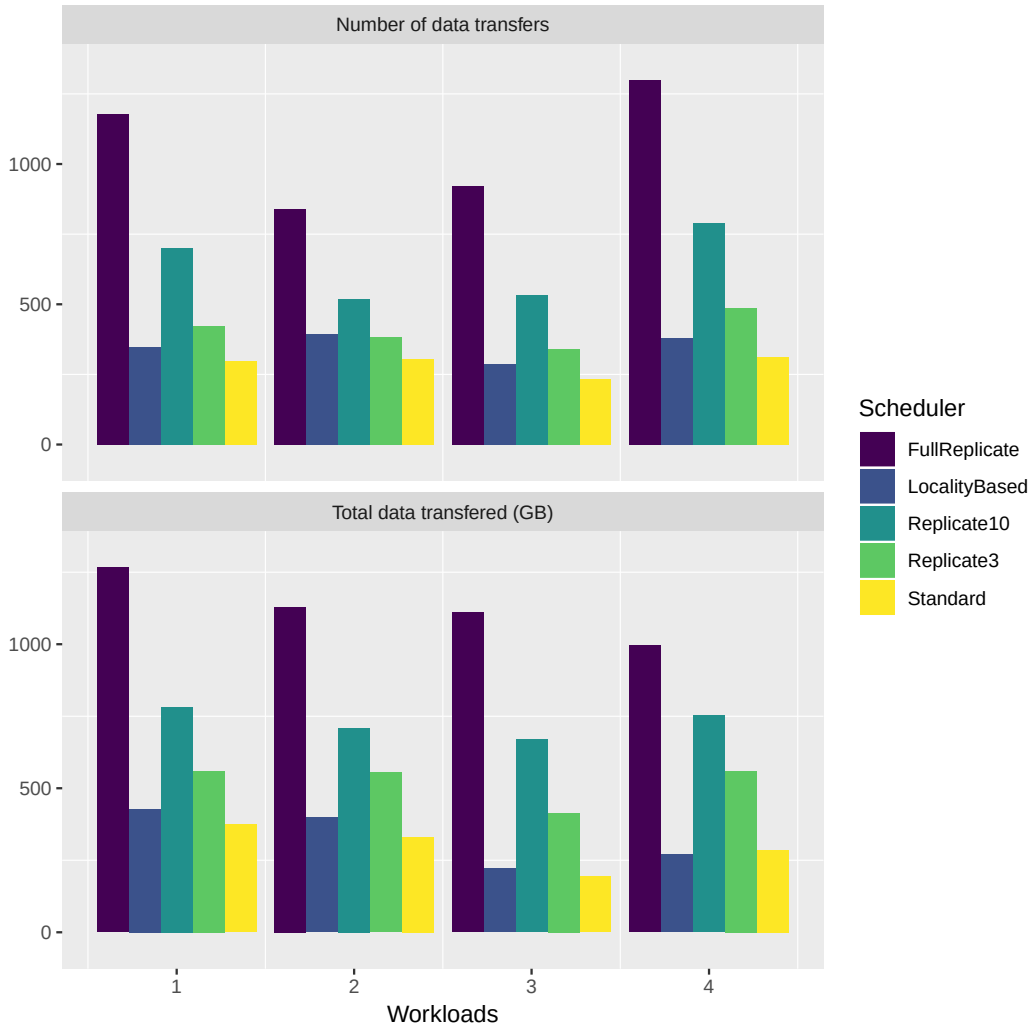


Figure 4.5 – Total data transferred for 1 week workload

From Figure 4.5 one can see that for both the *Number of data transfers* and the *Total size of data transferred (GB)*, the schedulers FullReplicate, Replicate10 and Replicate3 are the three with the highest values, with the exception of the Replicate3Least-Loaded for the workload 2 in the *Number of data transfers*. But, in general, this behavior is totally expected since, respectively, they replicate data sets in all, 10 and 3 QBoxes.

On the other hand, at the first time, we expected that the LocalityBased would reduce the number of data transfers compared to the Standard scheduler. But, as we analyzed the data sets dependencies, this behavior could be justified by the high popularity of few data sets among the workloads. We believe that what happens in this specific scenario is that the LocalityBased would reduce data transfers until a QBox get unavailable running as many instances as possible. Then, this very popular data sets should be transferred to other QBoxes and it would happen, maybe, during the whole simulation. This way, the LocalityBased got closer or higher values between itself and the Standard scheduler, even the last one does not consider any location or data set information in the allocation decision phase.

We believe that if the data sets dependencies would be well distributed among the workloads, the LocalityBased policy would reduce the number of data transfers, but it is not our case and we would need to simulate this specific kind of data to validate it.

4.3.2 Bounded Slowdown

The bounded slowdown is a classical metric that has been utilized in the literature [16] and was implemented in the context of this work because it was not computed from the toolkit utilized. We computed the bounded slowdown as $bounded\_slowdown = max\{(waiting\_time + execution\_time)/max\{execution\_time, \tau\}, 1\}$, such that τ denotes a threshold, equals 1 for our experiments, since this is the minimum size of our instances as the table 4.1. Here we also considered that *waiting_time* of an instance depends on the time to transfer the required data sets and the time to decide in which QMobo the instance should be executed. This metric has been utilized to analyze if the waiting time of a job is proportional of its size. Considering that one important goal of scheduling policies is manage the waiting time of the jobs, it is an important metric. In addition, it is known that this metric is more sensitive for short jobs, since if the execution time is close to zero, this formula depends on the waiting time and the threshold. If the waiting time of short jobs is higher than its execution time, the bounded slowdown will be high, which does not mean that the execution time was high, but the inverse.

From Figure 4.6 one can see that the FullReplicate scheduler presents the lowest values for both measurements, the *Mean bounded slowdown* and *Max bounded slowdown*. For almost all the other cases, the Replicate10 and Replicate3 are the next lowest ones, with the exception of the *Max bounded slowdown* with the second and fourth workloads. But, in general, this behavior is also totally expected since these schedulers replicate much more data sets than the LocalityBased and Standard. Then, the waiting time for jobs managed by FullReplicate, Replicate10 and Replicate3 tends to be small, because it does not depend on the data transfer time, it just depends on the decisions process time, in general.

Following the same justification as Section 4.3.1, the LocalityBased presents closer or higher values when compared with the Standard thanks to the data sets dependencies. We believe that because of the waiting time of the data transfers, the instances managed by these schedulers wait more time than the others managed by the replicate based schedulers (FullReplicate, Replicate10, Replicate3). In addition, we believe that the LocalityBased presents, in general, a bounded slowdown bigger than the Standard because the LocalityBased does, also in general, more data transfers and transfers more data than the Standard.

To analyze in more detail, as we explained above, the bounded slowdown is more sensitive for short jobs than long jobs, then in the next section we will present comparisons looking for grouped instances by their sizes.

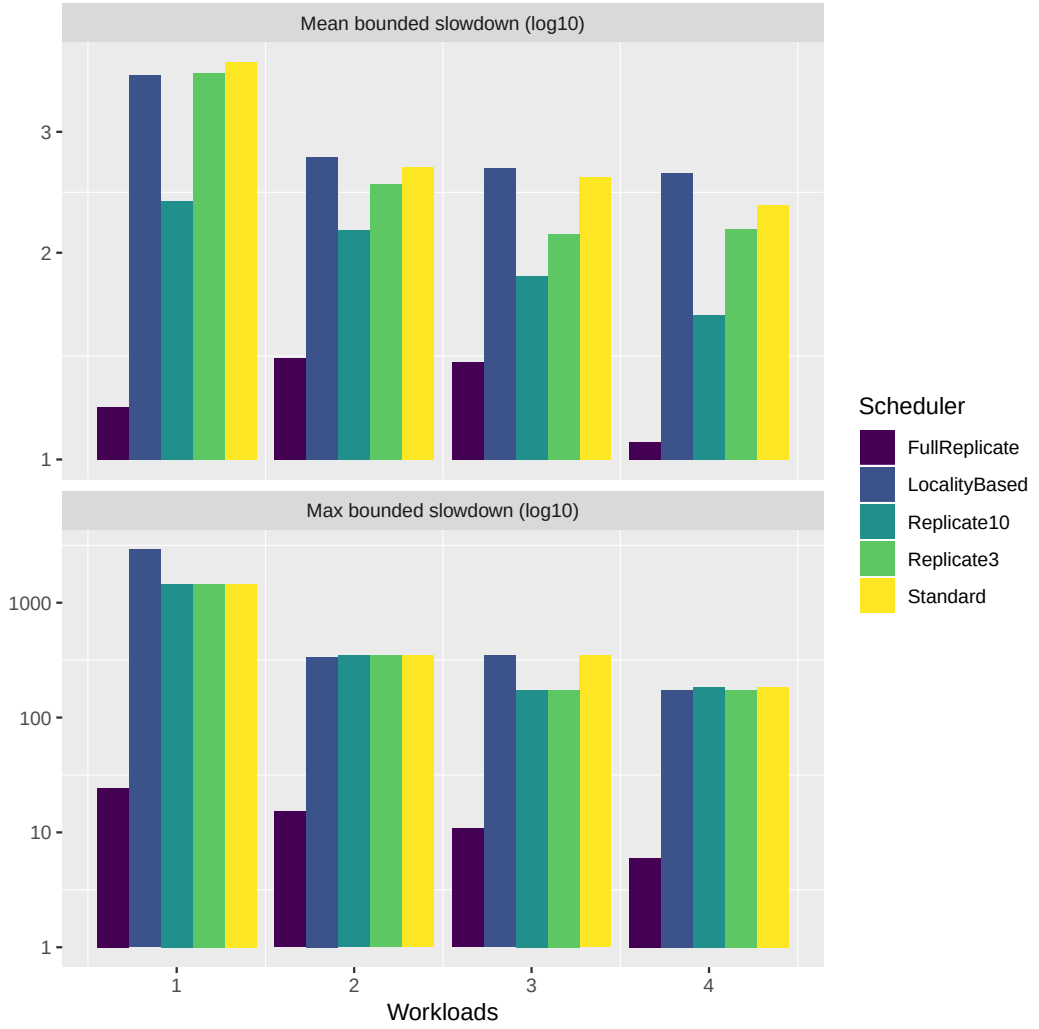


Figure 4.6 – Bounded Slowdown for 1 week workloads

4.3.3 Job's Size Effect in the Measured Metrics

Due to the characterization of 75% of the jobs as short and 25% as long, the results of the previous simulations were split into two others. This way we investigated the effect of the short and long jobs on the bounded slowdown. It is important to emphasize that the data comes from the same simulation, we filtered the results from the *all_jobs* and split into two other, as *small_jobs* and *long_jobs*, as explained in section 4.1.

As one can see, the behavior in Figure 4.7 is, in general, the same as Figure 4.6. As this data is composed only for short jobs, its *execution_time* is low, then, we attributed the high values visible in Figure 4.7 to its *waiting_time*. Finally, as the replicate based schedulers (FullReplicate, Replicate10, Replicate3) present low values when compared with the others, we attributed that the *waiting_time* for the LocalityBased and Standard schedulers is impacted much more by the data transfers time than the allocation decision process time.

The behavior in Figure 4.8 is different when compared with Figure 4.6. Here the values for the FullReplicate are the highest ones and we justify it by the *waiting_time* from the allocation

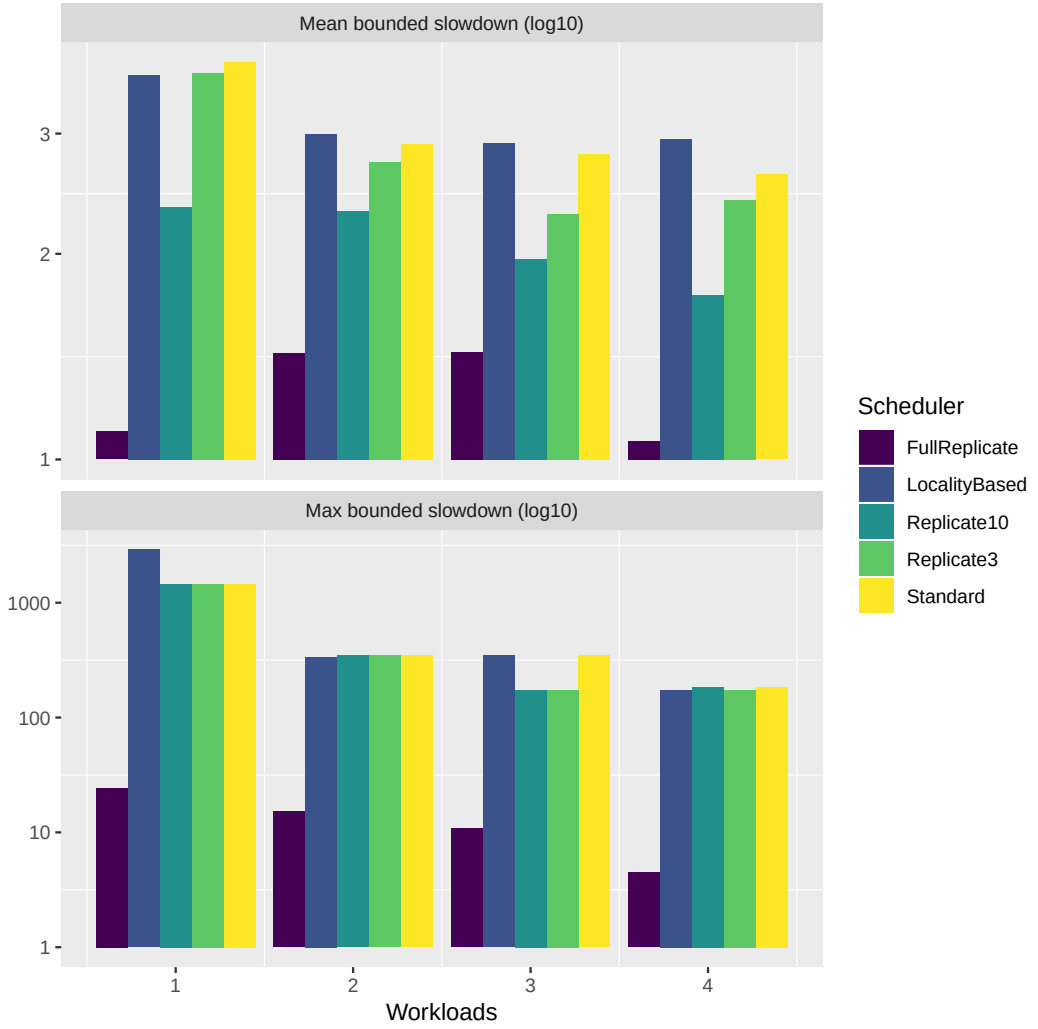


Figure 4.7 – Bounded Slowdowns for short jobs from 1 week workloads

decision process that, in general, takes more time than when it is done for short jobs. But, is important to emphasize here that the values of Figure 4.8 in the *Mean bounded slowdown* are lower than the presented in Figure 4.7, that is why is not possible to see the second tick in the y-axis of this plot.

