THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

Clément MOMMESSIN

Thèse dirigée par **Denis TRYSTRAM**
et codirigée par **Giorgio LUCARELLI**

préparée au sein du **Laboratoire d'Informatique de Grenoble**
et de l'École Doctorale **MSTII**

Gestion Efficace des Ressources dans les Plates-formes Hétérogènes

Efficient Management of Resources in Heterogeneous Platforms

Thèse soutenue publiquement le **11 décembre 2020**,
devant le jury composé de :

Olivier BEAUMONT
Directeur de Recherche, LaBRI, Inria, France, Examinateur

Anne BENOIT
Maître de Conférences, ENS Lyon, LIP, Lyon, France, Rapportrice

Noël DE PALMA
Professeur, LIG, Université Grenoble Alpes, France, Président

Giorgio LUCARELLI
Maître de Conférences, LCOMS, Université de Lorraine, Metz, France, Co-encadrant de thèse

Yanik NGOKO
Chargé de Recherche, Université Paris 13, et Ingénieur de Recherche, Qarnot Computing, France, Invité

Krzysztof RZADCA
Associate Professor, Institute of Informatics, University of Warsaw, and Google, Poland, Rapporteur

Denis TRYSTRAM
Professeur des Universités, LIG, Grenoble INP, France, Directeur de thèse

> *I solemnly swear that I am up to no good*
>
> — **Messrs. Moony, Wormtail, Padfoot, and Prongs**

Remerciements

(Acknowledgments)

Je dédie ces quelques lignes à Dobby, l'elfe libre...

Tout d'abord, je souhaite remercier mes encadrant et directeur, Giorgio LUCARELLI et Denis TRYSTRAM, qui m'ont accueilli, porté, supporté et aidé durant toutes ces années. Je remercie bien entendu également tous les membres du jury, et en particulier Anne BENOIT et Krzysztof RZADCA pour avoir réussi à lire l'ensemble de ce manuscrit et proposé des commentaires éclairés.

Mes remerciements vont ensuite à l'ensemble des équipes jumelles DATAMOVE et POLARIS – a.k.a. POLAMOVE – pour toute cette animation dans les couloirs, ces discussion au petit matin, ces goûters post-séminaires. Merci d'avoir fondé une aussi grande famille. Famille gérée par une main de maître par Annie, super Annie, qui fait un travail incroyable tous les jours pour tous nous aider et sans qui la boutique ne tournerait pas aussi bien. Merci aussi à tous mes collègues de bureau, de passage ou de long terme : merci aux vieux, Raphaël, Millian, Michael, au nouveau, Quentin, à Alexis qui nous a quittés trop vite, à Flora pour ces séances d'origami et de chasse au trésor. Muito obrigado Andrei, ευχαριστω Kimon και Mitsos, danke und gesundheit Malin. Merci également à Vincent, l'infiltré du bureau d'à côté, pour son imagination débordante, et bien sûr à tous les autres que je ne nommerai pas, stagiaires, doctorants, post-docs, ingénieurs, permanents des bureaux voisins. Enfin, *last but not least*, un énorme merci à Danilo, qui a les cheveux comme une piscine, et à Salah, Sa Luminosité, pour tous ces moments partagés au bureau. Je suis désolé d'avoir fait échouer un de tes objectifs les plus chers, tu n'auras pas réussi à me faire boire du café.

Plus généralement, un grand merci à la team Batsim, à tous les membres du groupe Info Sans Ordi de Grenoble et de Navarre, aux nombreux membres du club Rock de Grenoble INP, et puis à tous mes amis, qu'ils proviennent de l'équipage du Capitaine, de la petite bourgade de Claix ou de contrées plus lointaines.

Finalement, un grand merci à ma famille, qui pourra dire, je le pense, que ces années de thèse m'ont beaucoup raplauti.

This work is supported by the ANR GRECO project 16-CE25-0016-01.

The work presented in Chapter 6 was partially supported by FAPESP (São Paulo Research Foundation, grant #2012/23300-7) and ANR Moebus Project.
All the experiments of this chapter were performed on the Froggy platform of the CIMENT infrastructure[1], which is supported by the Rhône-Alpes region (GRANT CPER07_13 CIRA) and the Equip@Meso project (reference ANR-10-EQPX-29-01) of the program “Investissements d’Avenir” supervised by the French Research Agency (ANR).

[1] `https://ciment.ujf-grenoble.fr`

Abstract / Résumé

Abstract

The world of Information Technology (IT) is in constant evolution. With the explosion of the number of digital and connected devices in our everyday life, the IT infrastructures have to face an ever growing amount of users, computing requests and data generated. The Internet of Things has seen the development of computing platforms at the edge of the network to bridge the gap between the connected devices and the Cloud, called the Edge Computing. In the domain of High Performance Computing, the parallel programs executed on the platforms require always more computing power in a search for improved performances. Besides, we observed in the past years a diversification of the hardware composing these infrastructures. This complexification of the (network of) computing platforms pose several optimisation challenges that can appear at different levels. In particular, it led to a need for better management systems to make an efficient usage of the heterogeneous resources composing these platforms.

The work presented in this thesis focuses on resource optimisation problems for distributed and parallel platforms of the Edge Computing and High Performance Computing domains. In both cases, we study the modelling of the problems and propose methods and algorithms to optimise the resource management for better performance, in terms of quality of the solutions. The problems are studied from both theoretical and practical perspectives. More specifically, we study the resource management problems at multiple levels of the *Qarnot Computing* platform, an Edge Computing production platform mostly composed of computing resources deployed in heaters of smart-buildings. In this regard, we propose extensions to the Batsim simulator to enable the simulation of Edge Computing platforms and ease the design, development and comparison of data and job placement policies in such platforms. Then, we design a new temperature prediction method for smart-buildings and propose a formulation of a new scheduling problem with two agents on multiple machines.

In parallel, we study the problem of scheduling applications on hybrid multi-core machines with the objective of minimising the completion time of the overall application. We survey existing algorithms providing performance guarantees on the constructed schedules and propose two new algorithms for different settings of the problem, proving performance guarantees for both. Then, we conduct an experimental campaign to compare in practice the relative performance of the new algorithms with existing solutions in the literature.

Résumé

Le monde des Technologies de l'Information (IT) est en constante évolution. Avec l'explosion du nombre d'appareils numériques et connectés dans notre vie de tous les jours, les infrastructures IT doivent faire face à une constante augmentation du nombre d'utilisateurs, de requêtes informatiques et de données générées. L'Internet des Objets a vu le développement de plates-formes de calcul en bordure du réseau pour combler l'écart entre les appareils connectés et le Cloud, appelé le Edge Computing. Dans le domaine du Calcul à Haute Performance, les programmes parallèles exécutés sur les plates-formes demandent toujours plus de puissance de calcul à la recherche d'une amélioration des performances. De plus, il a été observé au cours des dernières années une diversification des composants matériels dans ces infrastructures. Cette complexification des (réseaux de) plates-formes de calculs pose plusieurs problèmes d'optimisation qui peuvent apparaître à divers niveaux. En particulier, cela a mené au besoin de meilleurs systèmes de gestion pour une utilisation efficace des ressources hétérogènes qui composent ces plates-formes.

Le travail présenté dans cette thèse se focalise sur des problèmes d'optimisation de ressources pour les plates-formes parallèles et distribuées du Calcul à Haute Performance et du Edge Computing. Dans les deux cas, nous étudions la modélisation des problèmes et nous proposons des méthodes et des algorithmes de gestion de ressources pour de meilleures performances. Les problèmes sont étudiés à la fois sur des plans théoriques et pratiques. Plus spécifiquement, nous étudions les problèmes de gestion de ressources à différents niveaux de la plate-forme *Qarnot Computing*, une plate-forme de production Edge principalement composée de ressources de calculs déployées dans des radiateurs de bâtiments intelligents. Pour cela, nous proposons des extensions au simulateur Batsim pour permettre la simulation de plates-formes Edge et pour faciliter le design, le développement et la comparaison de politiques de placement de données et de tâches sur de telles plates-formes. Ensuite, nous proposons une nouvelle méthode de prédiction de la température pour des bâtiments intelligents et nous formulons un nouveau problème d'ordonnancement à deux agents sur machines multiples.

En parallèle, nous étudions le problème d'ordonnancement d'applications sur machines multi-cœur hybrides dont l'objectif est la minimisation du temps total de complétion de l'application. Nous faisons une revue des algorithmes existants avec des garanties de performance, puis nous proposons deux nouveaux algorithmes pour différentes variantes du problème et nous donnons des preuves de leur garanties de performance. Enfin, nous conduisons une campagne expérimentale pour compa-

rer la performance relative de nos algorithmes avec des solutions existantes de la littérature.

Contents

1 Introduction

Since the dawn of time, mankind is obsessed with optimisation. It started with the creation of tools from silex stones with cutting edges, bows and arrows to make hunting easier. Then, the wheel appeared, people invented means of transportation, tasks were automatised. Iteratively, humans developed tools and improved their technique in the search for boosting their efficiency to mainly serve one purpose. Why would a caveman decide to go bow hunting? Why would a farmer in Middle Age transport a haystack on a wheel cart? Why would *you* drive 10 km/h faster than the speed limit on an empty country road? The reason is to perform a task more easily, faster, to *save time* and/or *energy*. Hence the optimisation.

Now, do not be mistaken! This manuscript does not discuss anthropological questions about the race of humans against the time. We are here to talk about optimisation in computer science, and, more specifically, about the efficient management of jobs and resources in heterogeneous distributed and parallel computing platforms.

1.1 Resource and Job Management in Heterogeneous Platforms

Since the emergence of large-scale distributed and parallel platforms in the past decades, we have observed a diversification of the computing units used in such systems. With the integration of hardware accelerators, such as *General Purpose Graphical Processing Units* (GPUs), that are often combined with multiple *Central Processing Units* (CPUs) on the same chip sharing the same common memory [Lee+10], the platforms become more and more heterogeneous. As an instance of this, the number of platforms of the TOP500 equipped with accelerators has significantly increased during the last years, going from 36 platforms in 2011 to 149 platforms out of 500 in 2020 [@top500]. In the future it is expected that the nodes of such platforms will be even more diverse than today: they will be composed of fast computing nodes, hybrid computing nodes mixing general purpose units with accelerators, I/O nodes, and specialised nodes such as *Field Programmable Gate Arrays* (FPGAs) or *Tensor Processing Units* (TPUs). The interconnect of a huge number of such nodes will also lead to more heterogeneity. Using heterogeneous platforms would lead to better performance through the use of more appropriate resources depending on the computations to perform, but it has a cost in terms of code development and more complex resource management, thus leaving a huge place for optimisation.

A key ingredient of any computing system is the scheduler, which is responsible for handling the tasks submitted by the users and the computing resources. Specifically, the scheduling algorithm has to decide which task to execute first, when to start its execution, and where to allocate it (i.e., which resources to use). Due to the importance of these decisions, the efficiency of the scheduler is crucial for the overall performance of the whole system.
Scheduling is a well understood problem in the context of homogeneous platforms composed of identical resources, since there exist both efficient theoretical algorithms [Dro09] and their practical counterpart implementations in actual batch schedulers, such as SLURM [YJG03]. However, the case of heterogeneous resources, which is crucial in practice due to the evolution of architectures, is not so well understood and it has been the focus of a vast literature in recent years.
The main focus of this manuscript is to study different aspects of scheduling problems in heterogeneous High Performance Computing and Edge Computing platforms.

1.2 Contextualisation of Edge Computing

In parallel to the development of distributed and parallel platforms, we have witnessed the emergence of *in-situ* processing. The in-situ processing paradigm is driven by the conviction that, because of fundamental limitations of network and storage performances growth compared to computation, the data processing and analysis should be performed as close as possible to where and when the data is produced.

With the explosion of the number of connected devices – Cisco predicts 29.3 billion devices in 2023 [@cisco] – and the ever growing amount of data generated, it has become clear that the current *Cloud-centric* vision, where most of the computation is off-loaded to Cloud data-centres, is out of order with a saturation of the network bandwidth and high latency penalties [Zha+15]. Not to mention the energetic, privacy and security concerns of sending every request to be processed in the Cloud. Hence emerged the *Edge (or Fog) Computing* paradigm [Shi+16], whose main objective is to move towards a *local* vision in which on demand computations are performed close to the place where the data are produced to mitigate data exchanges and provide real-time responses. This trend was further accelerated with the proliferation of *Internet of Things* (IoT) applications [AIM10], as well as the advent of new technologies such as *Mobile Edge Computing* [AA16] and *Network Function Virtualisation* [Mij+15] that favoured the deployment of Cloud Computing-like capabilities distributed at the edge of the network, such as in smart buildings and emerging smart cities.

In the meantime, the energy consumption of data-centres has become one of the main concerns in the past years. As data-centres gather thousands of servers, they are huge energy consumers with the biggest requiring more than 100 MW of power capacity. Between 2010 and 2018, the total number of computing requests in data-centre has been multiplied by 6.5 and the total number of IP traffic was multiplied by 11, while the global energy consumption only increased by 6% and stayed within the 1%-1.5% range of worldwide electricity use [Mas+20].
However, even if the power efficiency keeps increasing each year, the energy used only for the cooling of the servers continues to be an important portion of the total energy consumption. In 2017, Ni and Bai [NB17] studied the air conditioning energy performance of 100 data-centres. They reported that the average *Power Usage Effectiveness* (PUE) – the total power consumption of the facility divided by the effective power used by the *Information Technology* (IT) resources – of the data-centres was 2.13, meaning that for 100 W used by the IT resources there was an additional average of 113 W used by the power supply, the cooling system, fans and other equipments of the data-centre. The authors also reported that the cooling system alone accounted for 38% of the total energy consumption of the facility on average, with figures ranging between 61% for the least efficient systems to 21% for the most efficient. This introduces an increasing need for better power and thermal management within the computing infrastructures and an opportunity for companies like *Qarnot Computing* [@qarnot], *Cloud&Heat* [@cloudheat] or *DigiPlex* [@digiplex] to emerge, proposing approaches to reduce IT waste heat by directly using the servers as the heating systems in buildings. In addition, by bringing IT servers within the buildings, this becomes an answer to the ever growing computing needs of the IoT world.