Comparing Figure 4.7 and Figure 4.8 one can see that the premise that short jobs are more sensitive to these metrics is true in our case, because the first figure presents higher values than the second. And considering that 75% of the jobs are being represented in Figure 4.7 as the short jobs and 25% in Figure 4.8 as long jobs, we understood that the behavior of Figure 4.6 is much more impacted by the short jobs into these workloads.

4.4 Analyses of Results

All performed simulations were deterministic, then we ran one simulation with each scheduler for several inputs corresponding to 1 day, 3 days, 1 week and 2 weeks of the *Qarnot*

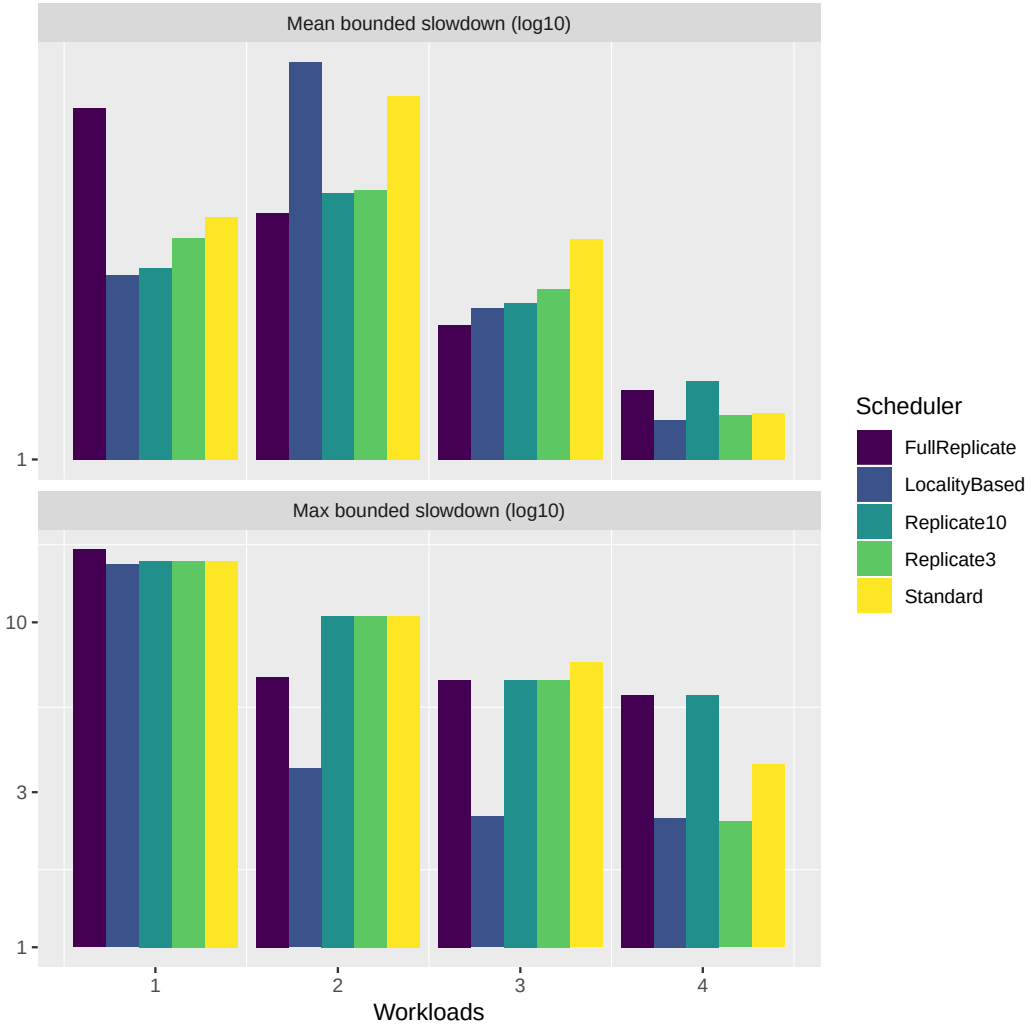


Figure 4.8 – Bounded Slowdowns for long jobs from 1 week workloads

platform. The running time of one simulation was less than 5 minutes for a 1-day simulation, around 10 minutes for a 3-day simulation, less than 35 minutes for a 1-week simulation and around 50 minutes for a 2-week simulation, with about 15% of the time was spent in the decision process.

We compared the different scheduling policies according to various metrics, including the number of transfers, the total transferred data and the mean and max bounded slowdown. In addition, we analyzed the job's processing time distribution and the data sets dependencies.

Due to lack of space, and as the simulations of the four periods (1-day, 3-days, 1-week and 2-week) lead to similar conclusions, we only present in Table 4.1 the results for the 1-week period.

The results show that, as expected, the three policies using replication improved the scheduling metrics compared to Standard, with a cost of an increasing size of the data transfers. Regarding the LocalityBased scheduler, the improvement of performance is quite low and even worse for all presented metrics, when compared with the Standard algorithm. This was quite surprising at first as the total transferred data actually increased, while the initial purpose of

this scheduler was to leverage the data transfer already performed. But, regarding the data set dependencies it was justified. Such results, which are counter-intuitive, clearly illustrates the importance of validating the behaviors of scheduling algorithms through simulations before envisioning their real deployment.

The first thing to notice is that all variants improve the scheduling metrics compared to the standard algorithm of *Qarnot*, except for the mean bounded slowdown of LocalityBased, which is very close to the baseline, with a cost of an increasing size of the data transfers.

Comparing the replication policies, we can see that *Replicate*3 and *Replicate*10 present similar results, and that the FullReplicate gains are almost double compared to the partial replication policies with a cost of doubling the size of the data transfers as well. We can deduce that replicating data-sets before applying the LocalityBased placement policy is beneficent users' point of view, but deciding how much replication the system should do is not trivial. Going from 3 to 10 replicas does not seem to improve much the quality of service while doubling the cost in terms of data transfers, and duplicating the data sets everywhere almost halves the mean waiting time and bounded slowdown compared to the standard *Qarnot* scheduler, at a cost of multiplying by 5 the total size of data transfers.

Finding a good job scheduling policy for *Qarnot Computing* is still an ongoing action. And regarding the data sets dependencies, further strategies could predict the data transfers time and then, the scheduling policies could be mixed between a replicated one whenever a very popular data set is recognized, and locality based if the data sets would be equally distributed among the jobs.

— 5 —
Conclusion and Future Remarks

5.1 Concluding Remarks

We utilized in this work a dedicated toolkit to evaluate scheduling policies in edge computing infrastructures. Its integration into a simulator leads to a complete management system for Edge Computing platforms that focus on the evaluation of scheduling strategies.

This work presented how to use a simulated Edge Platform to easily evaluate existing placement strategies, even if the platforms complete validation was still an on-going work. It may also serve at developing and testing new strategies thanks to its modular and clear interface. To assess the interest of such simulator, we instantiated the toolkit to simulate the whole Edge Platform of the *Qarnot Company* based on smart heaters.

We showed that to get a strategy as the best one, first of all, it is important to know the workload that will be managed and its data-set dependencies. Thanks to our use case, we investigated several scheduling strategies and compared them to the actual policy implemented in the *Qarnot* platform. We showed that the workloads managed by the Qarnot are composed, in general, of 25% of long jobs and 75% of short jobs. We also showed that a majority of instances in these workloads require the same data sets. Finally, we showed that considering the context of the Qarnot Computing, the best strategies are those which take into account replication of data-sets.

We considered the work that has been developed until here very important to the *Qarnot Computing* in the sense that it could be a very useful tool to simulate their strategies, intended modifications, to anticipate some behaviors and also to prepare environments and offices that would utilize their products. All of it thanks to a simulated tool, that allows engineers to perform this kind of studies without affecting their production system.

Since this context presents many challenges, we know that there are several possibilities to continue this work and the main future goal is to achieve the development of a complete Edge Computing Simulator, that is still on going due to the difficulties presented in Chapter 2. But, we consider that this manuscript was a step forward in this context.

Part III

The Logs Data Analysis

— 1 —
Methodology and Data Analysis Process

Our analysis followed four main steps: (i) the data understanding process, (ii) the measurement errors recognition, (iii) an analysis descriptive of the data, (iv) a method to categorize the Smart Heaters by their temperatures distributions.

1.1 Data Understanding

In the first step (i) we had our first contact with the Qarnot' logs. Then, it was necessary a first overview of the data, its structure, the size of the files and how long does the logs follow the QRADs temperatures. At this step we looked to all information available in those logs and decided which one would be useful for our objectives.

1.2 Measurement Errors Recognition

The second step (i) was performed in order to clean our data, for that, we first filtered the measurements errors from all data available, in order to better search and remove the errors from other measurements, which could mess around the temperatures ones. Then, we selected the five type of temperatures mentioned in the previous section, ignoring the other information available in the logs (i.e, powers, frequencies, etc).

1.3 An analysis descriptive of the data

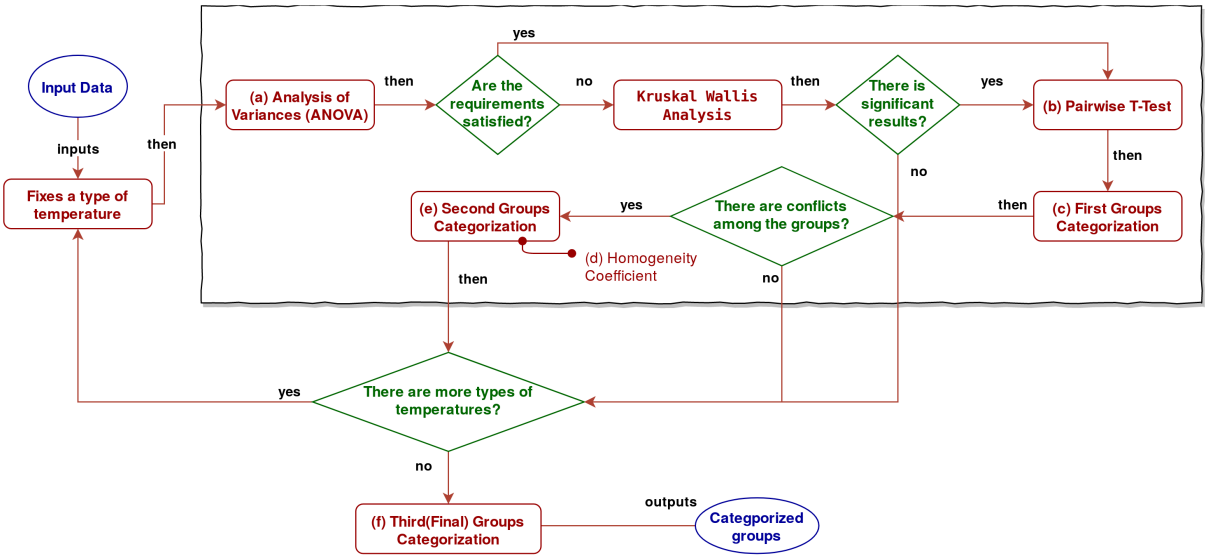
The third step (ii) was done in other two other steps: (a) by an overview of the data, since this is the first analysis of the Qarnot's logs, it was necessary to analyse how the data behaves in general. For that, we searched for particularities and patterns. After that, we saw the necessity to split the temperature range, which was originally from 0º to 80º C, in three others. Then, on step (b) we analysed the Smart Heaters for each range, analysing their temperatures distribution together and individually.

1.4 Smart Heater Characterization

In the fourth step (iii) we developed and applied a methodology to group the Smart Heaters based on their behaviors, reminding that we considered as behaviors the distributions of tem-

peratures. It is also important to remind here, that we performed the following methodology for each range of temperature from step (ii-a). This methodology followed other six steps, illustrated in the Figure 1.1.

Figure 1.1 – Workflow for the QRADs' Categorization



Depicting each step from the Figure 1.1: (a) on subsection 1.4.1 we fixed and performed an Analysis of Variances (ANOVA) for each type of temperature, in order to verify if there were similarities among the different Smart Heaters. Then, since we noticed on (a) that there were significant differences among the Smart Heaters, on step (b), in subsection 1.4.4 we investigated in which pair of Smart Heaters were those differences. For that, we performed pairwise t-test comparisons among all Smart Heaters, and for those pairs where did not result significant differences, we considered that it means that there were similarities between those machines. Then, on step (c), in subsection 1.4.5 we grouped all machines considered similar. But, looking for all groups created there were some conflicts, which means that there were different groups containing the same Smart Heaters. For that, we developed, as step (d), in subsection 1.4.6, the Homogeneity Coefficient, a measurement of the strength of a group, based on the results from (b). This way we measured in which group a fixed Smart Heater fit better. Using (d) we solved the existent conflicts on the step (e), in the subsection 8. At this point we got a set of groups of Smart Heaters for each type of temperature. It means that each set has the best groups of Smart Heaters based on only one temperature. As our final goal, we want to group them based on all temperatures. For that, we performed the last step (f), in subsection 1.4.8 which is the same as (e), but now we grouped the Smart Heaters from all groups and sets got so far.

The reminder of this section will depict this full method to group/categorize the QRADS, step by step.

1.4.1 One-Way Analyses of Variance

The analysis of variance (ANOVA) is a method to analyze the differences among group means in a sample. It is based on the law of total variance, where a fixed variable is partitioned

into components attributable to different sources of variation. In a simple analysis, ANOVA provides a statistical test of whether two or more population means are equal, generalizing the t-test beyond their means.

This analysis supported us to answer the following question: *Can we take the analyzed distributions as similar?* and the answer was a significant value, p-value, which compare all groups and answer if we could do it.

Since we want to categorize our QRADs based on its temperatures distributions we will use all distributions of the same type of temperature in the same data set, to compare all of them. This way we will use the Analysis of Variance to know if there are significant differences among those distributions in order to know and investigate each difference or similarities. It is important to emphasize now that, we will use the argument of non difference among QRADs to consider that there are a similarity among themselves.

Since this analysis do not show us information regarding each group (QRAD) in specific, we just now that for this data there are or not significant differences, but we do not know where. Then, we will perform a pairwise test to investigate it. But first, we need to examine two requirements for the ANOVA analysis.