Edge Computing can be viewed as an extension of Cloud Computing, with storage and computing capabilities brought closer to the users. This entails several challenges for an efficient management of, first, the resources, as they are heterogeneous, dynamic, not always accessible and distributed over the edge of the network, and, second, the multiple users accessing these resources which often have different objectives.
It is worth to notice that Edge Computing platforms are usually hierarchical, with connected devices, storage and computing resources at different levels of the hierar-

chy, and that resource management problems can emerge at multiple points of the platforms. We can take the example of the *Qarnot Computing* company [@qarnot], whose hierarchical platform is mainly composed of computing resources embedded in heaters and distributed in smart buildings. This platform is of particular interest since an efficient management of its resources is required at different levels of the platform with: 1) a global load balancing of the data placements and computing requests across the different Edge sites; 2) a local scheduling problem within an Edge site; 3) a thermal management of the heaters hosting the computing resources.

1.3 Infrastructure Simulation

In the High Performance Computing and Cloud Computing domains, the design, test and analysis of new resource management policies is mostly performed by simulation. Of course, the use of experimental test-beds, whenever they exist, can serve this purpose and would provide results of a real execution. However, such a solution is both costly and time consuming, as a one-month experiment on a large platform would consume a huge amount of core-hours and – obviously – take one month to execute in an environment that may not be fully controllable and noisy. In comparison, the same experiment in simulation can be run on a regular laptop for a few minutes, and would rarely last for more than one hour, enabling to quickly test various settings and resource management policies with a deterministic and controlled environment.

Another point in favor of the simulation is that it enables researchers and developers to confront the simulated infrastructure with larger-scale scenarios or emergency situations. For example, one could easily test the behaviour of a job scheduling policy on a platform whose number of machines and job submissions is multiplied by one or more orders of magnitude, or in a scenario where half the total number of machines crashes at some point of the execution. Besides, simulation also enables to test scenarios that are not – yet – feasible on a real platform, such as a new feature or a change in the infrastructure, to study whether it is worth investing time and money in this direction for a production platform.

It is worth to emphasize that simulation is not simply the holy grail of experimental testing. It also has some drawbacks. The simulation of a platform does not provide real but *realistic* results, as it only captures the effects that are modelled by the simulation engine. In this regard, an effort should be made in the implementation and validation of the models that are simulated.

The story goes the same way in the world of IoT and Edge Computing platforms. Similarly to what has been proposed for the Cloud Computing paradigm [Leb+19], we strongly believe that a dedicated simulation toolkit to help researchers investigate resource management strategies at the Edge should be released soon. Besides resource heterogeneity, network specifics (latency, throughput), and workloads, Edge Computing infrastructures differ from Cloud Computing platforms because of the uncertainties: connectivity between resources is intermittent, storage and computation resources are more heterogeneous than in the Cloud and can join or leave the infrastructure at any time for an unpredictable duration. In other words, a part of the infrastructure can be isolated or unavailable for minutes/hours, preventing the access to some data-sets or assigning new computations. Thus, Cloud Computing simulators are not appropriate to study Edge Computing challenges.
One of the biggest challenges that had received a lot of interest in the past years is the *Service Placement Problem* [ADL19], i.e., *where to transfer data-sets according to their sources and schedule computations to optimise metrics such as the Quality of Service and Quality of Experience of the users, or the platform utilisation.* Taking a look at the literature, one can find a myriad of works dealing with the Service Placement Problem in the IoT [Ait+19; BF17; Don+19; Naa+17; Ska+17; Xia+18; You+18]. However, it is difficult to understand how each proposal behaves in a different context and with respect to different objectives, such as scalability or reactivity. In addition to having been designed for specific use-cases, available solutions have been evaluated either using *ad hoc* simulators or through limited in-vivo experiments, limiting the capability to perform fair comparisons between them. We aim in this work at solving this weakness by extending an existing simulation toolkit to the Edge Computing paradigm and proposing methodology examples to help researchers/engineers evaluate state-of-the-art and new resource management policies.

1.4 Plan and Summary of Contributions

Throughout this dissertation, we will first take a focus at a typical Edge Computing platform to address different resources management problems appearing at multiple levels of such infrastructures. We study different parts of the platform involving an efficient management of resources and propose solutions for improvement both theoretically and through simulation, which can benefit other Edge Computing platforms.

In Chapter 2, we first present in details the *Qarnot Computing* platform and its interactions with the users and requests. We then expose some main challenges which Edge Computing owners and managers are confronted with, and provide some examples applied to such platforms. We introduce in Chapter 3 the simulation toolkit Batsim/SimGrid [Dut+16; Cas+14] and the modifications that were made to enable the simulation of Edge Computing infrastructures. Then, a complete simulation of the *Qarnot Computing* platform and a simple experimental campaign to compare scheduling variants are presented. The objective here is to demonstrate a tool allowing researchers and engineers to perform fair and easy evaluations and comparisons between various scheduling and data placement strategies in this kind of platform. Then, we address in Chapter 4 the problem of predicting the temperature of a smart-building environment. More precisely, we are interested in the evolution of temperatures of a room with heaters and propose different solutions for the prediction of the temperature for a heater and its surrounding air.

In the second part of the dissertation, we focus on theoretical scheduling problems in the domain of High Performance Computing. More specifically, we introduce in Chapter 5 different versions of the problem of scheduling a parallel application on a hybrid multi-core platform and quickly survey the best known results in the literature. Then, in Chapter 6, we detail our own contributions regarding this problem. We provide two new scheduling algorithms for different settings of the problem, prove theoretical guarantees on their performances and conduct an experimental campaign to compare in practice their performances with baseline algorithms of the literature.

Finally, we introduce in Chapter 7 the formulation of a new two-agent scheduling problem on parallel machines applied to the context of an Edge site, and conclude this dissertation in Chapter 8.

In addition to the scientific work, scientific *mediation* occupied a great time during the preparation of this dissertation, with the creation of mediation activities in mathematics and computer science, the participation of diverse manifestations such as the French *Fête de la Science*, and the help in the translation of the *CS Unplugged* website [@CSU].

2 Edge Infrastructures: a Case Study

This chapter discusses Edge infrastructures and takes focus on the platform of the *Qarnot Computing* company, as it will be our main case study throughout the first part of this dissertation. The first sections give a complete description of the Qarnot Computing platform, its users and the decisions taken to service the different requests that are made by the users. The last section presents the main challenges emerging from such Edge infrastructures and gives examples related to the Qarnot Computing platform.

2.1 The Qarnot Computing Platform

Incorporated in 2010, the *Qarnot Computing* company (Qarnot in short) aimed at developing *"a disruptive solution able to turn IT waste heat into a viable heating solution for buildings"*. In other words, Qarnot deploys a *Distributed Cloud Computing* platform that re-uses the heat produced from computing as a heating source for smart buildings [NDT16]. The Qarnot platform mainly consists of a collection of digital heaters embedding several processing units attached with a large aluminum heat-sink to diffuse heat locally. Totally silent and based on free cooling, these heaters leverage the heat dissipated by the processors making useful computations to provide the heating service. In 2016, Qarnot started to work on adapting the computing capabilities of the platform to be used both for smart-building processes and Distributed Cloud Computing.

2.1.1 Platform Overview

The Qarnot Computing platform is a hierarchical platform composed of 3 layers and is summarised in Figure 2.1. Is it mostly distributed at the edge of the network, where all the computing resources of the platform are located. The computing resources are disk-less machines – called **QMobos** – embedded in smart heaters

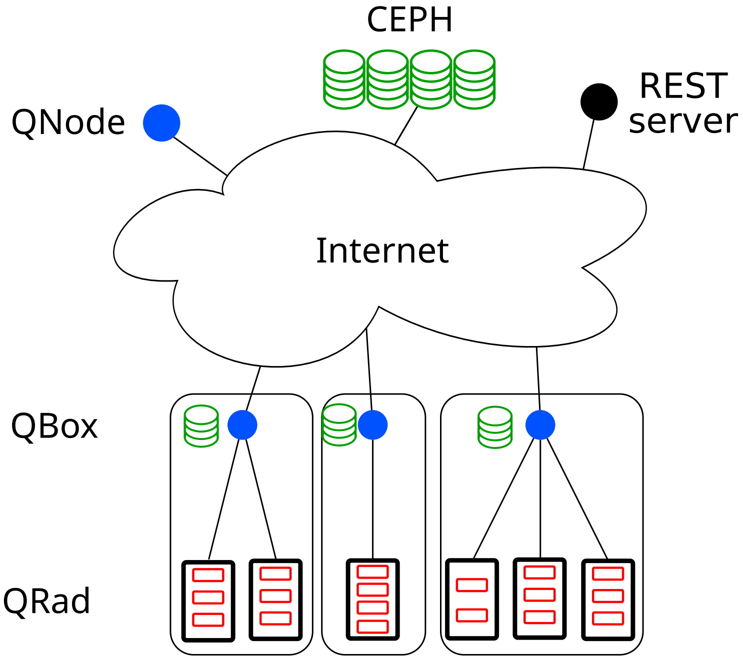


Figure 2.1: Scheme of the Qarnot Computing platform. User interfaces are colored in black, computing resources in red, storage resources in green and decision-making components in blue.

– the **QRads** – deployed in rooms of housing buildings, offices and warehouses across several geographical sites in France. This constitutes the bottom-level of the platform, or the **QRad-level**.

On each of the geographical sites there is a server – called the **QBox** – connected to every QRad of the site and in charge of monitoring and managing them. The QBox also embeds an NFS-based storage – called the **QBox disk** – with a few TBs of capacity that enables the computing resources to manipulate data. This constitutes the middle-level of the platform, or the **QBox-level**.

All QBoxes are connected to the Internet, and to French data-centres used by Qarnot. These data-centres – called the **QNodes**– are hosting the control and monitoring services of the whole platform and the central storage system **CEPH**. There are also servers implementing *RESTful* web services that serves as a gateway for the clients. This constitutes the top-level of the platform, or the **QNode-level**.

Each component of the Qarnot platform can be classified into four different categories: the user interfaces, the computing resources, the storage resources or the decision makers, which are respectively colored in black, red, green and blue in Figure 2.1. The roles of each component, and their interactions, are described in detail in Section 2.1.2.

At the time I write these lines, the whole platform is composed of about about 3,000 disk-less computing machines embedded in 1,000 QRads distributed across 20 Edge sites, and it is growing quickly. On a daily basis, the Qarnot platform processes from a few hundred to several thousands of batch jobs, thousands of cores are provisioned to dedicated corporate customers, and up to tens of GBs of data are replicated from the central storage to the QBox disks.

The platform acts both as a heating service provider for the inhabitants where the heaters are deployed, and as an offloading platform for Cloud Computing. Users can interact with the platform from two entry points, the QRads and the REST servers. We denote by **inhabitants** the users interacting with the QRads, while the ones interacting with the REST servers are **clients** or **Cloud Computing users** of the Qarnot platform.

Two different user requests originate from these interactions: the temperature requests and the computing requests, which are detailed in Section 2.2.

A **temperature request** comes from a QRad when the target temperature of its ambient air has been changed by the inhabitants. Depending on whether the new target is higher or lower than the current air temperature, the request is then translated into a need for heating or cooling of the QRad, inducing a change in the available computational power of the QRad.

A **computing request** comes from a REST server when a client wants to execute a program, denoted as a **job**, on a computing resource of the Qarnot platform. These computing requests are then to be scheduled and executed on the computing resources inside the QRads.

As one can see, these two requests are somewhat intricated: Heating requests mean that some processors inside a QRad are available and waiting for a job to execute, while computing requests mean that some jobs are waiting for available processors to execute them.

2.1.2 Platform Components in Details

REST Servers

The REST servers make the interface between the Qarnot platform and Cloud Computing users. The servers are connected to the QNodes and propose an API for the users to manage authentication and authorisation aspects, invoicing, as well as computing jobs submission and execution.

REST servers are also the interface between the CEPH and Cloud Computing users, and enables them to push data that will be used by the computing jobs, or get the results of jobs that completed successfully.

QNodes

QNodes are the “global” resource managers of the platform. They manage and monitor all the underlying QBoxes and implement scheduling and deployment engines. Due to security and privacy reasons, each private client of the Qarnot platform is associated to a separate QNode, and all QNodes manage and use concurrently the QBoxes and QRads of the platform. For simplicity, we will restrict our vision to a single *logical* QNode that manages all client requests.

The QNode maintains a scheduling queue containing Cloud Computing job requests received from the REST servers. Periodically, every QBox reports to the QNode information about their state and the state of their QRads, such as the number of QMobos that are currently computing or are available for computing. From the information received in the periodic reports, the QNode takes scheduling decisions and dispatches jobs to QBoxes providing them enough work load to satisfy the heating requirements of their QRads.

CEPH

Somewhere in the Cloud is deployed a CEPH cluster[1], which is the main storage server of the Qarnot platform.

Each Cloud Computing user has a given storage space in the CEPH where they can upload all input data related to their computing requests, and get all the output data that were uploaded by the computing requests once they have been serviced. These data are accessed by the clients via the REST servers. Note that it is the role of the clients to manage their own storage space. In particular, Cloud Computing users have to make sure that the required data for a given job have been completely uploaded to the CEPH and that there is enough available space for its output data before submitting this job.

[1] `https://ceph.com`

QBoxes

While a QNode acts as a global resource manager for the whole platform, the QBoxes handle this role closer to the edge of the network, being “local” resource managers of a deployment site (usually a smart building). A QBox is a server that hosts services related to the monitoring of the QRads they are connected to and the scheduling of jobs onto the computing resources embedded in these QRads. Thus, a QBox manages a sub-network of the whole platform.

A QBox frequently gathers information from the QRads, such as their state, their temperature and the ambient temperature of the room, and the number of QMobos currently running jobs or available for computing new ones. From these information, the QBox computes the heating requirements of the QRads to take scheduling decisions and to decide the execution of the jobs dispatched from the QNode.

During the periodic reports, the QBox sends to the QNode the number of QMobos that are available for computing, meaning that some QRads need heating and jobs should be executed on them. The periodic reports also contain information about the current execution of the jobs, which are then made available to the Cloud Computing users through the REST servers.

Figure 2.2 depicts a typical example of a deployment site of the Qarnot platform with a QBox and several QRads attached to it in a smart building.

QBox disk

Each QBox has its own local storage, the QBox disk, which is connected to the CEPH via the Internet. Since the computing resources of the Qarnot platform are disk-less, the QBox disk implements an NFS (Networked File System) enabling the jobs to read input data and write their output.

The QBox disk acts as a local cache, in the sense that all input data and the container holding the program of a job are replicated onto the QBox disk before a job starts. During execution, a job can read this input data and write its output to the disk, which is then uploaded to the CEPH upon job completion.

In practice, the storage capacity of the QBox disk is a few TBs, which enables the disk to keep in storage many job input data. This is to avoid redundant data movements, in the event that other jobs with the same requirements are dispatched to this QBox in the future.

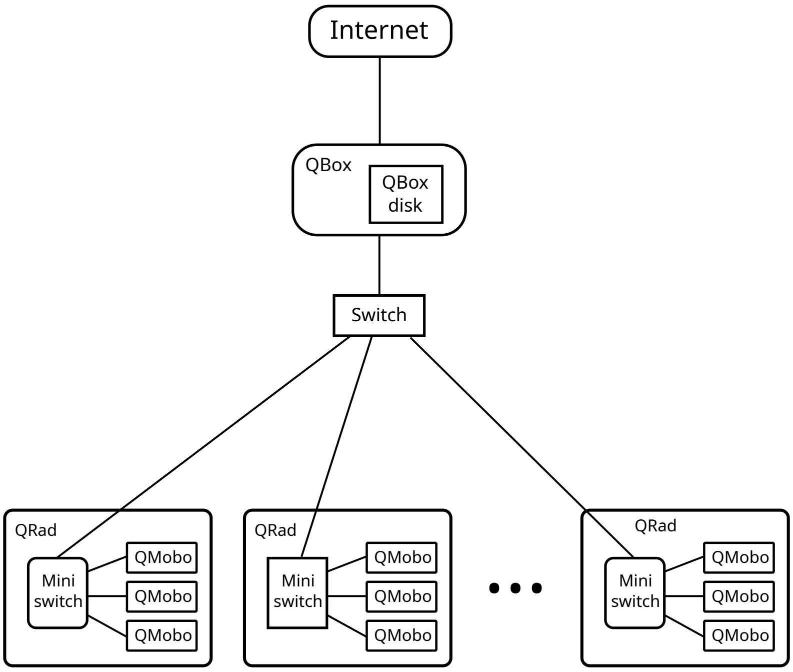


Figure 2.2: Example of a deployment site with one QBox and several QRads in a building. The switches, QRads and QMobos are in the same Local Area Network with 1 Gbps links while the QBox is connected to the Qarnot platform through the Internet.

In case the available space is not sufficient when input data of a job should be copied to the disk, a classic LRU (Least Recently Used) eviction policy is used to remove old data.

QRad

Totally silent and based on free cooling, the QRads are deployed in homes, offices and warehouses where they support two different roles in the Qarnot platform.

On the IT side, a QRad is a small multi-core server embedding between one and four QMobos interconnected by a network switch. All QRads of a deployment site are interconnected between them, and to the QBox, in the same local network through another switch, as depicted in Figure 2.2.

On the physical side, a QRad is an aluminum heat-sink case that allows the embedded processors to cool down, playing the role of the heater. It is also equipped with different sensors, such as temperature of the QRads and its ambient air, CO_2 levels, noise and humidity captors, etc., and makes the interface with the inhabitants enabling them to turn on and off the heating or select the desired target temperature of the ambient air.

The QRad is responsible for its embedded QMobos and heating management. At all times, a *Frequency Regulator* 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 processors executing jobs or completely turning off a QRad when it is too warm.

If the ambient temperature is too low, then the QRad needs more heating. This results in an increase of the computing capacity to satisfy the heating needs. This heating need can be serviced by an increase in the frequency of the processors already executing jobs in the QRad, or by starting the execution of new jobs. If no regular Cloud Computing job is available for computing, the Frequency Regulator will automatically start volunteer computing [Dav04] or crypto-currency mining programs. In addition, when the heat generated from computation is not sufficient to satisfy the heating requirements, auxiliary resistors embedded in the QRad can also be turned on to complement the heating. In total, a QRad can provide up to about 750 W of heat when all QMobos are computing jobs and all auxiliary resistors are turned on.

On the contrary, if the ambient temperature is above the target temperature, the QRad needs to cool down. In that case, the Frequency Regulator turns off all auxiliary resistors and lowers down the frequency of the processors executing jobs in the QRad to reduce the heat produced from computation while not stopping the execution of the jobs. As a last resort, if the QRad is too hot for a long period of time, all executing jobs are killed and the QRad is turned off for cooling. The jobs that have been killed are then returned back to the QNode to be re-dispatched later on.

QMobo

QMobos are the only computing resources of the Qarnot platform. Embedded in QRads, each QMobo is equipped with a multi-core CPU, either Intel Core i7 or AMD Ryzen 7 in the current platform, between 16 and 32 GBs of RAM and a network controller.

By default, every QMobo is turned off and is only waken up when a dispatched job should be executed on that QMobo. Once the job has completed, the QMobo automatically turns off until a new job arrives.

We recall that, due to security reasons since QRads are physically located in people's rooms and offices, QMobos (and QRads) do not embed resilient storage. Thus, upon starting a new job, QMobos are booted over the network with a clean environment, and data reading and writing during the execution of a job are performed directly to the QBox disk through NFS. Moreover, even if a job only uses one core during its execution, the whole QMobo is reserved for that job, to prevent jobs from different users to share a QMobo.

2.2 Users and Requests

There are currently two types of users of the Qarnot platform, and each is associated to a specific type of request, either a *temperature* or a *computing* request. We detail in the following these two types of requests and how they impact the platform. We also introduce another kind of computing request related to the *Internet of Things* world.

2.2.1 Temperature Requests

Temperature requests originate from QRads whose target temperature has changed from actions of the inhabitant. These requests directly impact the computing capacity depending on whether the new target is higher or lower than the current ambient temperature.

The temperature requests are resilient, in the sense that an action of the inhabitant simply changes the value of the target temperature but the heating or cooling needs must be continuously serviced. It is the role of the QRad and its Frequency Regulator to maintain an ambient temperature as close as possible to the target. If the absolute difference between the target and ambient temperatures is smaller than one degree[2] the temperature requirements are considered to be met and the Frequency Regulator does not change the QRad behavior. When the ambient temperature is more than one degree below the target, the temperature is too low and the QRad goes into heating mode. In that case, the Frequency Regulator will try to fulfill this heating need by increasing the frequencies of the QMobos, computing more jobs or turning on auxiliary resistors, as described in Section 2.1.2. Conversely, when the ambient temperature is more than one degree above the target temperature, the QRad goes

[2] Any mention of degree here and in the rest of the chapter will refer to Celsius degree.

into cooling mode. The Frequency Regulator will in that case turn off resistors, lower down the frequency of the QMobos or kill jobs being computed.

It is worth noticing that, even if the target temperature does not change, the heating and cooling needs of a QRad can vary due to the local environment directly impacting the ambient temperature. Among the factors impacting the ambient temperature is the level of thermal isolation of the room, combined with the weather conditions and the temperature outside the room. For example, passing from a bright sunny day to a stormy cold evening in spring may quickly decrease the ambient temperature near the QRads. The number of people inside the room also has a great impact on the ambient temperature, as a single human body at rest produces between 80 and 120 W of heat. Thus, the computing availabilities of the platform are not only impacted by the direct action of the inhabitants on the target temperatures, but also by the local environment of the QRads.

2.2.2 Cloud Computing Requests

Cloud Computing requests, or computing requests in short, are made through the REST servers by clients needing computing resources to run their programs.

A computing request submitted to the Qarnot platform is denoted by a **QTask**. To execute a program on the platform, a Cloud Computing user needs to upload in the CEPH all the input data of their program as well as a Docker image containing the program to run. Then, the user creates the QTask via the REST API by providing the Docker image, the input data to use and the command to execute. Note that a QTask can also use an image already present in the Docker Hub. A QTask can be decomposed as a bag of several jobs, denoted by **instances** in the Qarnot terminology. All instances of a QTask share the same Docker image and input data but each instance has a different command to execute. This is of particular interest for example when one wants to process each frame of a given movie by providing each instance a single frame or a range of frames to process.

Upon creation of a QTask by a user, the REST server submits all instances of the QTask to the QNode scheduler, which regularly dispatches the instances onto QBoxes where QMobos are available for computing. When new instances are dispatched to a QBox, their data dependencies are retrieved from the CEPH and their execution starts as soon as the data arrived on the QBox disk.

This kind of request represents the majority of the work load of the Qarnot platform, which can be seen as “batch” jobs as each job executes on a single computing unit and is independent from the others. Another type of computing request is the **cluster** job. A cluster job corresponds to a request of several computing resources that can communicate between each other. Such a request is made for users wanting to execute parallel programs or to perform service provisioning. For example, one could run on a cluster job a personal server that takes requests from a website through the internet and then deploys on one of the computing resources allocated to the cluster a program to service the request.

Contrarily to regular jobs where instances can be dispatched on different QBoxes, all computing resources of a cluster job must be within the same local network for the resources to communicate with each other. This imposes that a cluster job must be fully dispatched to a single QBox.

Servicing the Computing requests involves all three levels of the Qarnot platform. A first scheduling round is periodically performed by the QNode to dispatch as much QTask instances and cluster jobs as possible to QBoxes where computing resources are available. Then, the QBox schedules each received instance or cluster job on the available QMobos, depending on the heating needs of the QRads. Finally, the QMobos inside the QRads actually execute the computing requests.

2.2.3 IoT Requests

Since 2016, Qarnot Computing is working to integrate into the platform the execution of programs coming from the smart-building environment. Such programs can be for example the periodic monitoring of microphones for alarm sound detection [DNC17], a federated learning program that executes on multiple resources distributed in the building or sporadic programs of the type *sense-process-actuate* that are more regular in the IoT world.

This new kind of computing requests, that we denote by **IoT requests**, will be handled by a similar mechanism as for the Cloud Computing requests in the exception that they will be directly addressed to a particular QBox and must be executed on computing resources of that QBox.

The constraint of staying within a QBox sub-network follows an argument of **locality** and **security**: the data that will be used by these requests will most likely be generated close to the Edge site where the QBox is deployed (within a smart building for example) and stored in the QBox disk. This way, an IoT request and its input data do not have to transit through the network to the QNode and another QBox, and the request can be serviced with smaller delays.

2.3 Challenges

As any platform deployed at the edge of the network, the Qarnot platform has to deal with numerous challenges that make it both interesting to study and difficult to manage. We present in this section some challenges common to Edge Computing platforms and how they apply to the Qarnot platform.

2.3.1 Data Management and Communications

The main motivation of the Edge Computing paradigm is to reduce data movements and consume the data as close as possible to where they are produced, instead of relying blindly on the Cloud Computing capabilities. Going a step further, we could state that computing requests should also be serviced as close as possible to where the requests are made to improve the quality of service (e.g., reduce the response time). This paradigm also bears arguments of security and privacy since the data and computations would stay in the vicinity. For example, IoT requests originating from an inhabitant of a smart building could be processed on the computing resources located in their own apartment.

In this context of keeping the data and computation close to where they originate, an effort must be made to efficiently exploit the resources, either for computing or for storage. A simple policy that always selects the closest resource to store the data or service a request would be unrealistic since the resources at the edge of the network are usually limited. Thus, there is a trade-off to be made between using the closest resources with a risk of overloading, and using more farther, theoretically unlimited, resources at the cost of more data movements and latencies in the requests.

For the Qarnot Computing platform, this problem is related to the dispatching of instances to the QBoxes. Whenever an instance should be started on a QMobo, all its input data must be fetched from the CEPH, except for the data already in the local cache of the QBox disk. It may appear during the dispatch of instances that the QBoxes already having the input data in their disk have no available computing resources, while other QBoxes having available resources do not have the input data in their disk and data transfers would be required. In addition, since the network links are not the same for all Edge sites of the platform, the time to transfer the data to different QBoxes will vary, impacting the expected starting time of the instances.

To overcome such problems, a good solution is to have a prediction of the time it would take to transfer the data to different Edge sites and take scheduling decisions based on these predictions. Such a prediction mechanism would nonetheless be difficult to implement as a general knowledge on the state of the network is required, which is known to be uncertain and variable.

Another solution to reduce the response time of computing requests is to perform data replication. Whenever there are new data which may be used by multiple requests, these data are automatically replicated on several storage spaces. This way, a computing request depending on these data would have more computing resource candidates and its response time would be improved. Again, this solution is a matter of trade-off as it is in contradiction with the primary objective of reducing the data movement but with the goal of improving the quality of service for the users. As it will be witnessed in Chapter 3 for the case of the Qarnot platform, replicating data increases drastically the number and volume of data transfers for a performance gain that may not be satisfactory enough.