1.4.2 The requirements and relaxed metrics

The One-way Analyses of Variance requires that the distributions should follow the normal distribution, and their variances should be equal. For these reasons, we analysed the residuals for the ANOVA model computed.

1.4.3 The Kruskal-Wallis test

Although the two ANOVA requirements were not respected in our data, we computed the same steps again using the Kruskal-Wallis Test, which is a test based on the ANOVA but deal with the no respect of their requirements.

Even the ANOVA or the Kruskal-Wallis test, they do not show us where the found differences are, these analyses just tell us that there are differences among the analysed groups but they did not point where.

For this reason the next step is the pairwise analysis which goes one step further comparing all groups(QRADs). Then, it will be possible to investigate which one is similar and which one is not.

1.4.4 Pairwise T-Test

The Pairwise T-Test use the fact that there is a significant difference among the groups analysed by the Kruskal-Wallis test, and calculate pairwise comparisons between group levels with corrections for multiple testing.

The result is a table of p-values for the pairwise comparisons. In our case, the table shows the comparisons among all QRADs depicted as pairs *Q[01..15] x Q[00..14]*. For each line we have a fixed QRAD that is compared with all others. It is a lower triangular table, because the half part above the secondary diagonal repeat the same values. All cells shows the p-value for the comparison between the two QRADs.

Based on that, now, we were able to verify which QRAD can be taken as different and which one can not compared with the others. Since our goal is to find a method to group the similar

ones, we are searching for those line which a p-value > 0.05. In other words, we will consider as similar those QRADS, which can not be taken as different by their p-value computed.

1.4.5 Categorization

To categorize the QRADs merging them in groups we will use the tables produced in the previous step, each one for their own kind of temperature.

First of all, it is important to remember that we are interested in the QRADs with p-value > 0.05. As each p-value represents a pairwise test, we write it as *p-values[QA, QB] = alpha*, then if *alpha* > 0.05 we conclude that there are not results representative enough to show a significant difference between QA and QB, then we consider that they are similar.

For that, we performed the first categorization step, grouping the QRADs as follows: for instance, lets take the QA and QB if p-values[QA, QB] > 0.05, for a fixed QA and compared with all others, as QB. Finally, the QA is fixed in the end off all groups to show from which line of the p-values table that group were extracted.

Since there are different groups with the same QRADs, wich is not desired because our goal is to group the QRADs without replications. Then, we are also considering that as much as the p-values are closer to 1, the QRADs are more similar. Notice that the p-value = 1 means that both QRADs are equal. Based on this idea, the next step is a method to compare the different groups created in this step, in order to decide in which group a specific QRAD should be owned, since it was pre-grouped in two or more groups.

1.4.6 Homogeneity coefficient

Regarding the p-values computed and the significance of the measurements, as much as a p-value are close to 1 it means that those two QRADs are more similar from each other. The 7 presents how the homogeneity coefficient of the groups was computed.

Algorithm 7 Homogeneity Coefficient

```
Data: group, p_values_matrix
Result: the homogeneity coefficient of the group
if len(group) > 1 then
    total = 0
    count = 0
    aux_groups = group.copy()
    fixed_log = get_last_item(group)
    group = remove_from(group, fixed_log)
    for qrad in group do
        total += p_values_matrix[fixed_log, qrad]
        count += 1
    end
    return total/count
else
    return 0
end
```

1.4.7 Merge groups into categories

Whenever we found a conflict of elements among the groups, which means that there are the same QRAD or QRADs in different groups, we need to decide in each one this item or items should continue.

This way, we will use our *homogeneity coefficient* to decide in which group the QRAD contributes more to increase their homogeneity coefficient, and the Algorithm 8 presents how we did it.

Algorithm 8 Group decisions algorithm

```
Data: group_a, group_b, fixed_item
Result: 0 for group A, or 1 for group B
coeff_a = homogeneity_coefficient(group_a)
coeff_b = homogeneity_coefficient(group_b)
if (len(group_a)-1 == 1) and (len(group_b)-1 == 1) then
    if coeff_a ≤ coeff_b then
        # keeps the fixed_item on group B
        return 1
    else
        # keeps the fixed_item on group A
        return 0
    end
else
    coeff_aux_a = homogeneity_coefficient(group_a.remove(fixed_qrad))
    significance_a = coeff_aux_a - coeff_a
    coeff_aux_b = homogeneity_coefficient(group_b.remove(fixed_qrad))
    significance_b = coeff_aux_b - coeff_b
    if significance_a ≤ significance_b then
        # keeps the fixed_item on group B return 0
    else
        # keeps the fixed_item on group A
        return 1
    end
end
```

Then, in our case, we will have five sets of groups, one for each type of temperature, at the end of this step. The next and the final one is to merge these sets of group in an unique one.

1.4.8 Merge groups from different categories

As our last step, we need to group the QRADS based on many categories. Then, we will use the groups from the last step, which respect each type of temperature, and we will apply the same procedure to merge all of them to get the final groups.

For that, we used the same procedure. It is important here to emphasize that the step to compute the *homogeneity coefficient* takes in account the p-values table from that specific type of temperature, and now we have three different p-values tables to take into account. We take that there is no needs to differ the method because whenever we solve a conflict among groups from different type of temperatures our method decides to keep the QRAD which affects

more its group for its type. Which means that in the end, we will have always the highest homogeneity coefficient for all groups.

— 2 —
Results and Discussions

In this chapter we present the results from each step described above. It is important to remind that at some points the analysis were built using all data available, and at other points, it was done analysis with data filtered based on the temperature ranges.

2.1 Data Analysis

The first part of the results, presented in this section, will follow discussing the analysis of the logs from Qarnot, the data understanding process in Section 2.1.1, their measurement errors in Section 2.1.2, a general data description in Section 2.1.3, its details for each different ranges of temperature in Section 2.1.4, Section 2.1.5, Section 2.1.6, and an analysis regarding the stability of the data in Section 2.1.7.

2.1.1 Data Understanding

We had access to sixteen logs from smart heaters of Qarnot Computing, which vary from a minimum of 40 minutes to 3 weeks of measurements. Each registry in the logs contains raw measurements of a certain point in time. This measurement is usually set to be performed at each 10 seconds. For each timestamp we have the following measurements regarding the smart heaters: *temperatures*, *powers*, *frequencies*, and *states (on/off)*. At this first moment we are interested in study the temperatures of the smart heaters. For this task we therefore focused on five types of temperatures that are present in the logs, namely (i) *ambient temperature*, which is the temperature of the environment where the smart heater is situated, (ii) *CPU temperature*, (iii) *heatsink temperature*, which is the temperature of the main heat conductor between the CPU and the environment, and (iv) the *target temperature*, which is the temperature requested by the heating user.

2.1.2 Measurement Errors Recognition

We analysed the logs in order to find and remove measurement errors. For the temperatures the highest one was 55125 ºC for the *heatsink*, which we considered impossible to happen. As the minimum values, we found 0º C for the *motherboard* even with all other temperatures were higher than that, then we also considered these measurements as errors. In the end, we defined as valid, the temperatures higher than 0º C and lower or equal than 80º C.

Even if we were interested only on the temperatures, we took into account the errors measured from other parameters as the *powers*. We did it because we considered that if the power measurements were wrong, we could not trust in the other values for that sample. We figured out power measurements like -7344187 W and 7344259 W. We defined as possible values the powers inside the range starting from 0 to 1000 W.

At the end of this step we filtered 150258 samples from 1265825 originally, representing about 12% of useful data.

2.1.3 Data Description

To first have an overview of all QRADs and their temperatures distributions, we summarized all these information in the Figure 2.1, which presents all the sixteen QRADs and their temperatures distributions as box-plots. Each QRAD was named as 'Q' plus a number from 0 to 15:

At a first view of the Figure 2.1 it would be possible to conclude that there are many similar QRADs, but we should pay attention in some parameters, as the range of the temperatures which are from 0º to 80º, for many days. These box-plots could hide information as machines turned off almost all the analysed time and turned on just one day. Also, there are many point recognized as outliers for various QRADs.

Looking for the box-plots shapes, it is possible to see that many of them represent distributions that varied among the time, which means that the temperatures increased and decreased. We assume that it should happen for mainly two reasons: (i) the QRAD users changed the requested temperature, for more or lass than previously, or (ii) whenever a job was finished, it would take some time to receive and start to run another one, implying in the temperature decrease for some time.

Then, considering this behavior, we understood that whenever a temperature starts to decrease or increase, the other one should follow this behavior. But, from this figure is difficult to see details and to analyse behaviors such the increase or decrease of temperatures.

For that reason we decided to split our analysis in three moments, using three ranges of temperatures. This way, we could mitigate information that could be hided. From several attempts, we decided to split our data following the following ranges: a) [0,35]º C, b)]35, 60]º C and c)]60, 80]º C. The following sections will present the analysis of each one of them.

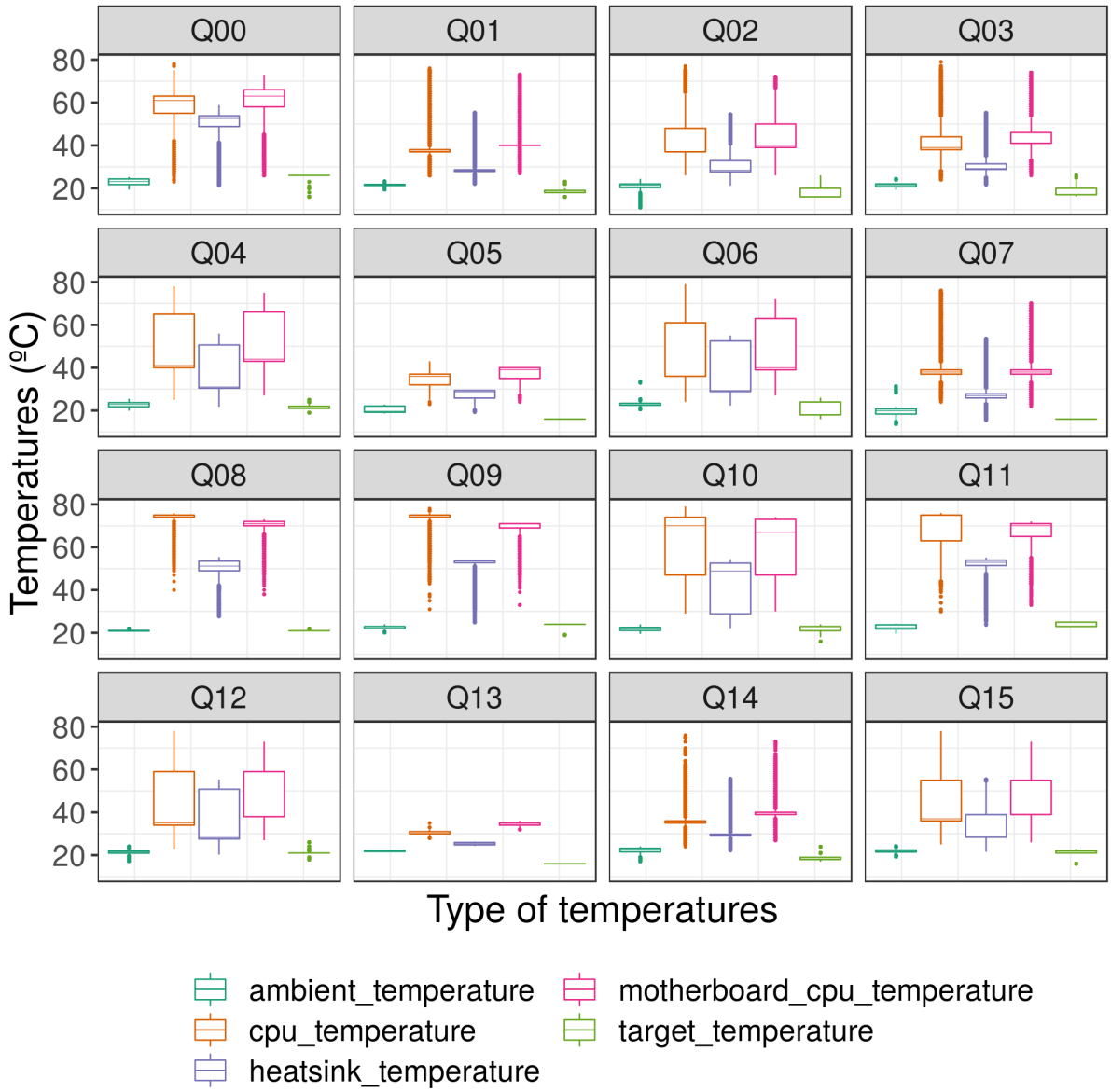
2.1.4 The first range

Analysing the 2.1 we could see that, mainly the ambient and the target temperature were centered around 20 and 33º C. Also, there are many QRADs with values under 30º C. Then, the first range of temperature starts from 0º C and overs at 35º C, included. The Figure 2.2 presents all QRADs with all distributions inside this range.

From the Figure 2.2 it is possible to see that the Q08 does not have some type of temperatures, which means that there is no register of that type of temperature in the current range. A second observation from this figure is that the type of temperatures achieving temperatures lower then 20º C are the ambient and the target, which is completely expected.

In order to go further in the details, we fixed each type of temperature to compare the QRADs. The following figures present the temperatures distributions as box-plots and densinty-plots for all QRADs:

Figure 2.1 – Distributions of Temperature Summarized by Box-Plots, for all QRADs



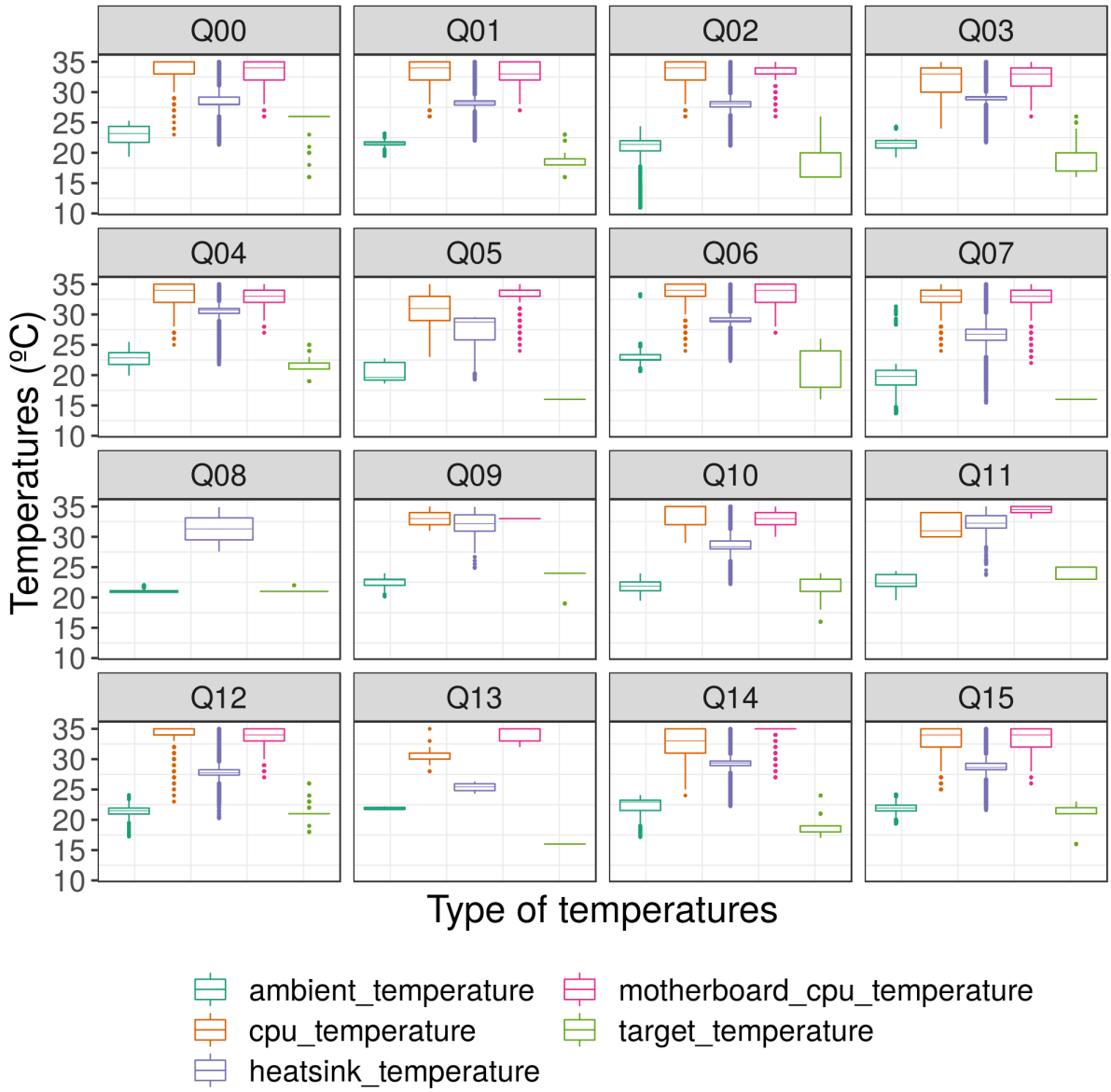
- **Ambient**

  The Figure 2.3 shows the ambient temperature which is more or less in the range from 18º to 25º C, with some exceptions achieved by the outliers.

- **CPU**

  The Figure 2.4 presents the CPU temperature distributions by box-plots (a) and density-plots(b). It possible to see that the distributions are inside the range [28,35]ºC, in details. A remark possible to be done here is that those machines could be idle based on its temperatures. The density plot (b) shows better the similarities among the distributions, because it is possible to see the same behaviors by the similar curves, with the difference of the slopes, representing the different frequencies.

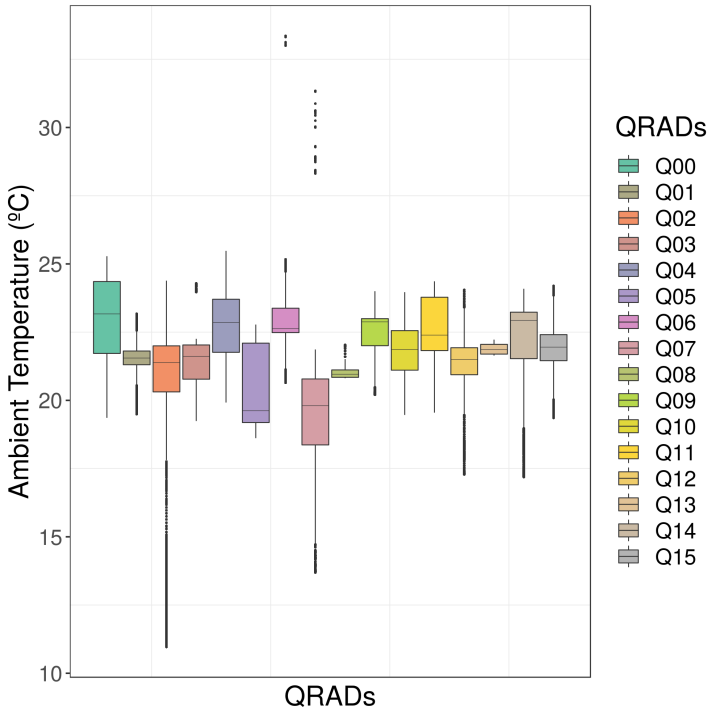
Figure 2.2 – Distributions of Temperature Summarized by Box-Plots, for all QRADs, in the first range



- **Heatsink**

  From the Figure 2.5 it is possible to take as minimum values 20º C with exception of one QRAD with a minimum value about 15º C. Although, the majority of values are around 25, 30º C. The main observation here, is that the minimum value of the heatsink distributions are about 20º C, but from the Table 2.6 is possible to see that the CPU minimum values are about 23º C. Then, this difference emphasizes that the heatsink temperatures is not the same as the CPU because the heatsink is also in contact with the ambient temperature. In other words, the heatsink temperature represents the merge between the inside components temperature and the ambient one.

Figure 2.3 – Ambient Temperature Distributions, for all QRADs, in the first range



- **Motherboard**

  From the Figure 2.6, it is possible to see that all QRADs follow the same behaviors when compared with the CPU distributions in the Table 2.6, as expected.

- **Target**

  From the Figure 2.7 it is possible to see that there are some QRADs with an unique temperature for all distribution, with exception of some outliers, as the Q00 about 26º, Q05, Q07 and Q13 about 16º, Q08 and Q12 about 21º, and Q09 about 24º all the time. The main remark here is that the majority of the target temperatures are almost fixed. Then, why the other types of temperatures does vary a lot, even the one in the ranges presented until here? We expected that once the target temperature would be achieved, the distributions of the other types of temperatures should stabilize, because, in the end, this is the main goal of the QRADs (aka. heat the users).

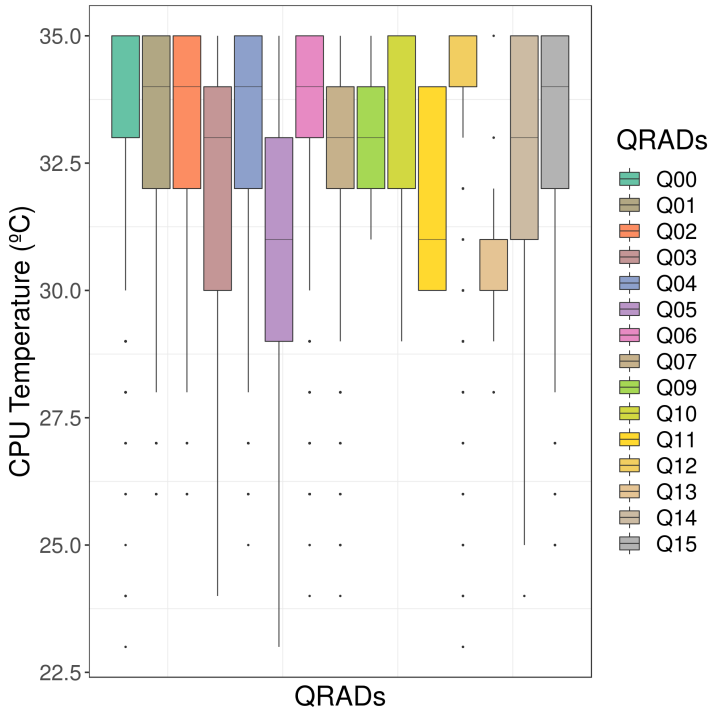
2.1.5 The second range

From the firs Figure 2.1 it is possible to see that the heatsink, cpu and motherboard temperatures vary, mainly from 35º to 80º. As it is also possible to see that just few QRADs achieve such high temperature as 80º, we fixed the second range starting from 35º to 60º, included. The Figure 2.8 presents the QRADs behaves inside this range.

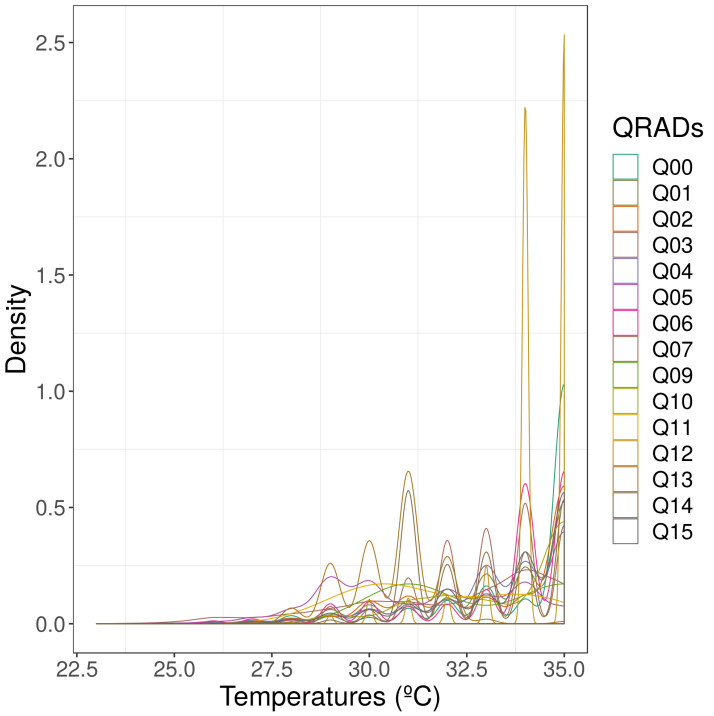
From the Figure 2.8 is possible to see that there is no ambient or target temperature in this range. Which means that the ambient are only present in the previous range, with the maximum of 30º. Since the target is also not present here, it means that the temperatures in this range are those achieved by these components to produce the heat required in the QRADs (smart heaters).

Figure 2.4 – CPU Temperature Distributions, for all QRADs, in the first range

(a) Boxplot of distributions.



(b) Density of temperatures.



The following topics present box-plots and density-plot for the distributions of all QRADs, for each fixed type of temperature.

Figure 2.5 – Heatsink Temperature Distributions, for all QRADs, in the first range

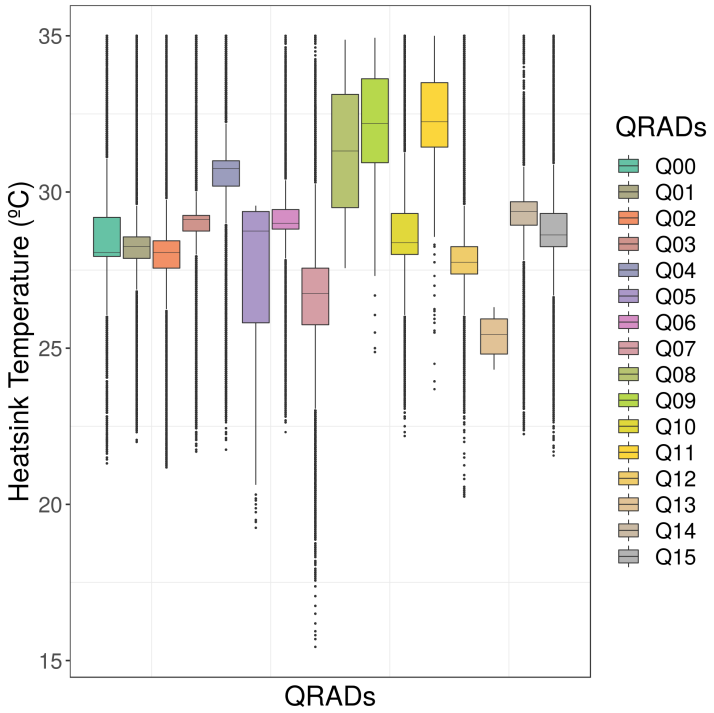
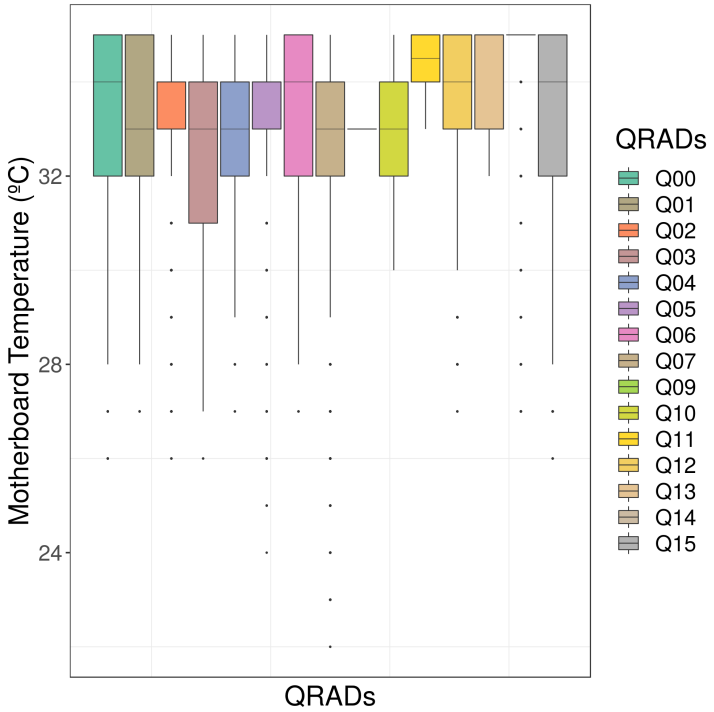


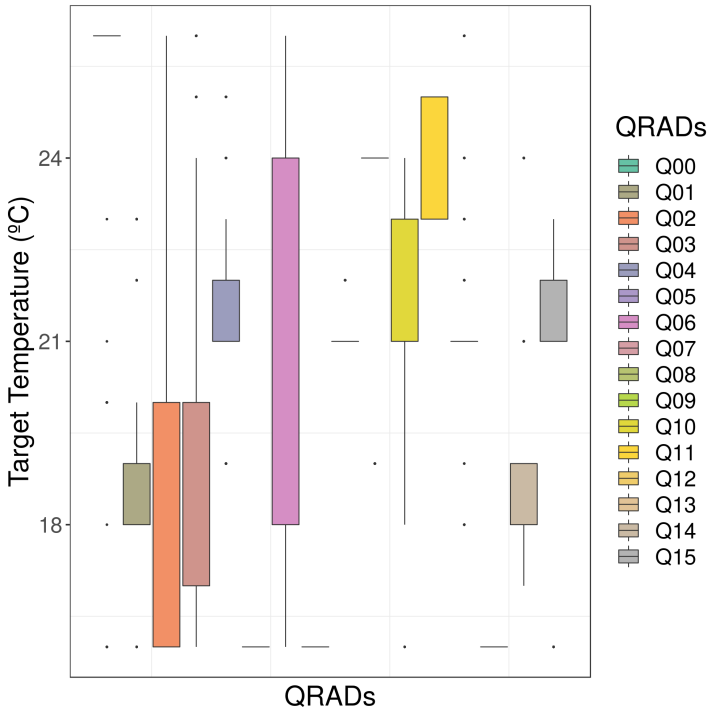
Figure 2.6 – Motherboard Temperature Distributions, for all QRADs, in the first range



- **CPU**:

  The Figure 2.9 shows that just one QRAD looks like that has changed from 35º to 58º C. There are 10 QRADs with a ceiling of 40º and other four inside 55º, 58º C. The remark

Figure 2.7 – Target Temperature Distributions, for all QRADs, in the first range



here is about the QRADs with the ceiling of 40º, which we suspect that were idle inside this range.