2.3.2 Resources Availability

The problem of resources availability is a common challenge of all Edge infrastructures. In many cases, the difficulty resides in having a strong connection between the user and the platform. As the users are usually *mobile* (e.g., someone using web services from a smartphone) or remote with limited power (e.g., sensors and actuators), the connection with the Edge platform is unreliable and unpredictable. Unfortunately, this is the price to pay for being remotely connected.

In some platforms, the computing resources may be unavailable at times. This is the case for volunteer computing which, adapted to an Edge scenario, consists in leveraging the idle computing power of registered smartphones and computers in a network (e.g., a smart building) to perform useful computation originating from the IoT instead of offloading these computation to more traditional servers in the Cloud. The Qarnot platform also falls in this category since the availability of the computing resources is directly linked to the heating demand of the QRads.

These computing resource availabilities are uncertain and very difficult to predict as they depend on factors that are external to the platform, such as the users' utilisation of their smartphones and computers in the first case, and the weather and actions of the inhabitants in the case of Qarnot.

One solution to overcome such uncertainties is to employ machine learning techniques to uncover availability patterns for the computing resources and habits of the users (inhabitant and smartphone owner). Unfortunately, in addition to the security and privacy aspects of learning users' habits, these users are human, which means that they can be unpredictable and the learning of usage patterns can be completely inefficient. A mean to strengthen this solution is to attribute a score to each computing resource indicating *how stable and predictable are the resource availabilities?* For example, the QRad of an inhabitant changing randomly the target temperature will have a very bad score for its embedded computing resources, while a QRad whose target temperature changes each day at 8:00 am and 6:00 pm will have a very good score.

In addition to the uncertainty in the availability of the computing resources, platforms have to deal with their dynamic heterogeneity. Keeping our examples with smartphones and QRads, these resources are not simply available for computing or not. Depending on external factors such as the battery discharge and the processor temperature of the smartphone, or the current need for heating of the QRad, the frequency of the processors can be limited to reduce the power usage and preserve battery life or limit the heat generation. Thus, the computing power of these resources is heterogeneous over time. Unfortunately, we do not have time to study and deal with these challenges in detail.

In other platforms, even if the resources are available, they may not be reachable. This is the case for instance with users of the platform that are *mobile* (e.g., someone using their smartphone) or *remote* with limited power (e.g., sensors and actuators). Such users may experience unreliable connection to the network, or connection with highly variable bandwidth or latency, rending the use of the resources difficult or even impossible. This could entail unexpected consequences on the behaviour of an IoT application and its impact on the physical world if the connection between servers and the sensors and actuators is not stable.

It is also worth mentioning that resources availability can also be limited by design, due to their environment. In the case of the Qarnot platform, since the only computing resources are embedded in *heaters*, these resources are only available when heating is required, which is greatly dependent of the weather and the period of the year. As I write these lines, it is close to 28 degrees in my office and I would not be happy if the heater on the wall suddenly started to warm even more the room to compute jobs for users I do not know. Thus, the design of the platform tends to offer more computing power during winter and close to null in summer. Fortunately in this case, this limitation can be overcome by embedding processors in other systems where the demand is uninterrupted, such as *digital boilers* that would use the heat generated from computation to warm up water for domestic use. This is the main solution of the *Cloud&Heat* company [@cloudheat] and is currently in development in the Qarnot company.

2.3.3 Users and Objectives

In classical Edge Computing platforms, multiple users compete for the resources and services of the platform, with different objectives to optimise. For example, one type of user could be interested in the minimisation of the response time of the servers when submitting requests, while another type of user would prefer to minimise the overall completion time of all its requests. This matter is related to the problem of scheduling with multiple agents and is further discussed in Chapter 7.

Taking a look at the Qarnot platform, the first type of user would be a client asking to deploy cluster jobs or to execute programs emerging from the IoT. The second type would correspond for example to a user submitting thousands of small jobs to perform image processing for an animation movie. In that case, the fast completion of a few jobs does not matter much if some others are delayed for a long time, since the only concern is to have the whole set of jobs processed as soon as possible.

Besides, there may be other types of users of an Edge platform who do not make computing requests but can impact the behaviour of the platform and the availability of the resources. In the example of the Qarnot platform these are the inhabitants of the smart buildings. Such users have a direct impact on the computing availability of the QRads as they can turn them off or change the target temperature, and their objective may be opposed to computing users' objectives when the weather is too warm.

In addition to the users' objectives, the personal objectives of the owner or the platform manager has to be taken into account. Usually, the primary objectives are *"make profit"* or *"maximise platform utilisation"* while more secondary objectives can be *"make clients happy"* and *"reduce platform energy consumption"*. Sometimes these objectives are contradictory with each other, for example maximising the platform utilisation while minimising its consumption, and a compromise has to be made. Conversely, some objectives blend well together and with the clients' objectives. This is the case for example when the client's objectives are favoured, making clients happy with more requests serviced and thus more profit made.

This applies without surprise to the Qarnot company: On the one hand, Qarnot earns money from successfully executing jobs of the Cloud Computing and IoT clients, which tend to favour the clients' objectives as well. On the second hand, the platform provides free heating for the inhabitants, meaning that the Qarnot company pays the electricity bill of the QBoxes and QRads deployed at the Edge sites, which brings the need to minimise the energy consumption of the platform. In addition, recall that the computing availabilities of the resources are limited and driven by the heating demand, and that, on a thermal point of view, a useful or a useless computation makes no difference in the quantity of heat dissipated by the processors. Thus, it is preferable for the Qarnot company to favor the execution of computing jobs before performing non-profit heating such as volunteer computing or with auxiliary resistors.

On an energy point of view, one could have difficulties in apprehending how the energy consumption could be minimised, as the amount of energy to reach a target temperature starting from a given temperature is theoretically constant. However, the combination of an overheating and cooling phases to increase the temperature consumes more energy than a constant heating with the proper amount of power consumption. In this regard, having a temperature model being able to predict what will be the temperature after a given time considering an initial temperature and a constant power consumption would be of great interest for this objective of energy consumption minimisation. This matter is further discussed in Chapter 4.

3 A Simulation Tool for Edge Computing Infrastructures

In this chapter, we present in the first part several extensions implemented on top of the Batsim/SimGrid toolkit [Dut+16; Cas+14] to favour fair evaluations and comparisons between various scheduling and data placement strategies for Edge Computing infrastructures. In particular, we developed an external module to allow injecting in the simulation any type of unforeseen event that could occur (e.g., a machine became unavailable at time t). We also implemented a Storage Controller to supervise all transfers of data-sets within the simulated platform.

We chose to rely on Batsim/SimGrid instead of any other available Edge simulators [SOE17; Qay+18] for the following reasons:

- Batsim has been especially designed to test and compare resource management policies in distributed infrastructures. In other words, the design of Batsim enforces researchers to use the same abstractions and, thus, favours straightforward comparisons of different strategies, even if they have been implemented by different research groups;

- Batsim promotes separation of concerns and enables the decoupling between the core simulator and the decision process implementing the scheduler algorithm. Moreover, Batsim uses a text-based protocol to communicate with the decision process that makes the development of a scheduling strategy accessible for a large number of researchers using any programming language of their choice;

- The accuracy of the internal models (computation and network) of SimGrid has been already validated [Deg+17; Vel+13] and extensively used [@SGpublis];

- SimGrid provides a plug-in mechanism, which is of particular interest to deal with the diversity of Edge devices: it lets researchers add new models of specific Edge facilities without requiring intrusive modifications into the simulation engine.

- Most simulator projects seem abandoned or vaguely maintained, while Batsim and SimGrid are active with regular feature additions and releases.

By extending Batsim to the Edge Computing paradigm, we aim at proposing a tool that will enable researchers/engineers to re-evaluate major state-of-the-art load balancing strategies. In particular, we think about scheduling strategies that have been proposed in desktop computing platforms, volunteer computing and computational grids [All+02; Dav04] as these infrastructures have several characteristics in common with Edge platforms.

In the second part of the chapter, we demonstrate the utility of these new extensions with a complete simulation of the *Qarnot Computing* platform, which enabled us to test and study the behaviour of the platform with different job and data placement strategies. The Qarnot platform is a good use-case to simulate an Edge infrastructure as it is mainly composed of computing units distributed across several sites with a mix of local and global computing jobs with data-set dependencies.

This chapter presents a work done in collaboration with Adwait BAUSKAR, Anderson Andrei DA SILVA, Adrien LEBRE, Pierre NEYRON, Yanik NGOKO, Yoann RICORDEL, Denis TRYSTRAM and Alexandre VAN KEMPEN. The work was published in the SBAC-PAD 2020 conference and in an extended Research Report [Bau+20a; Bau+20b].

3.1 Related Work

We present in this section the main competitors and argument for our simulation tool.

There are in the literature many simulation tools for Edge platforms, but most of them focus on the IoT applications and the simulation of connected devices such as sensors and actuators. Among the proposed solutions, we find open-source projects such as MobIoTSim [Pfl+16], DPWSim [Han+14] and FogTorch [BF17], but we also find commercially available simulators like SimpleIoTSimulator [@simpleIoTsim] or even papers presenting solutions that could not be found on the Internet, like IoTSim [Zen+17] or SimIoT [Sot+14].

In our work, we are more interested in the simulation of Edge Computing platforms that will handle the requests issued from the connected devices. It is still worth mentioning that, as it will be explained in Section 3.2.3, the simulation of any type of model and device from the IoT world is made possible with our solution thanks to the plug-ins mechanism of SimGrid.

In the past five years, Edge Computing simulation frameworks close to our work have emerged, such as iFogSim [Gup+16] and EdgeCloudSim [SOE17]. However, these solutions have been built on top of the reference Cloud Computing platforms simulator CloudSim [Cal+11]. Although widely used to study algorithms and applications for the Cloud, CloudSim is based on a top-down approach of Cloud environments. This approach is efficient to deliver the right abstractions to the end-users but unfortunately lacks of validation for the underlying low-level models, as opposed to our solution relying on the validated models of SimGrid.

Another weakness to highlight for most of the existing simulation solutions is that they fail at proposing a modular approach for the resource management policies of the platform. In addition to the above mentioned simulators, we can cite FogNetSim++ [Qay+18], a tool to simulate large Edge networks based on OMNet++ [@omnet]. With these simulators, researchers have to implement a lot of business logic that is redundant each time they want to investigate a new scheduling policy, and they are bound to use the same programming language as the simulator. On the contrary, Batsim delivers all this logic in a language-independent and generic manner, making it more versatile and user-friendly for researchers/engineers.

3.2 Batsim and SimGrid in a Nutshell

Our proposed Edge infrastructure simulator relies on extensions developed in the Batsim/SimGrid toolkit. Released in 2016, Batsim [Dut+16] delivers a high-level API on top of SimGrid [Cas+14] to ease the development and simulation of resource management algorithms.

We briefly introduce in this section the different components, namely SimGrid, Batsim and the decision process connected to it, as well as their role and interactions during the simulation, as schematised in Figure 3.1. For more details on SimGrid and Batsim mechanisms, we invite the reader to refer to Heinrich's [Hei19, Chapter 4] and Poquet's manuscripts [Poq17, Chapter 4]. Besides, we present the plug-in mechanism of SimGrid that researchers can leverage to provide models of particular Edge devices for the simulation.

3.2.1 SimGrid

SimGrid [Cas+14] is a state-of-the-art simulation toolkit that enables the simulation of distributed systems and applications. SimGrid's relevance in terms of performance

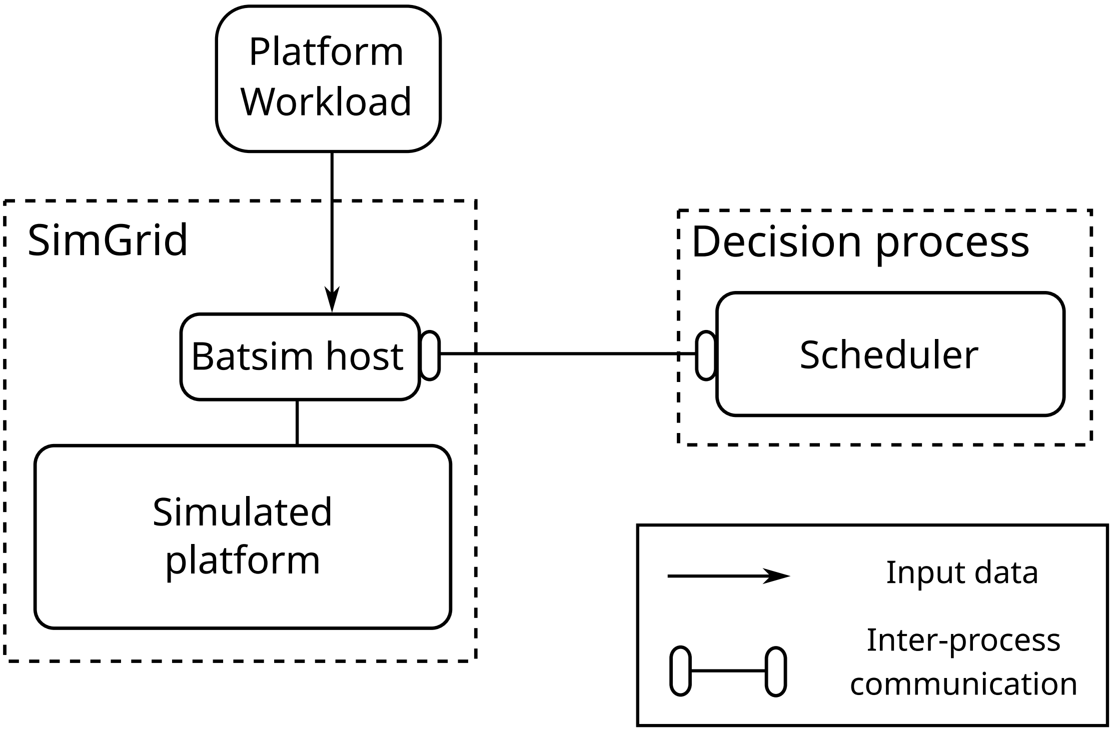


Figure 3.1: Scheme of a simulation with Batsim and SimGrid.

and validity has been backed-up by many publications [@SGpublis]. In addition to providing the program to be evaluated, performing simulations with SimGrid requires writing a platform specification and interfacing the program to simulate. SimGrid enables the description of complex platforms, such as hierarchical infrastructures composed of many interconnected devices with possibly highly heterogeneous profiles, such as the Edge ones.