- **Heatsink**:

  The Figure 2.10 present distributions which look very spread in the range 30º, 55º C, with about 8 QRADs. But, there are some QRADs very restrict in the range 50º ,55º C. Here we can discuss again the relation between the ambient temperature and the other QRADs inside components, once the heatsink represent the merge of these temperatures. Here, it is possible to see that the heatsink achieve temperatures like 55º C, but the ambient temperature is not present in this range. It is also possible to see that the ambient temperature impacts in a high variation of the heatsink, as the box-plots shapes present, and emphasizes the difficult to produce and keep the targeted temperature.

- **Motherboard**:

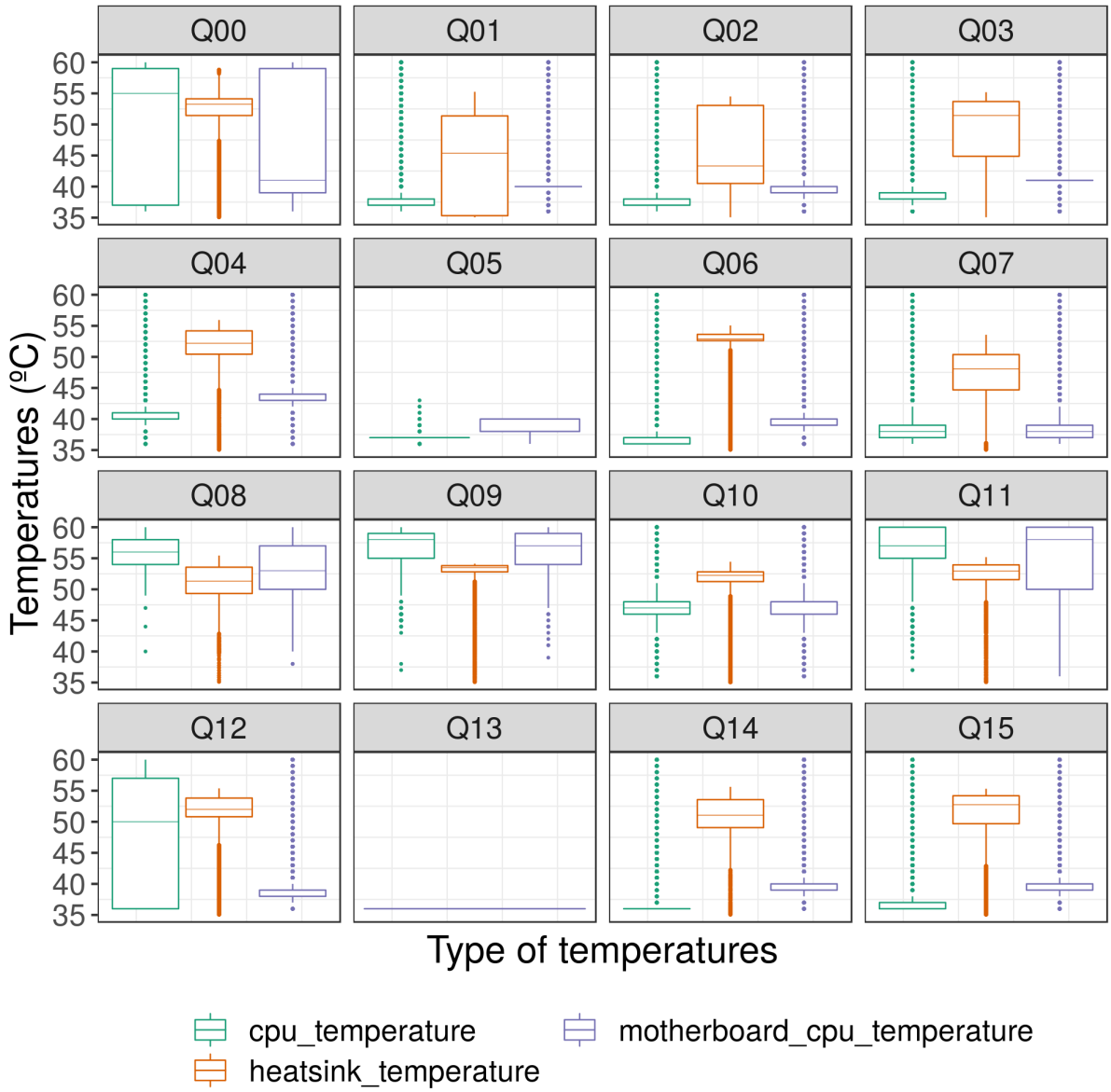
  From the Figure 2.11 is possible to see that the motherboard temperatures are very close to the CPU ones, as the well as the first range, representing the strong relation between both temperatures, CPU and motherboard.

2.1.6 The third range of temperatures

Since there are few machines achieving high temperatures, we would like to investigate them. Then, the Figure 2.12 presents the distributions of temperatures inside the range starting from 60º to 80º C, for all QRADs.

From the Figure 2.12 is possible to see that there is no target for this range, which means that there is no users that would like to be heated as much this range represents. Also, there is

Figure 2.8 – Distributions of Temperature Summarized by Box-Plots, for all QRADs, in the second range



no ambient or heatsink for this range, which means that nether any ambient that achieve this temperatures. We understood that the motherboard and CPU getting so hot means that they are doing to produce the heat required. Even that, the heatsink does no spread this temperature, since it is not present in this range of temperatures.

The following topics present the distributions by box-plots and density-plots for each type of temperature and all QRADs:

- **CPU**:

  The Figure 2.13 shows that there are more or less 8 QRADs with similar shape inside 63º, 75º C.

Figure 2.9 – CPU Temperature Distributions, for all QRADs, in the second range

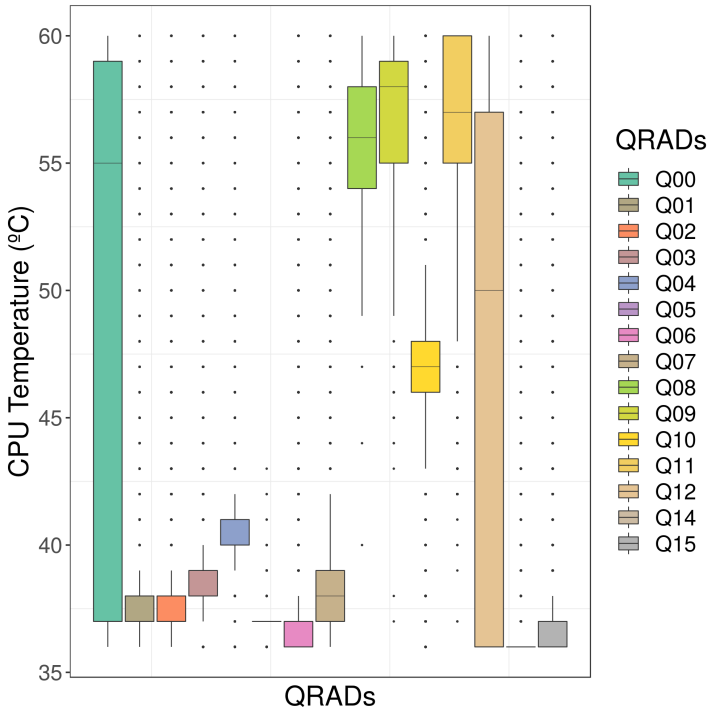
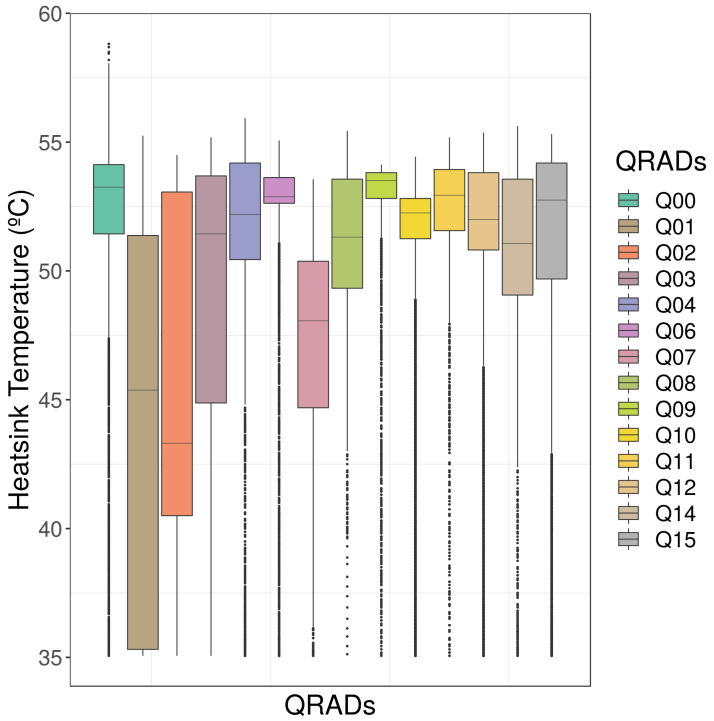


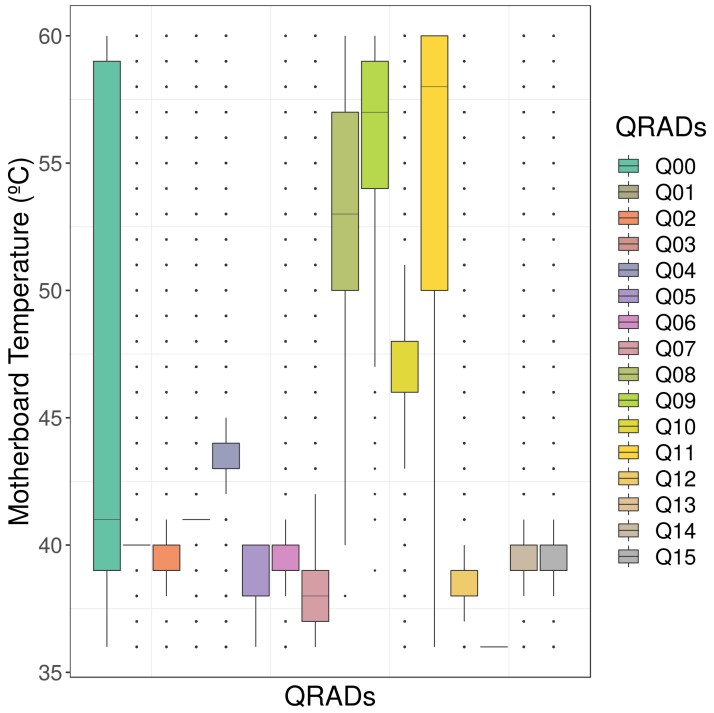
Figure 2.10 – Heatsink Temperature Distributions, for all QRADs, in the second range



- **Motherboard**:

  The Section 2.1.6 shows, again, that the motherboard temperature are very close to the CPU ones. It was a behavior followed in the three ranges of temperatures. Represent-

Figure 2.11 – Motherboard Temperature Distributions, for all QRADs, in the second range



ing that this relation is not affected by the increase or decrease of the temperatures, as expected.

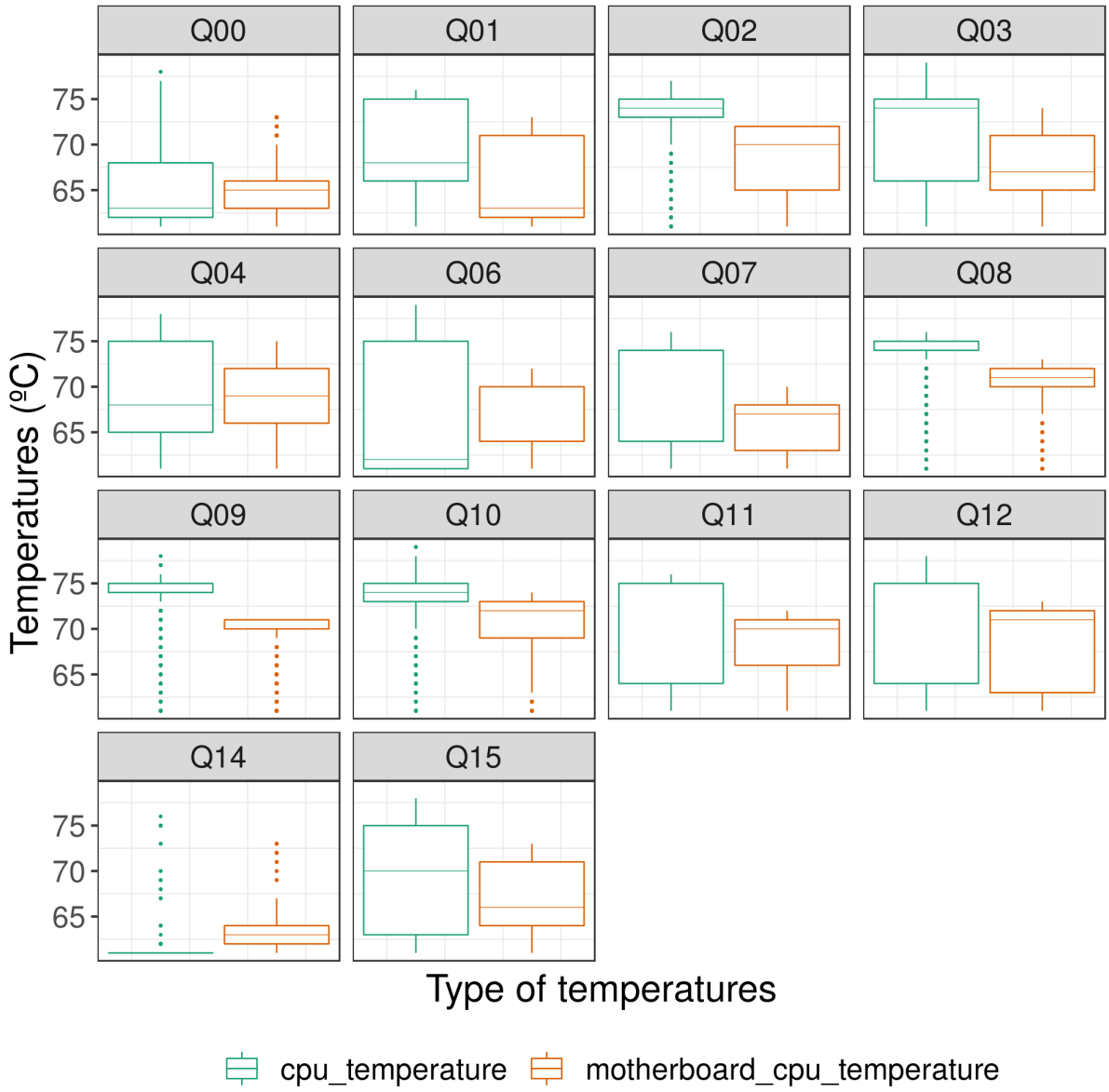
2.1.7 Stability

The previous analysis allowed us to question topics as the stability of the distributions, which means, the consequences of the heat produced by some component in another one. Does the heat produced by the CPU affects the ambient one? In this section we discussed this point and use some of the QRADs to illustrate a good and a doubt scenario.

How the temperatures change in order to deliver the target temperature by the ambient temperature? Looking to the shapes of each QRAD in the Figure 2.1, there are some ones which looks like very stable in terms of this point of view. For example, the Q13 has the target temperature fixed in 27º C and the other temperatures vary few. This one is a good example but it is the log with less data available, with 300 samples representing 50 minutes, while the other ones represent about 3 week. The Q07 represents the same good and expected behavior, as the Figure 2.15 shows. With exception of some outliers, it presents a small variation for the CPU, motherboard and heatsink temperature. These outliers could represent the periods or instants when a job finishes and other one is allocated to that QRAD. The Figure 2.15 shows in the left the full period of temperatures measured for this QRAD, and in the right we presented a slice of this period emphasizing the described behavior.

In contrast, it is possible to see in the Figure 2.1 that the Q00 presents its target temperatures concentrated about 25º C with exception of some outliers. But, the other distributions, cpu, motherboard and heatsink vary a lot. The Figure 2.16 presents, in the left, the distributions for all period measured, and in the right, it presents a slice of this period emphasizing that the

Figure 2.12 – Distributions of Temperature Summarized by Box-Plots, for all QRADs, in the third range



variation of the CPU, motherboard and heatsink temperatures does not affect the ambient one in this QRAD.

Then, we raised the hypothesis of a non very accurate mechanism for cooling when needed. We expected that since the cpu, motherboard and heatsink temperatures vary, the ambient one would vary in the same proportion, either for heating or for cooling. The Q00 show us the contrary, that the variation of the cpu, motherboard or heatsink temperatures do not affect the ambient one, in a direction of cooling. The same is observed for some other QRADs when analysed in this same sense. If by one side we can consider this behavior as expected since the QRADs are radiators and not air conditioners, by the other side it is very reasonable that since an user wants to reduce their ambient temperature, which was heated by the QRADs, this

Figure 2.13 – CPU Temperature Distributions, for all QRADs, in the third range

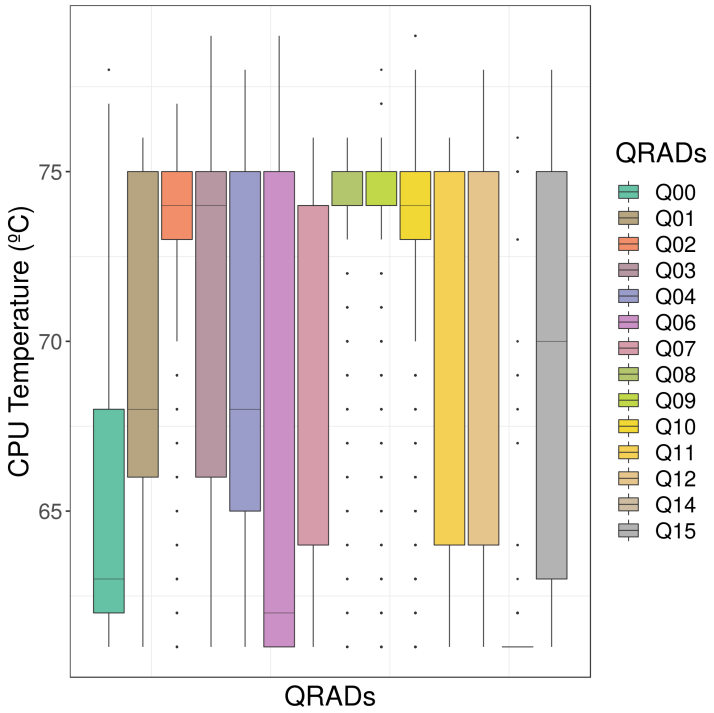
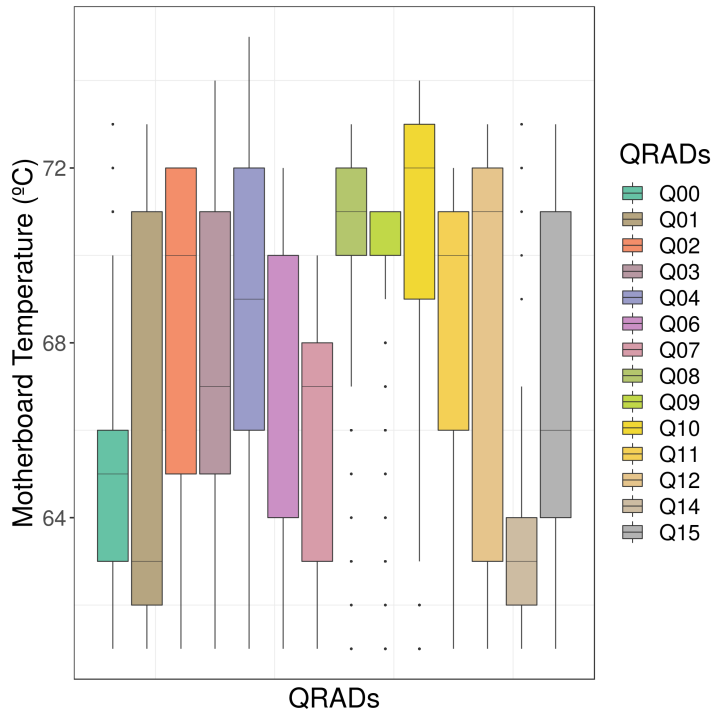


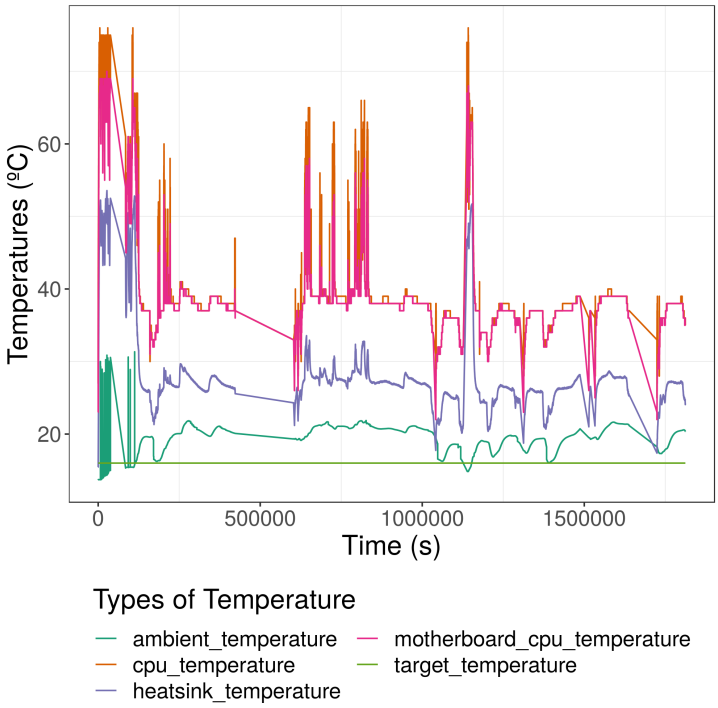
Figure 2.14 – Motherboard Temperature Distributions, for all QRADs, in the third range



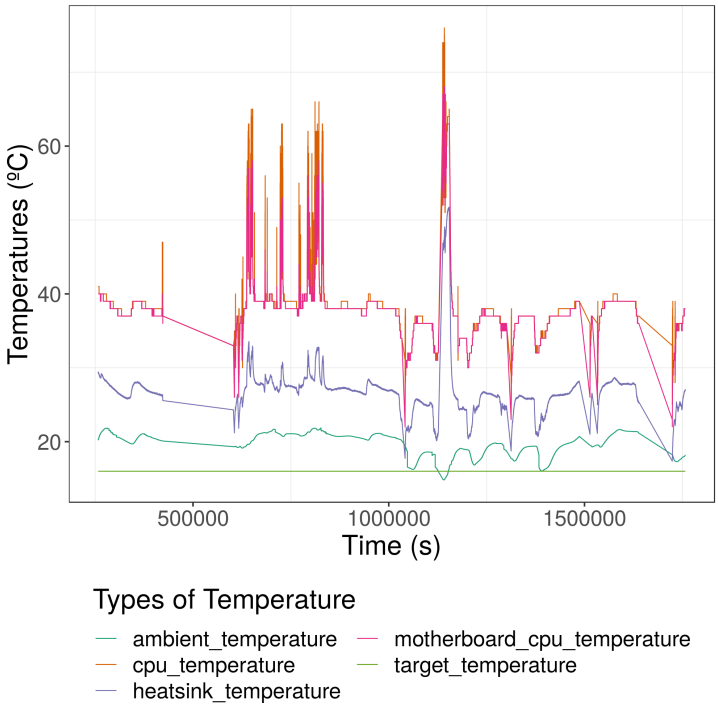
machine could do it.

Figure 2.15 – Temperature Distributions over Time, QRAD07

(a) Full Measurement.



(b) A Focused Interval of Time.

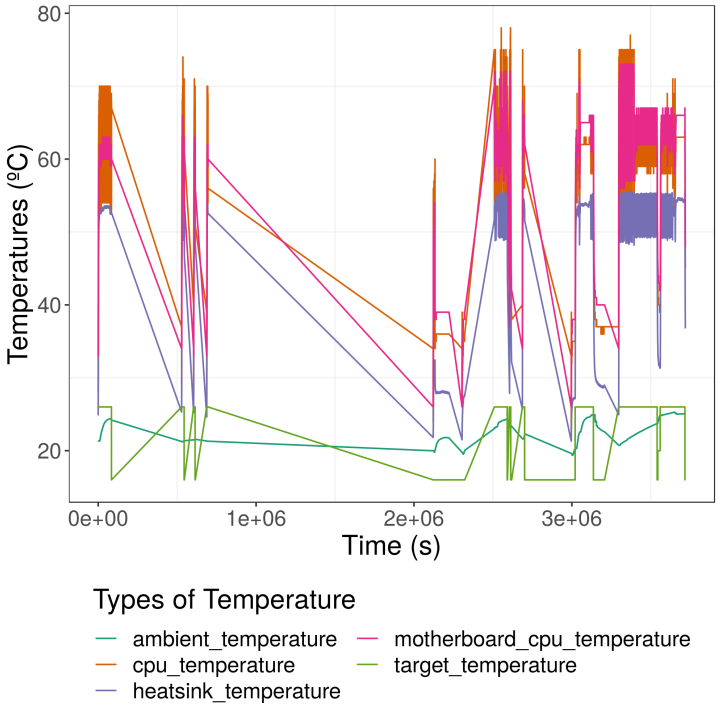


2.2 Categorization based on Distributions Analysis of Variances

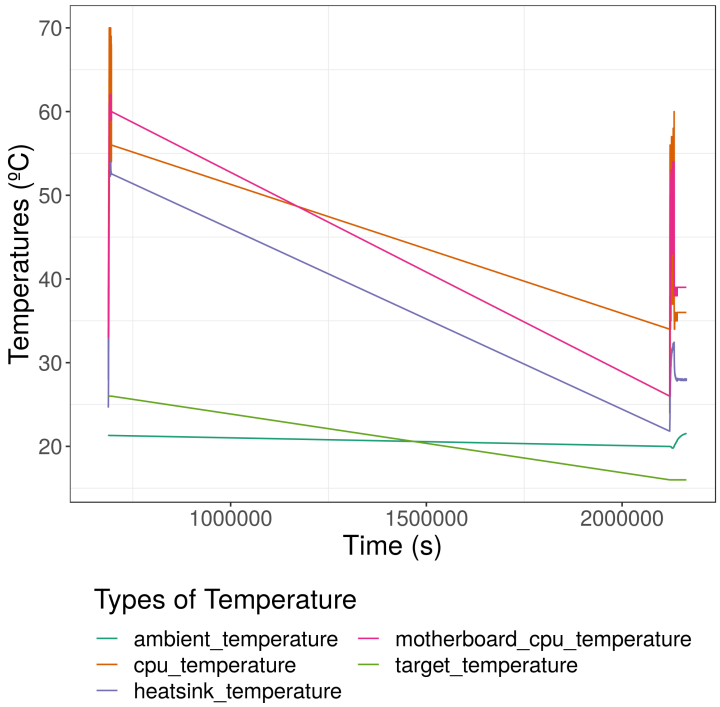
In this section we will present some results of our method to group/categorize the QRADS, based on the analysis of variances of their distributions.

Figure 2.16 – Temperature Distributions over Time, QRAD00

(a) Full Measurement.



(b) A Focused Interval of Time.



2.2.1 One-Way Analyses of Variance

When we performed the Analysis of Variance based on our data, we got an ANOVA model, with the following results:

Table 2.1 – Anova Summary

	Df	Sum	Mean	F value	Pr(>F)	
log	14	79288	5663	2163	<2e-16	***
Residuals	166664	436463	3			

—

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

From the Table 2.1, the *Pr(>F)* column represents the p-value, and the *Signific. code* measures their quality. The resulted p-value < 0.05 is very significant since it is close to 0. Which means that there are significant differences among the groups, for us, the QRADS. Then, we can conclude that is not possible to consider that all QRADS behave at the same way.