SimGrid is an open source project whose public repository is available on GitHub [@SG-git].

3.2.2 Batsim and the Decision Process

Batsim [Dut+16] is an infrastructure simulator built on top of SimGrid, which proposes a specialised API to help researchers design and analyse jobs and I/O scheduling systems. Such systems are for instance Batch Schedulers (*a.k.a. Resources and Jobs Management Systems*), in charge of managing resources in large-scale computing centres. Batsim allows researchers to simulate the behaviour of a computational platform in which workloads are executed according to the rules of a **decision process** communicating with Batsim.

The decision process connected to Batsim embeds the actual scheduling code to be evaluated. In other words, to simulate a given scheduling algorithm, an experimenter has to implement this decision process. Internally, the decision process: (i) reacts to the simulation events received from Batsim; (ii) takes decisions according to the given scheduling algorithm; (iii) drives the simulated platform by sending back its decisions to Batsim.

Batsim and the decision process communicate via a language-agnostic text-based synchronous protocol. The communication uses a simple event-based interface: as soon as an event occurs, Batsim stops the simulation and reports what has happened to the decision process. The decision process will in turn answer Batsim with the decisions it has taken and that must be applied in simulation.

Events reported by Batsim can be for example the new submission of a job, or the completion of a job that was running on the platform. To such events, the decision process can for example ask to start the execution of the submitted job on a certain machine, or to wait a given period of time before starting another round of scheduling.

Batsim is an open source project whose public repository is available on GitHub [@bat-git]. Since its first release, a rich ecosystem has developed around Batsim, with several projects implementing the Batsim communication protocol and exposing an API to implement scheduling algorithms in different languages, including C++ [@batsched], Python [@pybatsim] and Rust [@batrust], as well as projects to ease experimentations with Batsim [@robin] and visualise results [@evalys].

3.2.3 SimGrid Plug-ins

When designing an Edge simulator, it is a nonsense to foresee all the models and devices that may compose the platform. There are just too many. However, we claim that leveraging generic models is not the right solution either and so a trade-off should be found. We thus propose to leverage the SimGrid plug-ins capability that facilitates the implementation of new models without requiring intrusive changes in the simulation engine. An existing plug-in of particular interest for us is the *host energy* plug-in [Hei+17] that enables the computation of energy consumption of every host (a computing resource) of the platform, which will be used to compute temperatures during the simulation of the Qarnot platform. There are also plug-ins computing the load or enabling DVFS on computing resources, as well as computing the energy consumption of the network links.

Unfortunately, we underline that this plug-in mechanism is part of SimGrid, and there is no generic manner of exposing information captured by the plug-ins directly to the scheduler through Batsim. Hence, some modifications might be required to extend the communication protocol of Batsim and exchange information between a particular plug-in and the decision process. Examples of such modifications are explained for the case of the Qarnot platform in Section 3.5. This is the trade-off to be able to cope with the high heterogeneity of Edge infrastructures while targeting accuracy of sub-models.

3.3 New Extensions

To ease the study of job scheduling and data placement strategies for Edge platforms, we have been working on a couple of extensions for Batsim. We present in this section the extensions already available, namely the **external events injector** and the **storage controller**. Modifications made to Batsim and its Python API for this work are integrated in the main branch of their repositories [@batgit; @pybatsim].

3.3.1 External Events Injector

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

An external event is represented as a **JSON** (JavaScript Object Notation) object composed of two mandatory fields: a *timestamp* that indicates when the event occurs, and the *type* of the event. Depending on the type of event, other fields can complement the event description, such as for instance the name of the unavailable resource, the new value of an environment parameter such as the network bandwidth, or anything of interest to the decision process.

Similarly to the workload submissions, external events are replayed thanks to the injector process of Batsim. For each external event file given as input to Batsim, with one aforementioned JSON object per line in the file, an **external events submitter** is created during the initialisation of Batsim. Each submitter parses the list of external events from the input file and iterates over the list to submit the external events to the main process of Batsim at the right simulation times. Then, the external event is processed by Batsim, the state of the platform is updated and the occurring external event is forwarded to the decision process.

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

3.3.2 Storage Controller

The **Storage Controller** is a module included in Batsim's Python API to ease the management of storage entities and data-sets, and supervise data transfers during the simulation.

At the beginning of the simulation, the Storage Controller retrieves the list of *storage resources* of the platform and initialises one storage object per resource. These created storages are empty by default, but they can be filled on demand by the decision process by providing a single or a list of data-sets to be added to a storage. A data-set is represented by two fields, *id* and *size*, denoting the unique identifier of the data-set and its size in bytes.

The Storage Controller exposes to the decision process an API to add data-sets to storages during the initialisation of the simulation. It also exposes functions to ask, for example, for the copy of a data-set from one storage to another, or to retrieve the list of all storages holding a copy of a given data-set during the simulation.

When a data-set should be copied from one storage to another, the Storage Controller creates a specific Batsim job for data transfers describing that a given amount of bytes should be transferred from the source to the destination storage resource. Once Batsim notifies that this job has completed, the Storage Controller notifies back the decision process that the requested data transfer has completed.

A timestamp is saved for each data movement. In other words, there is a timestamp associated to each data-set in each storage. This timestamp corresponds to the last time the data-set has been requested on this storage.

When adding a new data-set to a storage, the Storage Controller makes sure that there is enough available space in the destination storage before starting the data transfer. In the case there is not enough space, an eviction policy is used to determine which data-sets should be removed to free space for the new data-set. The default policy in use is LRU (*Least Recently Used*), which removes the data-set with the smallest timestamp in the storage. However, this eviction policy can be easily overridden by end-users of our simulator without diving into the main code. When implementing their decision process, end-users should simply create a call that inherits from the Storage Controller and override the eviction method. This enables the evaluation of more advanced eviction policies that can impact the overall scheduling decisions without requiring direct modifications in the code of the Storage Controller.

Finally, the presence of a particular data-set on a storage can be enforced through the Storage Controller API by assigning meta-information on a data-set. This information can then be used by the eviction policy to prevent for instance the deletion of the data-set while being used by running jobs.

3.4 Future Extensions

In addition to the external events injector, we envision to design and integrate in Batsim an automatic and probabilistic injector of machine and network failures based on statistical studies of the platform logs and learning techniques. Being able to model the dynamic of Edge infrastructures would be also an important added-value for our simulation framework to capture side effects of such events on scheduling strategies.

Regarding the Storage Controller, there are a few improvements in our plans. We are working on extensions to monitor additional information regarding the data on the platform, such as the number of on-going data transfers. We are also discussing with the SimGrid team to see how we can leverage information about the current load of the links between storage resources to have estimations of bandwidth and latency or the time to transfer a particular data-set from a source to a destination.

Finally, one weakness of the Storage Controller is that it is implemented in Batsim's Python API and can only be used by schedulers using this API. We are currently working on the integration of the Storage Controller directly in the main code of Batsim, which would make its features accessible to decision processes written in any language, and not only in Python.

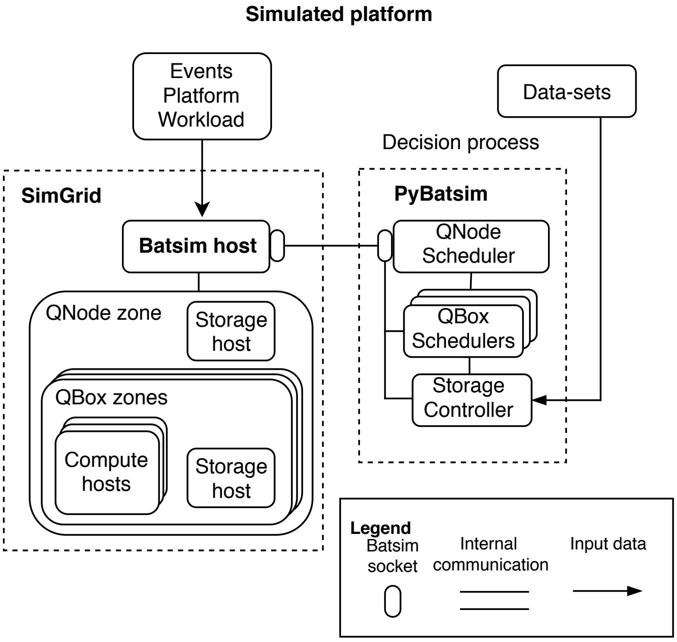


Figure 3.2: Scheme of the simulated *Qarnot Computing* infrastructure.

3.5 Simulating the Qarnot Platform

We detail in this section how we modelled and instantiated the *Qarnot Computing* infrastructure with our simulation toolkit, summarised in Figure 3.2.

We provide details about the description of the platform, the workload, the data-sets and the external events, as well as the temperature modelling and how the input files of a simulation can be generated from logs of the real platform. We also give information about the scheduling algorithms that are implemented in the decision process.

The modifications of Batsim that are specific to the simulation of the Qarnot platform, such as the simulation and management of the temperature, are available in a separate branch of the repository [@battemp]. The temperature plug-in is also available in a separate personal SimGrid repository [@SGtemp].

Unfortunately, even if I strongly advocate open source and reproducible research, we cannot provide access to the source code of the input files generator from the Qarnot logs and the generated files used for the experiments of this chapter due to the company policy and users' privacy reasons.

3.5.1 Platform Description

All details about the platform description are modelled by the XML platform file given to SimGrid at the beginning of the simulation.

Each QMobo is simulated as a single-core SimGrid **host** (representing a machine) as they are the only computing units of the platform. A host holds information such as its unique identifier, the list of speeds and corresponding power usage of its processor, and additional values such as the thermal coefficients required for the temperature modelling, the QRad and QBox ids to which the QMobo belongs to and the name of the city where it is deployed. We keep the same hierarchical structure of the Qarnot platform: QMobos belonging to the same QRad are aggregated in the same SimGrid **zone** (representing a network) where we also added a **router** (acting as a switch). Similarly, all QRads of a same QBox are aggregated within the same zone with a router connected to all QRad routers, as well as all QBoxes of the QNode.

The simulation of storage spaces is done by adding to the platform description special hosts, which carry the Batsim *storage* role. Thus, in each QBox zone, there is one additional storage host for the QBox disk. Similarly, there is one storage host and a router in a separate zone within the QNode zone to represent the central storage server. The router of this zone was connected to each QBox router to simulate the internet connection between the QBoxes and the storage server.

REST servers are not part of the platform description since they only are the interface between the platform and the computing users. The simulation of job submissions is directly handled by Batsim, as described in the next section.

Finally, for Batsim's simulation mechanisms to work properly, a **master host** was added in a separate zone of the QNode zone and connected to each QBox routers.

3.5.2 Workload Description

The workload to simulate is represented by a JSON file fed to Batsim that contains a list of job and profile descriptions.

Job descriptions are defined by the user requests and contain: the id of the job, its submission time, and the job profile to use. A profile description represents how a job should be simulated, plus other information specific to our use-case, and contains: the type of the job to simulate, the number of flops to compute, the job priority and the list of data-sets required as inputs.

Each instance of a given QTask in Qarnot can run independently from the others, so we transcribed each instance as one Batsim job and profile. Instances belonging to the same QTask have the same data-set dependencies and submission times.

3.5.3 Data-sets Description

The list of data-sets is described as a list of JSON objects (one per line). Each data-set is represented by the unique identifier of the data-set and its size in bytes. This file is read by the *decision process* and fed to the Storage Controller at the beginning of the simulation to initialise the state of the storage server.

3.5.4 External Events Description

The list of external events to replay is described as a list of JSON objects (one per line). We designed four types of external events that can occur during a simulation of the Qarnot platform.

A **qrad_set_target_temperature** event means that the target temperature of a QRad was modified by the inhabitant. In addition to the type and timestamp fields that appear in every external event, the JSON object describing this event contains two other fields: *qrad*, the identifier of the QRad whose target temperature changed, and *new_temperature*, the new target temperature in Celsius.

A **site_set_outside_temperature** event means that the temperature of the *outside world* of an Edge site has changed. The additional fields for this event are *site*, the name of the city whose outside temperature changed, and *new_temperature*, the new outside temperature in Celsius.

These two types of external events are related to the temperature simulation and their handling will be detailed in the next section.

The two other types of external events are **machine_available** and **machine_unavailable** and they both contain an additional field *resources* which lists the computing hosts of SimGrid (representing the QMobos) impacted by the events. These events simulate the fact that some machines have become (un)available at the time the event occurs, and they are handled by the decision process.

When some QMobos become unavailable, this means that the QRad containing those QMobos was turned off or that the QBox in charge of the QMobos is no longer reachable by the QNode (due to network partitioning for example). In such a case, the QBox scheduler in charge of these QMobos kills every job that was executing on these QMobos. The scheduler also marks these QMobos as unavailable and will not try to schedule jobs on them until a *machines_available* external event for these QMobos is received.