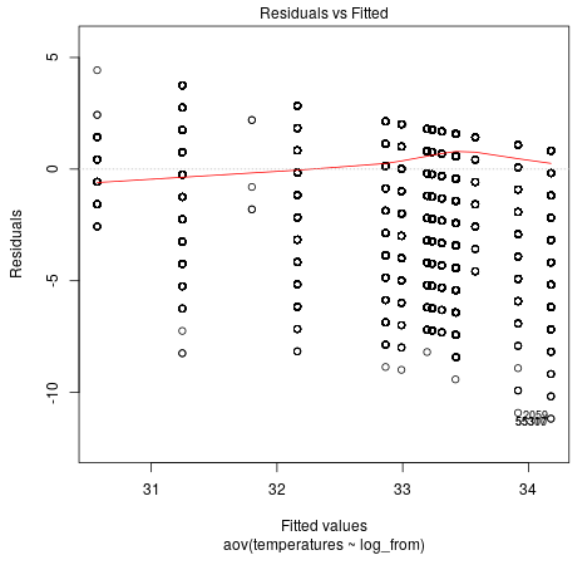
2.2.2 The requirements and relaxed metrics

To analyse the ANOVA requirements we investigated the residuals for the ANOVA model computed.

Homogeneity

When we verify the homogeneity of the data we are verifying if the distributions variances are equal or not. The Figure 2.17 shows the plot of the models' residuals.

Figure 2.17 – Residuals for homogeneity



Based on the Table 2.2 we can see that the data is not homogeneity, since the plot does not show its points well distributed.

We also used the Levene's Test, as a computational method to do this verification, presented in the Table 2.2

Table 2.2 – Levene's Test for Homogeneity of Variance (center = median)

	Df	F Value	Pr(>F)	
group	14	1327.7	<2.2e-16	***
	166664			

—

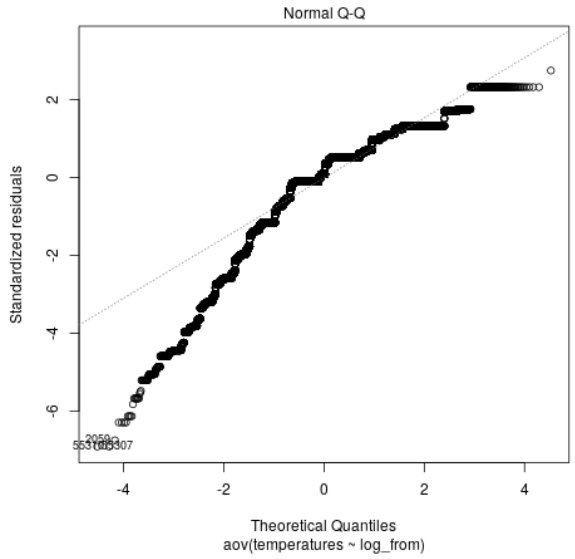
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Based on the Table 2.2 , the p-value shows that the homogeneity of the variance is not respected.

Normality

The Figure 2.18 shows the quantiles of the residuals plotted against the quantiles of the normal distribution. If our distribution would follow the normal one, all points would concentrate in the secondary diagonal line.

Figure 2.18 – Q-Q Plot of Normal Distributions over CPU Temperature Distributions



Then, looking to the Figure 2.18 we can conclude that the residuals are not normally distributed, then the analysed distribution does not follow the normal one, as well.

In addition, we used the Anderson-Darling normality test, that is a computational method to verify the normality of the data, which is presented in the Table 2.3.

Table 2.3 – Anderson-Darling normality test

A	p-value	
4705.4	« 2.2e-16	***

—

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

And based on the Table 2.3, the resulted p-value means that the normality of the distributions is not respected.

2.2.3 Kruskal-Wallis rank sum test

Although the two ANOVA requirements are not respected, we computed the same steps again using the Kruskal-Wallis Test, which is presented in the Table 2.4.

Table 2.4 – Kruskal-Wallis Rank sum Test

chi-squared	**df**	**p-value**
23804	14	« 2.2e-16

Then, from the Table 2.4, the resulted p-value < 0.05 means that there is significant differences among the QRADS, the same as the ANOVA did, but now we have the guaranties of the correct treatment of our data.

2.2.4 Pairwise T-Test

We performed the Pairwise T-Test to verify if there is a significant difference among the groups analysed by the Kruskal-Wallis, which is presented in the Table 2.5.

Table 2.5 – Pairwise comparisons using T-Test for the CPU temperatures, p: p-values < 0.05

	Q00	Q01	Q02	Q03	Q04	Q05	Q06	Q07
Q01	p	-	-	-	-	-	-	-
Q02	p	p	-	-	-	-	-	-
Q03	p	p	p	-	-	-	-	-
Q04	p	**0,20**	p	p	-	-	-	-
Q05	p	p	p	p	p	-	-	-
Q06	p	p	p	p	p	p	-	-
Q07	p	p	p	p	p	p	p	-
Q08	p	p	p	p	p	p	p	p
Q09	**0,48**	**0,87**	**0,83**	**0,51**	**0,88**	**0,15**	**0,76**	**0,99**
Q10	p	p	p	p	p	p	**0,08**	p
Q11	p	**0,06**	**0,05**	**0,67**	**0,07**	**0,50**	p	**0,12**
Q12	p	p	p	p	p	p	p	p
Q13	p	p	p	p	p	p	p	p
Q14	p	p	p	p	p	p	p	p
Q15	p	p	p	p	p	p	**0,87**	p

	Q08	Q09	Q10	Q11	Q12	Q13	Q14
Q01	-	-	-	-	-	-	-
Q02	-	-	-	-	-	-	-
Q03	-	-	-	-	-	-	-
Q04	-	-	-	-	-	-	-
Q05	-	-	-	-	-	-	-
Q06	-	-	-	-	-	-	-
Q07	-	-	-	-	-	-	-
Q08	-	-	-	-	-	-	-
Q09	p	-	-	-	-	-	-
Q10	p	**0,67**	-	-	-	-	-
Q11	p	**0,43**	-	-	-	-	-
Q12	p	**0,35**	p	p	-	-	-
Q13	p	**0,05**	p	**0,11**	p	-	-
Q14	p	**0,91**	p	**0,17**	p	p	-
Q15	p	**0,76**	**0,10**	p	p	p	p

Based on the Table 2.5, we were able to verify which QRAD can be taken as different and which one can not, compared with the others.

2.2.5 Categorization

We performed the first categorization step and the Table 2.6 shows the resulted groups for the CPU temperature in the first range.

Table 2.6 – QRADs grouped based on the CPU Temperature - Step 1

Groups	Components
Group 1	Q10, Q09, Q06, Q15
Group 2	Q11, Q09, Q14
Group 3	Q11, Q09, Q13
Group 4	Q09, Q12
Group 5	Q09, Q07, Q05, Q04, Q03, Q01, Q11
Group 6	Q09, Q06, Q10
Group 7	Q07, Q06, Q05, Q04, Q03, Q02, Q01, Q00, Q09
Group 8	Q02, Q00

Merge groups of the same type of temperatures

Since we found conflicts of elements (QRADs) among the groups in the Table 2.6 we needed to decide in each one this QRAD or QRADs should continue in. This way, we used the Algorithm 8 to decide in which group the QRAD contributes more to increase their homogeneity coefficient. Then, we reduced the groups from the Table 2.6 to the Table 2.7 presented below.

Table 2.7 – QRADs grouped based on the CPU Temperature - Step 2

Groups	Components
Group 1	Q09, Q07, Q05, Q04, Q03, Q01, Q11
Group 2	Q10, Q06, Q15
Group 3	Q02, Q00

And so on, performing the same steps for all type of temperatures we will get the following four other groups:

Table 2.8 – QRADs grouped based on the Ambient Temperature - Step 2

Groups	Composition
Group 1	Q13, Q15
Group 2	Q09, Q11

Table 2.9 – QRADs grouped based on the Target Temperature - Step 2

Groups	Composition
Group 1	Q07, Q05, Q13
Gtoup 2	Q00, Q09

Table 2.10 – QRADs grouped based on the Motherboard Temperature - Step 2

Groups	Composition
Group 1	Q09, Q07, Q06, Q05, Q04, Q02, Q01, Q00, Q10
Group 2	Q11, Q15

Table 2.11 – QRADs grouped based on the Heatsink Temperature - Step 2

Groups	Composition
Group 1	Q10, Q14
Group 2	Q09, Q11

Then, we have on these five tables, 2.7, 2.8, 2.9, 2.10 and 2.11 the categorization for each type of temperature.

Merge groups from different types of temperature

Grouping the five sets of groups presented in 2.7, 2.8, 2.9, 2.10 and 2.11 we computed the final groups, for each range of temperatures, presented in the Table 2.12, as follows:

Table 2.12 – QRADs grouped based on the Temperatures in the first range

Groups	Composition
Group 1	Q13, Q15
Group 2	Q09, Q07, Q03, Q01, Q11
Group 3	Q06, Q05, Q04, Q02, Q00, Q10

Then, we can attribute the following characteristics, for each group from the Table 2.12:

- **Group 1**: The QRADs with the Ambient Temperature concentrated in the 22.5º C with few variation,
- **Group 2**: The QRADs with similar Ambient, CPU, Motherboard and Heatsink at the same time,
- **Group 3**: The QRADs with similar CPU and Motherboard, which both vary from 30º to 35º C.

And repeating the same process for the other ranges, we got the groups presented in Table 2.13 and Table 2.14.

Table 2.13 – QRADs grouped based on all temperatures in the second range

Groups	Composition
Group 1	Q10, Q14
Group 2	Q09, Q08 Q11

Then, we can attribute the following characteristics, for each group from the Table 2.13:

- **Group 1**: The QRADs with similar Heatsink temperature, varying from 47.5º to 52.5º C,
- **Group 2**: The QRADs with similar CPU and Motherboard Temperatures varying from 50º to 60ºC, and Heatsink Temperatures, which varying from 50 to 55ºC.

Table 2.14 – QRADs grouped based on all temperatures in the third range

Groups	**Composition**
Group 1	Q03, Q12
Group 2	Q08, Q10
Group 3	Q01, Q04
Group 4	Q02, Q11

And we can attribute the following characteristics, for each group from the Table 2.14:

- **Group 1**: The QRADs with similar CPU Temperatures, from 60º to 75º C with high variation,
- **Group 2**: The QRADs with similar Motherboard Temperatures, varying from 68º to 70º C,
- **Group 3**: The QRADs with similar CPU Temperatures, varying from 65º to 75º C with median about 67ºC,
- **Group 4**: The QRADs with similar Motherboard Temperatures, varying from 67.5º to 72.5º C.

In other words, the last three tables: Table 2.12, Table 2.13 and Table 2.14 represent a possible categorization based on all types of temperature, for each range defined.

— 3 —
Conclusions and Future Work

In this chapter we performed a first step towards understanding the temperature characteristics of the smart heaters of Qarnot Computing. We analyzed the logs of sixteen smart heaters and a first observation is the large number of noisy/measurement errors data present in the logs. This can indicate that the measurement system of the smart heaters needs to be improved in order to give more accurate data.

It was noticed the importance to investigate the Qarnot's data behavior splitting the data in ranges of temperatures, to compare their Smart Heaters during different periods, which means during the process to deliver different range of temperatures. With that, we saw that there are some QRADs which has its target temperature almost constant, but the other types of temperatures varies a lot, even with this fact. We assumed that the variation was because the reallocation of jobs to the machines over the time. But, there are other possibilities that should be investigated, for example, looking to this variations during the execution of the same job. Looking to the different ranges we also saw that the heatsink temperature is present in the second range, above 35º C, even if there is no targeted or ambient temperature on that range. It indicates that even if the heatsink is over than 35º C, when the air is spread in the ambient temperature, it does not changes so much. Then, because of these and other observations pointed during the text we believe that to split the distributions among ranges was a good approach. For further works we believe that is possible to investigate the relation of the temperatures when increasing and decreasing, in order to see how each component gets warm whenever the other ones are also doing it.

We also performed a grouping of the temperature characteristics of the smart theaters. These categories may facilitate conceiving temperature models for the smart heaters, since it would be possible to create models for the categories and not for each smart heater individually. For future work, we will include other measurements in our analysis, notably data regarding the power consumed and we will also move towards creating temperature models of these smart heaters. From the final groups for each range of temperature we noticed that in the first range it was possible to group more QRADs thant in the other ranges. We consider that is because of what this range represents, in other words, the first range includes the temperatures under 35º C which in terms of computation can mean that the QRAD was not running anything at during that time, then its easier to caracterize the machines as similar. We believe that future steps that could improve the categorization process can be (i) the outliers removal, (ii) the merge of groups containing the same QRADs instead of decides in which one the QRAD should belongs to.

Part IV

General Conclusions

— 1 —
General Remarks and Acquired Knowledge

This work was done in two parts, focusing in two individual topics, at first, but since they belong to the same context, they can be merged and took as two steps among an the same environment, an Edge platform simulation and its data analysis. Many challenges are involved in this context, and in particular, in this work. It was presented for each part a focused conclusion and some future remarks, as Chapter 5 for Part II and Chapter 3 for Part III. In this chapter we present general discussions, thinking about the global view of this work, and the acquired knowledge as well.

1.1 Simulated Platforms

One of the main gaols of this work, presented in Part II was the development of the simulated platform, which was already in progress, but this work contributed to its conclusion. For that it was necessary the understoodthe simulators SimGird and Batsim. These both simulators, one built on the top of the other, allow researchers to implement schedulers and simulations for general Edge Platforms, developing each component separately, step by step. In addition, it is possible to get many types of outputs and follow the simulations logs, which is very useful to the validation of the desired platform and the general analysis. Its important to emphasize here the importance of such tool, once that it allow researchers and companies to implement accurate simulations to then study modifications without putting in risk their production systems.

1.2 Edge Platforms

It is very interesting and challenging the behavior that such platforms have been developed and growth, also it is challenging the necessity of multiple computations by machines far from centralized servers, at the same time with its machine/ devices own behaviors with levels of autonomous intelligence, such as smart phones, smart watches, smart heaters in our context, and so on. Then, it is important to study how jobs and data sets are allocated and transmitted among the network. It is also important to know that the components on Edge Platforms are intermittent and if by one side they have got smart and capable to perform their own computations, by the other side they are limited when referred to disk, processor, memory, battery and mainly dependent of connection with a internet network. Then, this context are followed by such high heterogeneity and has growth in this terms and many others discussed in Chapter 2.

1.3 Job Allocation

Scheduling or job allocation is a very important process among the management of jobs, data sets and resources because it impacts directly in many other things as computation performances and energy/ power consumption. When referred about HPC and Cloud jobs its know many techniques and algorithms already well validated and improved, but when it is inside the context of Edge Computing we find many challenges, as multiple layers of network infrastructure or many process decisions nodes, whose need to be aligned and under communication as much as possible. One possibility is to use one of those known algorithms, with combinations and modifications regarding the heterogeneities and novelties of the Edge Platforms.

1.4 Scheduling Metrics

In order to evaluate the performance and the differences among the different types of job allocation policies, there are different metrics. For example, the waiting time, to analyse if the scheduler is improving the time that the jobs wait to be executed since they were received into the system. Another example is the bounded slowdown, which verifies if the jobs waiting time are proportional with its sizes. In addition to the heterogeneity and the non default strucutre of the Edge platforms, these metric should be also adapted.

1.5 Data analysis

In order to investigate some data frame its is important to follow some basics steps to do not get wrong or biased conclusions. An very important step is to know the data frame, it characteristics, the information provided there, how the measurements were done, how accurate is the sensors or the mechanisms which measured the data available. After, it is very import to recognise and remove the measurement errors. In general, all the data analysis process is very delicate in each step or decision, which should be done very carefully and argued.

Even when performing the analysis of the results from the simulated platform, or when performing the descriptive analysis of the Qarnot Computing logs, it was possible to acquire skills such data investigation, which follow skills to determine/ create workflows and to ask questions that could direct a remaining research. Of course, all of those should be followed by constant questions about the veracity and accuracy of the data, which helps in the development of skill as research, comparison, and adjustment of different methodologies to achieve a common goal. Such skills allowed us to based on an analysis of variance, develop a methodology to categorize the Smart Heaters based on their temperature distributions, which follows an algorithm created during this work.

1.6 The Qarnot Computing Use Case

When applied all the concepts described above in an use case, it is also necessary to understand this use case, its particularities, behaviors, objectives, necessities and environment. The Qarnot Computing as a company that run HPC/ Cloud jobs in machines which works as Smart Heaters, innovated, when did these tasks managing the machines' and the environment. For that they found many challenges and one of the highest is the translation of heat requirements

in amount of jobs that should be ran into the machines. How it is possible to know the size of the job that should be scheduled to some machine, based on the heat required by a home/ smart heater user? They also have other challenges based on the accuracy of the temperature and how to maintain that, since they do not control the final user. This one could turn on its machine and require a high temperature which will turn the scheduling to pay attention in there, and then, the user could just turn off the machine, because he changed mind.Then, the temperature delivered were not the required one, which will be part of the logs and will be difficult, in the future, to identify why does the required temperature was not achieved. It will also affect the scheduler, that once dispatched a job, will need to manage it again.

We investigated if the temperatures' distributions make sense among each other, which means, if the different components that get warm inside the machines, affect each other. We tried to search into the logs, the presence of an expected behavior, and to understand the not expected ones. By one side we achieved some possible conclusions remarked on this manuscript, but by the other, we analysed just 16 machines among 300 other, which compose the Qarnot platform.

A further step of the work group is to build a model of temperature based on its logs in other to estimate the amount of jobs needed based on the temperature required. Then, it is intended to use this model inside the simulator, to go one step further in the validation of the simulated platform, which is still ongoing because of these challenges: the simulator has as power consumption estimator, but not a temperature one, properly. And since the simulator receives just the information extracted from the Qarnot Platform, which does not include their users' behaviors, as doors or windows opening and closing, it is not possible to compare the exactly temperature estimated in the simulator and the logs ones. What is possible to do with the simulator, as a very good characteristic, is the implementation and simulation of jobs allocations policies, which can be very useful to researchers or companies, as the Qarnot Computing, interested in the field.

1.7 Tools and Programming Language

What allowed us to do everything that was presented, to work on the development of the simulated Edge Platform and all the other analysis, was the utilization of different type of tools or programming languages, for each one of the different goals and steps present in this work. The simulated Edge platform was built on top of other two simulators, the Batsim and the Simgrid. Even if we did not develop anything directly in the Simgrid, we used the Batsim, and for that we needed to understand how does the Batsim communicates with the Simgrid, to debug and follow the logs. For the usage of the Batsim and PyBatsim we used the programming language Python. To mange its results we also utilized this programming language, but to manage all the graphics and statistics based on them we used the R Language. But, before getting any full script in R, we developed its functions, step by step using the Jupyter Notebook, a personalized notebook which allows the usage of Python, R and Julia languages. Since we needed to automatize the execution of the simulations, the manipulations of its results, the statistical and descriptive analysis, we developed scripts on Linux Bash. Then, these scripts ran the whole environment utilizing the other scripts in Python and R.

To perform the descriptive data analysis part, we utilized mainly the R language and the Jupyter Notebook, which the goals to investigated in very well split steps, the Qarnot Computing's logs.

1.8 Scientific Research

Even if the first part of this work was very practical, it demanded from us the study of the state of the art of its context, Edge Computing, and its contributions looks like good to the field, since the simulator can be used and has been developed with the goal of helps researchers to investigate job allocation policies in simulated Edge Platforms. This work also presents an use case very exciting because of its novelty, which can also be considered as scientific contribution.

Looking by the side of the reproducibility, this work aimed to helped with a free license tool, the simulator based on Batsim/ Simgrid, and all scripts and methodology utilized so far are available on git based repositories, with some examples of input data. For reasons of privacy, it was not possible to keep available the full input data from the Qarnot Computing to protect its users and the licenses which the company is based.

All of were presented in the full manuscript, also pointed in the Chapter 1 should be understood as acquired knowledge, but in addition, in this chapter we point some other topics, not necessary about the content which was presented, but more about the skills, techniques and general knowledge learned so far.

Bibliography

[1] Qarnot computing. https://www.qarnot.com.

[2] Simgrid publications. http://simgrid.gforge.inria.fr/publications.html.

[3] A. Ahmed and E. Ahmed. A survey on mobile edge computing. In *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, pages 1–8, Jan 2016.

[4] Kento Aida. Effect of job size characteristics on job scheduling performance. In Dror G. Feitelson and Larry Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 1–17, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[5] Farah Ait Salaht, Frédéric Desprez, Adrien Lebre, Charles Prud'Homme, and Mohamed Abderrahim. Service Placement in Fog Computing Using Constraint Programming. In *SCC 2019 - IEEE International Conference on Services Computing*, pages 1–9, Milan, Italy, July 2019. IEEE.

[6] Bill Allcock, Joe Bester, John Bresnahan, Ann L Chervenak, Ian Foster, Carl Kesselman, Sam Meder, Veronika Nefedova, Darcy Quesnel, and Steven Tuecke. Data management and transfer in high-performance computational grid environments. *Parallel Computing*, 28(5):749–771, 2002.

[7] David P Anderson. Boinc: A system for public-resource computing and storage. In *proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 4–10. IEEE Computer Society, 2004.

[8] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[9] Luiz Bittencourt, Roger Immich, Rizos Sakellariou, Nelson Fonseca, Edmundo Madeira, Marilia Curado, Leandro Villas, Luiz DaSilva, Craig Lee, and Omer Rana. The internet of things, fog and cloud continuum: Integration and challenges. *Internet of Things*, 3-4:134 – 155, 2018.

[10] Flavio Bonomi, Rodolfo A. Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing, MCC@SIGCOMM 2012, Helsinki, Finland, August 17, 2012*, pages 13–16, 2012.

[11] A. Brogi and S. Forti. QoS-Aware Deployment of IoT Applications Through the Fog. *IEEE Internet of Things Journal*, 4(5):1185–1192, Oct 2017.

[12] Henri Casanova, Arnaud Giersch, Arnaud Legrand, Martin Quinson, and Frédéric Suter. Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917, June 2014.

[13] Bruno Donassolo, Ilhem Fajjari, Arnaud Legrand, and Panayotis Mertikopoulos. Fog Based Framework for IoT Service Provisioning. In *IEEE CCNC*, January 2019.

[14] Pierre-François Dutot, Michael Mercier, Millian Poquet, and Olivier Richard. Batsim: a Realistic Language-Independent Resources and Jobs Management Systems Simulator. In *20th Workshop on Job Scheduling Strategies for Parallel Processing*, Chicago, United States, May 2016.

[15] A. Essafi, D. Trystram, and Z. Zaidi. An efficient algorithm for scheduling jobs in volunteer computing platforms. In *2014 IEEE International Parallel Distributed Processing Symposium Workshops*, pages 68–76, May 2014.

[16] Dror G. Feitelson. Metrics for parallel job scheduling and their convergence. In Dror G. Feitelson and Larry Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 188–205, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[17] Harshit Gupta, Amir Vahid Dastjerdi, Soumya Ghosh, and Rajkumar Buyya. ifogsim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments. *Software: Practice and Experience*, 06 2016.

[18] F. C. Heinrich, T. Cornebize, A. Degomme, A. Legrand, A. Carpen-Amarie, S. Hunold, A. Orgerie, and M. Quinson. Predicting the energy-consumption of mpi applications at scale using only a single node. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 92–102, Sep. 2017.

[19] Lu Huang, Hai-shan Chen, and Ting-ting Hu. Survey on resource allocation policy and job scheduling algorithms of cloud computing1. *Journal of Software*, 8, 02 2013.

[20] Hameed Hussain, Saif Ur Rehman Malik, Abdul Hameed, Samee Ullah Khan, Gage Bickler, Nasro Min-Allah, Muhammad Bilal Qureshi, Limin Zhang, Wang Yongji, Nasir Ghani, Joanna Kolodziej, Albert Y. Zomaya, Cheng-Zhong Xu, Pavan Balaji, Abhinav Vishnu, Fredric Pinel, Johnatan E. Pecero, Dzmitry Kliazovich, Pascal Bouvry, Hongxiang Li, Lizhe Wang, Dan Chen, and Ammar Rayes. A survey on resource allocation in high performance distributed computing systems. *Parallel Computing*, 39(11):709 – 736, 2013.

[21] A. Lebre, J. Pastor, A. Simonet, and M. Südholt. Putting the next 500 vm placement algorithms to the acid test: The infrastructure provider viewpoint. *IEEE Transactions on Parallel and Distributed Systems*, 30(1):204–217, Jan 2019.

[22] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys Tutorials*, 19(4):2322–2358, Fourthquarter 2017.

[23] Ruben Mayer, Leon Graser, Harshit Gupta, Enrique Saurez, and Umakishore Ramachandran. Emufog: Extensible and scalable emulation of large-scale fog computing infrastructures. In *FWC*, pages 1–6. IEEE, 2017.

[24] Jie Meng, Samuel McCauley, Fulya Kaplan, Vitus J. Leung, and Ayse K. Coskun. Simulation and optimization of hpc job allocation for jointly reducing communication and cooling costs. *Sustainable Computing: Informatics and Systems*, 6:48 – 57, 2015. Special Issue on Selected Papers from 2013 International Green Computing Conference (IGCC).

[25] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262, 2015.

[26] Mohammed Islam Naas, Philippe Raipin Parvedy, Jalil Boukhobza, and Laurent Lemarchand. iFogStor: An IoT Data Placement Strategy for Fog Infrastructure. In *ICFEC'17*, pages 97–104, 2017.

[27] Y. Ngoko, N. Saintherant, C. Cerin, and D. Trystram. Invited paper: How future buildings could redefine distributed computing. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1232–1240, May 2018.

[28] C. Pahl and B. Lee. Containers and clusters for edge cloud architectures – a technology review. In *2015 3rd International Conference on Future Internet of Things and Cloud*, pages 379–386, Aug 2015.

[29] S. M. Parikh. A survey on cloud computing resource allocation techniques. In *2013 Nirma University International Conference on Engineering (NUiCONE)*, pages 1–5, Nov 2013.

[30] Millian Poquet. *Simulation approach for resource management. (Approche par la simulation pour la gestion de ressources)*. PhD thesis, Grenoble Alpes University, France, 2017.

[31] Muhammad Bilal Qureshi, Maryam Mehri Dehnavi, Nasro Min-Allah, Muhammad Shuaib Qureshi, Hameed Hussain, Ilias Rentifis, Nikos Tziritas, Thanasis Loukopoulos, Samee U. Khan, Cheng-Zhong Xu, and Albert Y. Zomaya. Survey on grid resource allocation mechanisms. *Journal of Grid Computing*, 12(2):399–441, Jun 2014.

[32] M. Randles, D. Lamb, and A. Taleb-Bendiab. A comparative study into distributed load balancing algorithms for cloud computing. In *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, pages 551–556, April 2010.

[33] A. S. M. Rizvi, T. R. Toha, M. M. R. Lunar, M. A. Adnan, and A. B. M. A. A. Islam. Cooling energy integration in simgrid. In *2017 International Conference on Networking, Systems and Security (NSysS)*, pages 132–137, Jan 2017.

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

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

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

[37] W. Shi and S. Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, May 2016.

[38] Olena Skarlat, Matteo Nardelli, Stefan Schulte, Michael Borkowski, and Philipp Leitner. Optimized IoT Service Placement in the Fog. *SOC*, 11(4):427–443, Dec 2017.

[39] C. Sonmez, A. Ozgovde, and C. Ersoy. Edgecloudsim: An environment for performance evaluation of edge computing systems. In *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 39–44, May 2017.

[40] Ye Xia, Xavier Etchevers, Loïc Letondeur, Thierry Coupaye, and Frédéric Desprez. Combining Hardware Nodes and Software Components Ordering-based Heuristics for Optimizing the Placement of Distributed IoT Applications in the Fog. In *Proc. of the ACM SAC*, pages 751–760, 2018.

[41] Ashkan Yousefpour, Ashish Patil, Genya Ishigaki, Jason P. Jue, Inwoong Kim, Xi Wang, Hakki C. Cankaya, Qiong Zhang, and Weisheng Xie. QoS-aware Dynamic Fog Service Provisioning. 2017.

[42] Xuezhi Zeng, Saurabh Kumar Garg, Peter Strazdins, Prem Prakash Jayaraman, Dimitrios Georgakopoulos, and Rajiv Ranjan. Iotsim. *J. Syst. Archit.*, 72(C):93–107, January 2017.

[43] Ben Zhang, Nitesh Mor, John Kolb, Douglas Chan, Ken Lutz, Eric Allman, John Wawrzynek, Edward Lee, and John Kubiatowicz. The Cloud is Not Enough: Saving IoT from the Cloud. In *HotStorage*, 2015.