3.5.5 Temperature Modelling

As temperature plays an important role in the platform dynamicity and the scheduling decisions, we leveraged the plug-in mechanism of SimGrid to implement our own temperature model. Built on top of the existing energy plug-in [Hei+17], our plug-in computes the temperature of a QRad and its ambient air after a given time period using a novel prediction method that will be detailed in Chapter 4.

As stated previously, the target and outside temperature changes are simulated through external events. When a change in the target temperature of a QRad occurs, the information of the external event is forwarded to the QBox scheduler in charge of the QRad and its target temperature is updated. We also modified Batsim to relay an event related to the outside temperature directly to the SimGrid temperature plug-in, and we modified the communication protocol to periodically forward the ambient air temperature of each QRad to the QBox schedulers in the decision process.

Recall that one QRad of the Qarnot platform may embed several QMobos, and that a QMobo is simulated as a SimGrid host. Thus, in the description of the SimGrid platform, we attached a particular property *temperature_role* to each host. For this property, one host of a QRad was attributed the value *master* and the others the value *slave* to have only one host responsible of computing the temperatures.
During the simulation, when a change in the power consumption of a host is detected, the *update* function of the plug-in for that host is called. If this host is a *master*[1] then it will retrieve the energy consumed by each host of the QRad since the last update and compute the new temperature of the QRad and the ambient air. If the host is a *slave* then it will call the *update* of its corresponding master host.
Notice that it is not necessary to periodically update the temperature but only at the end of a (potentially long) time period during which the power consumption was constant.

In addition to the *update* function, the temperature plug-in also exposes functions to get the temperature of the QRad and of the ambient air, to set the current

[1] Master in the sense of the temperature plug-in, not to be confused with the general master_host of Batsim.

temperature of the outside and to get the total energy consumed by all hosts of a QRad.

3.5.6 Input Files Generation

A log extractor was built to generate all the input files from real logs of the Qarnot platform, for a given time period. This includes the XML file for the platform description, the JSON file describing the workload to simulate, and the two files containing the list of data-sets present in the central storage server and the list of external events to replay during the simulation.

For our experiments, the external events describing the outside temperature changes of an Edge site were generated on a one-hour basis, retrieving temperature records from the Meteo60 website [M60]. Moreover, since we want to simulate the Qarnot platform during an exact time period, we added in the external events input file a special event that enforces the simulation to stop at a particular time.

3.5.7 Decision Process Implementation

To leverage the Storage Controller component presented in Section 3.3.2, we implemented the decision process that will be connected to Batsim using its Python API [@pybatsim]. As explained in Section 2.1, scheduling decisions in the Qarnot platform are taken at two different levels of the hierarchy: at the QNode and at the QBox levels. Thus, we implemented the scheduling algorithms for both levels as two different Python objects. At the initialisation of a simulation, the decision process creates one instance of the QNode scheduler, one instance of the Storage Controller provided by Batsim's Python API and one instance of the QBox scheduler for each QBox of the simulated platform, as depicted in Figure 3.2.

During a simulation and upon receiving a message from Batsim, each simulation event is forwarded to the Storage Controller or the scheduler that should handle it. For example, an event notifying the submission of a new job in the system will be forwarded to the QNode scheduler that will dispatch the job in its next scheduling round, while an event notifying the completion of a job will be forwarded to the QBox scheduler in charge of the QMobo that was executing this job. Moreover, even if no simulation event occurred, the decision process asks Batsim to periodically *wake* him *up* to initiate the periodic reports mechanism between the QBoxes and the QNode, and to perform the frequency regulation of the QMobos.

3.6 Investigating Placement Strategies

We test in this section various job and data placement strategies in simulation to have a better understanding of how the *Qarnot Computing* platform can adapt with different scenarios. We expect the results to drive future design of policies for the real platform.

Two kinds of experiments have been performed to investigate the Qarnot use case. The first aimed to compare the standard scheduling policy used in the real Qarnot platform with a policy based on locality of the data-sets. The second experiment enabled us to study the impact of replication policies for the data-sets that are uploaded on the platform (i.e., how they affect the scheduling decisions). The code of all the evaluated schedulers is available in a dedicated branch of Batsim's Python API repository [@pytemp] in the *schedulers/greco* folder.

3.6.1 Data/Job Scheduling Policies

Along with the real **Standard** Qarnot scheduler that serves as a baseline for our experiments (see Section 3.5), we implemented a variant of the QNode scheduler using the data locality to take scheduling decisions, denoted by **LocalityBased**. Upon dispatching instances, LocalityBased gives priority to the QBoxes already having the data-set dependencies of the instances on their storage disk. This variant aims at taking benefit from the data locality and reducing the data transfers.

To evaluate the impact of data placement on the scheduling decisions, we also implemented three variants of replication policies upon the submission of a QTask[2]. The question we want to answer with these variants is whether replicating data-sets can achieve significant improvements for the Quality of Service, and at which cost? The first two variants, denoted by **Replicate3** and **Replicate10**, respectively replicates the data dependencies of a submitted QTask on the 3 and 10 least loaded QBox disks among the 20 QBoxes in the platform, before applying the LocalityBased scheduling algorithm. These two variants aim at reducing the waiting time of the instances by providing more QBox candidates for the LocalityBased dispatcher. The last variant, denoted by **DataOnPlace**, instantaneously copies all data-set dependencies on all QBox disks upon the submission of a QTask. Even if it is unrealistic, this variant aims at visualising the behaviour of the standard scheduling policy without having any impact caused by the data transfers.

[2] Recall that a QTask is a group of instances of a user submitted at the same time with identical data-set dependencies.

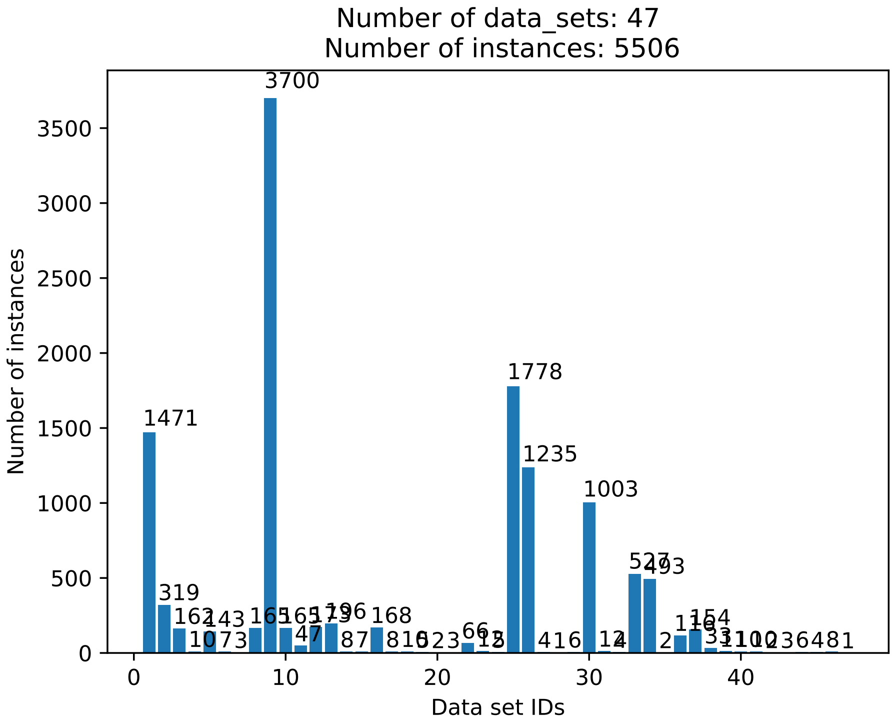


Figure 3.3: Number of instances using each data-set for the third workload.

3.6.2 Simulated Workloads

We extracted 4 different simulation inputs corresponding to logs of the Qarnot platform for a 1-week period each. Since the simulation and the scheduling algorithms are deterministic, we ran one simulation with each combination of scheduler and workload. Each simulation took less than 20 minutes to run, with about 60% of the time spent in the decision process.

The considered workloads contained between 5,000 and 9,000 instances and between 40 and 60 different data-sets. In each workload, there was at least one data-set used by 50% of the instances, and up to 7 data-sets were used by almost 500 instances out of 5506 in workload 3 (as depicted in Figure 3.3). This information shows that using replication of data-sets should improve the quality of the schedules compared to standard scheduling decisions.

In our simulations, we compared the quality of the produced schedules using the waiting time of the instances, the total number of transfers that occurred, and the total data transferred in GB. For one instance, the waiting time denotes the difference between its starting and submission times.

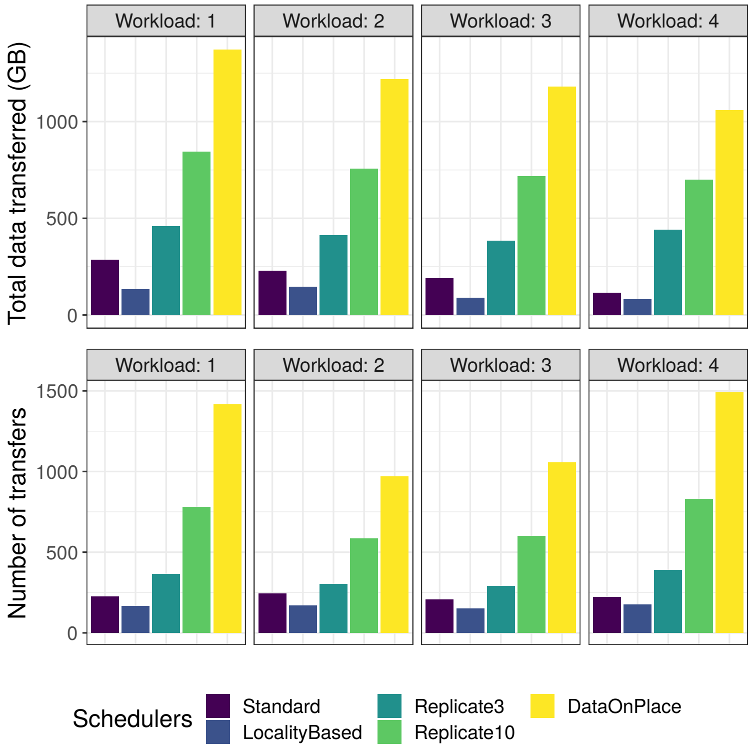


Figure 3.4: Number of transfers and total data transferred in GB.

3.6.3 Simulation Results

Figure 3.4 shows the amount of manipulated data we observed through simulation for the four workloads, while Figure 3.5 shows the waiting time distribution achieved by each scheduler, separated in 3 intervals for better clarity. Note that Figure 3.5 only shows the waiting time distribution for workload 3. Since the other workloads showed similar results, we omit the corresponding figures and focus our analysis on workload 3.

Impact of data locality

As depicted on Figure 3.5, for each scheduler, more than 60% of the instances waited less than one second before starting their execution. The last column shows that a few instances waited a long time before starting their execution (around 1,455 seconds). This is due to the long transfer time of one of their data dependencies that was as large as 36 GB, while other data-sets were smaller than 5 GB. Comparing the behaviour of Standard and LocalityBased, we do not observe a big difference in the distribution of the waiting times, except for the amount of instances that waited

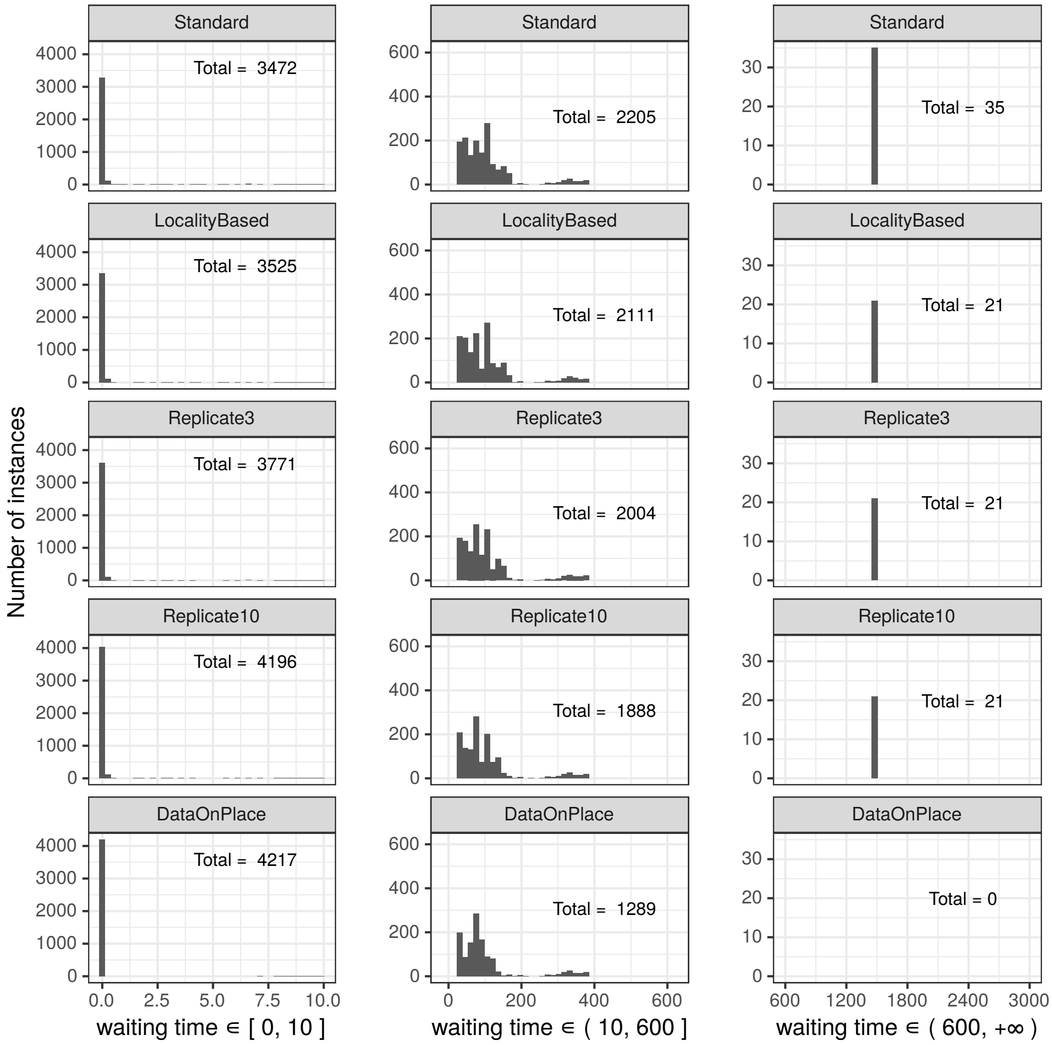


Figure 3.5: Waiting time distribution (in seconds) of all instances of the third workload.

for 1,455 seconds. This is confirmed by the average value over all instances of 39 seconds for Standard and 34.6 for LocalityBased.

Regarding the amount of data manipulated that Figure 3.4 shows, the results from the LocalityBased scheduler are as expected: dispatching instances on QBoxes already having the data-set dependencies on their disk permits to reduce the number of transfers by about 44%, and between 30 and 65% the total data transferred, compared to Standard.

To conclude, considering data locality decreases the amount of data transfer as expected but does not seem to be satisfactory enough to significantly improve the waiting times of the instances.

Transferring data has a cost

Replicating data-sets permits to reduce the mean waiting times of the instances but at a cost of more data transfers, as depicted in Figure 3.4. More precisely for the 3rd workload, the mean waiting time of the instances decreases from 34.6 to 32.6, 28.6 and 22.2 seconds respectively for LocalityBased, Replicate3, Replicate10 and the unrealistic DataOnPlace strategies. While these results look encouraging, it is important to take into account the associated overhead in terms of data transfer: from 90 GB to 384 GB for Replicate3, 599 GB for Replicate10 and 1,056 GB for DataOnPlace. This respectively corresponds to an overhead in terms of data transfer of 4.3x, 6.7x and 11.7x.

Consequently, it is not clear whether replicating data-sets at a high ratio is a valid approach. On the first hand, taking into account only the data locality is not sufficient to have good waiting time performance (LocalityBased). On the second hand, it is crucial to control data-set exchanges as they have an impact on the overall performance. For instance, it may make sense to have a replication ratio that is dynamic according to the popularity of the data-set and the status of the platform. In other words, it is crucial to also consider the time spent in data transfer before taking scheduling decisions. This is critical as the size of data-sets should be increased with respect to IoT-based scenarios envisioned by *Qarnot Computing*. In this regard, we plan to extend the Storage Controller to estimate the transfer time of a data-set to a given storage entity at a certain time. This information is valuable for the schedulers to decide when triggering data transfers and on which QBoxes. Besides, we plan to leverage our proposal to evaluate whether exchanging data-sets directly between QBoxes can help us reduce the data transfer time.

Finally, we recall that our goal through this study was not to find the best scheduling algorithm but to illustrate the use of our simulation toolkit on a concrete scenario, and to demonstrate how such a simulator would help to drive the design of scheduling and data placement strategies. Capturing the aforementioned observations in the *Qarnot Computing* production platform would have been impossible.

4 Temperature Modelling and Prediction

We study in this chapter the problem of modelling the temperature of a body and its surroundings and propose different solutions to predict the evolution of these temperatures. More precisely, we apply this problem in a smart building environment and are interested in studying the evolution of temperature of the ambient air in a room with heaters. At the end of this chapter, we would like to be able to answer the following question: *"given a room whose ambient temperature is at* $20\ °C$*, what will be the temperature after one hour if a radiator dissipates heat with a constant power consumption of* $500\ W$*?*[1]*"*

In the following, we first argument in the need for temperature prediction models and review existing works in the literature related to thermal prediction and management. Then, we introduce two basic thermodynamics formulae and the thermal model under study and propose different prediction methods to solve our problem. Finally, we conduct an experimental campaign to compare the temperature predictions of the different solutions and the execution times to compute these predictions.

This chapter presents an ongoing work in collaboration with Danilo CARASTAN-SANTOS, Anderson Andrei DA SILVA and Denis TRYSTRAM.

4.1 A Need for Temperature Prediction

With the constant increase in power consumption of the computing resources, thermal regulation has become an important part of the management of computing environments to, first, limit fault and failures of hardware components due to their operating at high temperatures and, second, to reduce the power consumption of the cooling systems.
In 2017, Ni and Bai [NB17] studied the air conditioning system of 100 data-centres

[1] Any mention of degree here and in the rest of the chapter will refer to Celsius degree, unless the unit is specified.

and showed that the total energy consumption of the cooling infrastructure accounted for, on average, 38% of the total energy consumption of the facility and that it could be even larger than the IT equipment energy consumption. In this regard, any mean of reducing the energy consumption of the cooling system is explored. For example, the usual inlet airflow temperature in data-centres is in the range 20-24 °C, but Wang et al. showed that every increase of the inlet temperature by one degree could reduce the energy consumption by 4.3% to 9.8% [WZX13].
However, increasing the inlet temperature increases the average operating temperature of the hardware components and increases as well their failure rate. Hence the need for thermal management at the level of the servers and processors.

The thermal regulation of servers in data-centres is usually performed with a feedback control loop that monitors the processors temperature and reacts to events, such as reducing the processor speed with DVFS when the temperature exceeds a given threshold. This is a *reaction*-based thermal management, which may not be appropriate to deal with high variabilities in the temperatures, as highlighted by Ramos and Bianchini [RB08]. For example, when the temperature of a server exceeds the upper limit for which the server should operate, a reaction to lower the temperature is taken. However, the reaction may not be effective fast enough and the temperature could continue to rise until a critical limit is reached, forcing to shut down the server to prevent damaging its components.

On the contrary, being able to predict the temperature of a processor or a server after a certain time period, based on the actual temperatures and power consumption, would permit *prediction*-based thermal management. In such a case, the quick rise in temperature would have been anticipated and a faster and more drastic reaction to lower the temperature of the server would have been taken.

Other use-cases of such a temperature prediction model for servers and processors is for example to compute the maximal power consumption of a processor to ensure that the temperature remains below a given threshold, or to combine with a task scheduling algorithm to take better placement decisions when the thermal/energy profile of the tasks is known.

4.2 Related Work

One can find in the literature many works to predict the temperature of processors, servers or airflow in data-centres [WB04; Cho+07; Hea+06; Zha+18; TZQ18]. One model of particular interest is the *Lumped Thermal Model* [Sub13; Woj14], that leverages the duality between heat transfer and electric current to model the heating/cooling of a body as an exponential charging/discharging process. However widely used [FMO07; Ska+04; Hua+06; POD15; Get+15; JM08; LQ10], this model assumes a constant temperature of the surroundings of the studied body during the transient process of heating or cooling, and cannot be applied to our tackled problem where the temperatures of the heater and the ambient air varies over long periods of time.

In the context of smart buildings, the forecasting of indoor temperature is usually performed with machine learning techniques and/or time series analysis using various inputs such as temperature, power consumption, humidity and light sensors [Rua+06; TS07; Pau+18; Xu+18].

A work similar to ours, by Hietaharju et al. [HRL18], presents an analytical method to predict the air temperature of large buildings from indoor and outdoor temperature and heating power values. The equations used by the authors are close to the ones presented in this chapter, but consider the heating power of the air as an input, whereas in our solution it is computed from the temperature difference between the heating device and the air and is variable depending on the inertia of the heating device.

To the best of our knowledge, we are the first to study with this work an analytical method that takes into account the power consumption of the heating device to predict temperatures of both the heater and air temperatures in a smart building. With such temperature prediction methods, we aim at proposing solutions for better simulation and thermal management of smart buildings, which could be of interest in companies such as *Qarnot Computing* and *Lancey* [@lancey].

Regarding the scheduling community, there also exist works offering temperature-aware scheduling and data placement algorithms. We think for example of Moore et al. [Moo+05] and Kaushik and Nahrstedt [KN12] that respectively propose workload and data placement algorithms that leverage hot and cool spots in data-centres to reduce the cooling costs.

Other works can be found for reducing the peak temperature in data-centres, such as task sequencing and scheduling algorithms proposed by Ramkumar and Mitra [JM08] or Liu and Qiu [LQ10].

4.3 Problem Formulation and Thermodynamics

We consider the problem of modelling and predicting the evolution of temperatures of a heater and its ambient air in a room.
The system under study can be modelled with three bodies as follows: A first body, the heater or **radiator**, consuming power that is dissipated in the form of heat, is encased inside a second body, the **ambient air** of the room, which itself is encased in the third body, the **outside world**.

From this model, we want to compute the temperature of the radiator and the ambient air of the room after a given time period from initial temperature values and a constant power consumption of the radiator, assuming that the temperature of the outside world remains constant during the process.

Before diving into the details of our temperature prediction methods, we must introduce some thermodynamic basics and formulae [Dem17].
Table 4.1 summarises the parameters and units used in the rest of the chapter.

For the simplicity of the equations and computations, we make the following usual assumptions [Woj14]:

- The temperature of the outside world remains constant;
- The temperatures within the radiator and the air bodies are uniform;
- The physical properties (mass, volume and thermal properties) of the bodies remain constant;
- The internal thermal resistance of the radiator is very low and negligible compared to the external thermal resistance between the radiator and its surrounding air (denoted by R_{rad} in Table 4.1);
- The energy consumed by the radiator is entirely dissipated in the form of heat, as no mechanical process is involved.

Moreover, we will also assume that the air body is a well-mixed fluid that can be considered as a solid body to neglect heat convection, and that its internal thermal resistance is very low and negligible compared to the external thermal resistance at its boundaries with the outside world (denoted by R_{air} in Table 4.1).

Please note that having a small ratio between the internal and external thermal resistances of a body, also denoted as the **Biot number**, is required to ensure a

Table 4.1: Summary of the notations used in the thermodynamics formulae

Parameter	Notation	Unit
Radiator temperature	T_{rad}	$°C$
Ambient air temperature	T_{air}	$°C$
Outside temperature	T_{out}	$°C$
Energy lost from radiator to the air	$E_{\text{lost\_rad}}$	J
Energy lost from air to the outside	$E_{\text{lost\_air}}$	J
Radiator power consumption	P_{rad}	W
Radiator external thermal resistance	R_{rad}	$°C/W$
Radiator thermal capacitance	C_{rad}	$J/°C$
Ambient air external thermal resistance	R_{air}	$°C/W$
Ambient air thermal capacitance	C_{air}	$J/°C$

relatively uniform temperature within the body compared to its surroundings, which is required to ensure the validity of the studied models.

Since we are working with temperature and heat, the first formula of interest is the one for **thermal energy**, or **heat capacity**. This formula links the temperature variation ΔT $[K]$ of a body of mass m $[g]$ and specific heat c $[J.g^{-1}.K^{-1}]$ to an amount of energy Q $[J]$ transferred to the body as follows[2]:

$$Q = m \times c \times \Delta T. \tag{4.1}$$

Note that the value of Q will be negative for a temperature decrease.
This formula can be rewritten into:

$$Q = C \times \Delta T \tag{4.2}$$

where $C = mc$ $[J.K^{-1}]$ is the **thermal capacitance** of the body.

When there is a temperature difference between two bodies, or between two regions of a body, heat transfer occurs from the warmer region to the colder one. This process can involve two types of heat transfer: the **heat conduction** and the **heat convection**. On the one hand, heat conduction occurs within a solid body or between a solid body and its surroundings, without matter movement. On the other hand, heat convection occurs within a fluid body where the heat is transferred through movement of the matter.

From the assumptions we made, there are in this work two energy exchanges by heat conduction: the first between the radiator and the ambient air, and the second between the air and the outside world.

[2] The Celsius degree and Kelvin units can be used interchangeably in the equations of this chapter.

The **conductive heat transfer** formula denotes the rate at which heat is transferred between two bodies based on the temperature difference between the bodies, and can be expressed as follows:

$$\frac{Q}{dt} = \frac{\Delta T}{R} \tag{4.3}$$

where Q denotes the energy transferred by conduction in dt seconds between two mediums having a temperature difference of ΔT and a **thermal resistance** of R $[K.W^{-1}]$.

Note that there is also a third type of heat transfer, the **thermal radiation**, that emits heat from a body to its surroundings. However, this radiation does not have a sufficient impact for the range of temperatures we are dealing with in this work and will thus be neglected.

4.4 Temperature Prediction Methods

We present in this section the Lumped Thermal Model and introduce different methods to solve our temperature prediction problem.
All the presented methods use a different approach to predict the **radiator temperature** and the **air temperature** after a given time period from the following inputs:

- the *time period duration* (n);
- the *power consumption* of the radiator (P_{rad});
- the *initial radiator temperature* $(T_{\text{rad}}(0))$;
- the *initial air temperature* $(T_{\text{air}}(0))$;
- the *outside temperature* (T_{out});
- the constant physical properties *thermal capacitance* and *thermal resistance* of the radiator and the air $(C_{\text{rad}}, R_{\text{rad}}, C_{\text{air}}$ and $R_{\text{air}})$.

4.4.1 Lumped Thermal Model

The *Lumped Thermal Model* [Sub13] leverages the similarity of a heat transfer between two mediums with an electric current passing through a resistor to model the transient temperature variation resulting from heat exchange between a body and its surroundings.

This model may be used to solve a simplified version of our problem where we are only interested in predicting the temperature of a body A while the temperature of the surrounding body B remains constant. This corresponds for example to predicting the temperature of a radiator while the temperature of the surrounding air remains constant, or to predicting the temperature of the ambient air with a constant temperature of the outside world. In the latter case, we assume that the energy transferred from the radiator to the air is equal to the energy consumed by the radiator, and corresponds to the internal energy gain of the air in the system composed only by the air and the outside world.

Thus, there is only one heat flow between the body of higher temperature to the body of lower temperature. If there is no internal energy gain in A (e.g., the power consumption of the heater is null), its temperature follows an exponential variation towards the temperature of B. In this case, the temperature of A after n seconds can be written as

$$T_A(n) = T_A(0) \cdot e^{-\alpha n} + T_B \cdot (1 - e^{-\alpha n})$$

where $T_A(0)$ is the initial temperature of A, T_B is the constant temperature of B and $\alpha = \frac{1}{R_{\text{rad}} \cdot C_{\text{rad}}}$ $[s^{-1}]$ is the inverse of the product of the thermal capacitance and resistance of the radiator, which can be interpreted as *"how fast the transient process is?"*

If A is the body of higher temperature, then the exponential variation will be a decrease. Conversely, if A is the body of lower temperature, then the heat flow will be from B to A and the temperature of A will follow an exponential increase.

If the internal energy gain of A is positive, the temperature of A will tend to $T_B + P_A \cdot R_A$ following the equation

$$T_A(n) = T_A(0) \cdot e^{-\alpha n} + (T_B + P_A \cdot R_A) \cdot (1 - e^{-\alpha n})$$

where P_A and R_A respectively are the power consumption and thermal resistance of A during the process, and can be rearranged as

$$T_A(n) = P_A \cdot R_A + T_B + (T_A(0) - P_A \cdot R_A - T_B) \cdot e^{-\alpha n}. \tag{4.4}$$

More details about how these equations are derived from the thermal energy and conductive heat transfer formulae, as well as extensions of this model, can be found in the *Encyclopedia of Thermal Stresses* [Woj14].

4.4.2 A Naive Iterative Approach

The first solution to solve our temperature prediction problem is a naive iterative approach, which consists in computing the energy gains and losses of the radiator and the air during one second, update the new temperatures of the radiator and the air, and repeat the process for each second in the time period. Thus, with a time period of n seconds, we will make n computation steps.

Suppose we are at step $t \in [0, n)$, the energy gained by the radiator due to its power consumption is

$$E_{\text{gained\_rad}} = P_{\text{rad}} \times 1.$$

The energy lost by the radiator is computed from the conductive heat transfer formula and is

$$E_{\text{lost\_rad}} = \frac{T_{\text{rad}}(t) - T_{\text{air}}(t)}{R_{\text{rad}}} \times 1.$$

Similarly, the energy lost by the air is

$$E_{\text{lost\_air}} = \frac{T_{\text{air}}(t) - T_{\text{out}}}{R_{\text{air}}} \times 1.$$

The new temperature of the radiator at the end of the step, using the thermal energy formula, is

$$T_{\text{rad}}(t+1) = T_{\text{rad}}(t) + \frac{E_{\text{gained\_rad}} - E_{\text{lost\_rad}}}{C_{\text{rad}}}. \tag{4.5}$$

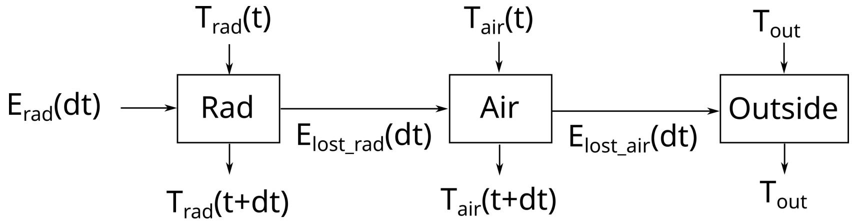


Figure 4.1: Summary of the energy transfers and temperature changes in a computation step.

Similarly, the new temperature of the ambient air is

$$T_{\text{air}}(t+1) = T_{\text{air}}(t) + \frac{E_{\text{lost\_rad}} - E_{\text{lost\_air}}}{C_{\text{air}}}. \tag{4.6}$$

Figure 4.1 summarises a computation step of dt seconds with the different energy transfers to and from the three bodies.

Finally, by setting to $T_{\text{rad}}(0)$ and $T_{\text{air}}(0)$ the initial temperatures of the radiator and the ambient air, the resulting temperatures at the end of the process are $T_{\text{rad}}(n)$ and $T_{\text{air}}(n)$.

4.4.3 A Linear Algebra Approach

The naive solution presented in the previous section is very simple. However, the number of computations and the time it takes to compute them grow linearly with the time period. We now present a second approach based on linear algebra to reduce the time complexity.

From Equations (4.5) and (4.6), we can write the temperatures of the radiator and the air in the form of sequences as

$$T_{\text{rad}}(n) = T_{\text{rad}}(n-1) - \frac{T_{\text{rad}}(n-1) - T_{\text{air}}(n-1)}{R_{\text{rad}}C_{\text{rad}}} + \frac{P_{\text{rad}}}{C_{\text{rad}}} \tag{4.7}$$

$$T_{\text{air}}(n) = T_{\text{air}}(n-1) + \frac{T_{\text{rad}}(n-1) - T_{\text{air}}(n-1)}{R_{\text{rad}}C_{\text{air}}} - \frac{T_{\text{air}}(n-1) - T_{\text{out}}}{R_{\text{air}}C_{\text{air}}} \tag{4.8}$$

with the parameters $T_{\text{rad}}(0)$ and $T_{\text{air}}(0)$ being the initial temperatures.

Then, the two sequences can be grouped into

$$\begin{pmatrix} T_{\text{rad}}(n) \\ T_{\text{air}}(n) \end{pmatrix} = A \cdot \begin{pmatrix} T_{\text{rad}}(n-1) \\ T_{\text{air}}(n-1) \end{pmatrix} + \begin{pmatrix} \frac{P_{\text{rad}}}{C_{\text{rad}}} \\ \frac{T_{\text{out}}}{R_{\text{air}}C_{\text{air}}} \end{pmatrix}$$

and then

$$\begin{pmatrix} T_{\text{rad}}(n) \\ T_{\text{air}}(n) \end{pmatrix} = A^n \cdot \begin{pmatrix} T_{\text{rad}}(0) \\ T_{\text{air}}(0) \end{pmatrix} + S_n \cdot \begin{pmatrix} \frac{P_{\text{rad}}}{C_{\text{rad}}} \\ \frac{T_{\text{out}}}{R_{\text{air}}C_{\text{air}}} \end{pmatrix} \tag{4.9}$$

where

$$A = \begin{bmatrix} 1-a & a \\ b & 1-b-c \end{bmatrix} \tag{4.10}$$

with $a = \frac{1}{R_{\text{rad}}C_{\text{rad}}}$, $b = \frac{1}{R_{\text{rad}}C_{\text{air}}}$, $c = \frac{1}{R_{\text{air}}C_{\text{air}}}$, and $S_n = (Id_2 + A + A^2 + \cdots + A^{n-1})$ with Id_2 being the identity matrix of dimension 2.

Note that, since $Det(Id_2 - A) = ac \neq 0$, $(Id_2 - A)$ is invertible and we can rewrite S_n as

$$S_n = (Id_2 - A)^{-1} \cdot (Id_2 - A^n). \tag{4.11}$$

We could stop here and simply use Equation (4.9) to compute the radiator and air temperatures since we have reduced our time complexity to a matrix exponentiation and a few side computations. However, we can continue our efforts and further reduce the complexity of computing A^n by diagonalising the matrix A. In other words, we want to find an invertible matrix P and a diagonal matrix D such that $A = PDP^{-1}$ [GV13]. This way, the exponentiation of A would reduce to the exponentiation of a diagonal matrix, which is straightforward.

To do so, we first compute the eigenvalues of A, which are the roots of the polynomial

$$\begin{aligned} p(\lambda) &= Det(A - \lambda Id_2) \\ &= (1-a-\lambda) \cdot (1-b-c-\lambda) - ab \\ &= \lambda^2 + \lambda \cdot (a+b+c-2) + (ac-a-b-c+1). \end{aligned}$$

Its discriminant is

$$\begin{aligned} \Delta &= (a+b+c-2)^2 - 4(ac-a-b-c+1) \\ &= a^2 + b^2 + c^2 + 2ab - 2ac + 2bc \\ &= (a-c)^2 + b^2 + 2ab + 2bc \end{aligned}$$

and is strictly positive. Thus, A has two eigenvalues:

$$\begin{aligned} \lambda_1 &= \frac{1}{2}(2-a-b-c-\sqrt{\Delta}) \\ \lambda_2 &= \frac{1}{2}(2-a-b-c+\sqrt{\Delta}) \end{aligned}$$

Second, we compute the eigenvector associated to each eigenvalue of A. For λ_1, its eigenvector x satisfies

$$\begin{bmatrix} 1-a-\lambda_1 & a \\ b & 1-b-c-\lambda_1 \end{bmatrix} \cdot \mathrm{x} = 0$$

leading to:

$$\begin{cases} (1-a-\lambda_1)\,x_1 + & a\,x_2 & = & 0 \\ b\,x_1 + & (1-b-c-\lambda_1)\,x_2 & = & 0 \end{cases}$$

Taking $x_2 = 1$ we obtain for the second row

$$\begin{aligned} x_1 &= \frac{\lambda_1 + b + c - 1}{b} \\ &= \frac{a - b - c + \sqrt{\Delta}}{-2b} \end{aligned}$$

which makes the eigenvector

$$V_1 = \begin{pmatrix} \frac{a-b-c+\sqrt{\Delta}}{-2b} \\ 1 \end{pmatrix}. \tag{4.12}$$

With a similar reasoning with λ_2 we get

$$V_2 = \begin{pmatrix} \frac{a-b-c-\sqrt{\Delta}}{-2b} \\ 1 \end{pmatrix}. \tag{4.13}$$

Finally, our diagonal matrix D is composed of the two eigenvalues of A on its diagonal and P is the concatenation of the two associated eigenvectors:

$$\begin{aligned} D &= \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \\ P &= \begin{bmatrix} xV_1 & xV_2 \\ 1 & 1 \end{bmatrix} \\ P^{-1} &= \frac{1}{xV_1 - xV_2} \cdot \begin{bmatrix} 1 & -xV_2 \\ -1 & xV_1 \end{bmatrix} \end{aligned}$$

with xV_1 and xV_2 respectively being the first coordinate of the eigenvectors V_1 (Equation (4.12)) and V_2 (Equation (4.13)) and we can write $A^n = PD^nP^{-1}$.

The exponentiation of A to the power n is now pretty fast and reduces to the exponentiation of λ_1 and λ_2, and a few fixed number of matrix multiplications.

We can even go a step further and use the diagonalised form to manually compute each coefficients of A^n and S_n. For A^n we have

$$A^n = \frac{1}{xV_1 - xV_2} \cdot \begin{bmatrix} xV_1 \cdot \lambda_1^n - xV_2 \cdot \lambda_2^n & xV_1 \cdot xV_2 \cdot (\lambda_2^n - \lambda_1^n) \\ \lambda_1^n - \lambda_2^n & xV_1 \cdot \lambda_2^n - xV_2 \cdot \lambda_1^n \end{bmatrix}. \tag{4.14}$$

Combining Equations (4.10), (4.11) and (4.14) we have

$$(Id_2 - A)^{-1} = \frac{1}{ac} \cdot \begin{bmatrix} b + c & a \\ b & a \end{bmatrix}$$

and

$$(Id_2 - A^n) = \frac{1}{xV_1 - xV_2} \cdot \begin{bmatrix} xV_1 \cdot (1 - \lambda_1^n) + xV_2 \cdot (\lambda_2^n - 1) & xV_1 \cdot xV_2 \cdot (\lambda_1^n - \lambda_2^n) \\ \lambda_2^n - \lambda_1^n & xV_1 \cdot (1 - \lambda_2^n) + xV_2 \cdot (\lambda_1^n - 1) \end{bmatrix}$$

which gives

$$S_n = \frac{1}{ac} \cdot \frac{1}{xV_1 - xV_2} \cdot \begin{bmatrix} s_{1,1} & s_{1,2} \\ s_{2,1} & s_{2,2} \end{bmatrix} \tag{4.15}$$

with

$$\begin{aligned} s_{1,1} &= (b + c) \cdot (xV_1 \cdot (1 - \lambda_1^n) + xV_2 \cdot (\lambda_2^n - 1)) + a \cdot (\lambda_2^n - \lambda_1^n), \\ s_{1,2} &= (b + c) \cdot xV_1 \cdot xV_2 \cdot (\lambda_1^n - \lambda_2^n) + a\,(xV_1 \cdot (1 - \lambda_2^n) + xV_2 \cdot (\lambda_1^n - 1))\,, \\ s_{2,1} &= b \cdot (xV_1 \cdot (1 - \lambda_1^n) + xV_2 \cdot (\lambda_2^n - 1)) + a \cdot (\lambda_2^n - \lambda_1^n), \\ s_{2,2} &= b \cdot xV_1 \cdot xV_2 \cdot (\lambda_1^n - \lambda_2^n) + a\,(xV_1 \cdot (1 - \lambda_2^n) + xV_2 \cdot (\lambda_1^n - 1))\,. \end{aligned}$$

Finally, we can plug Equations (4.14) and (4.15) into Equation (4.9) and unroll all matrix multiplications. We end up with two independent – rather ugly – closed-form expressions for $T_{\text{rad}}(n)$ and $T_{\text{air}}(n)$, whose time complexity consists of the exponentiation of λ_1 and λ_2, and a fixed number of scalar additions and multiplications:

$$\begin{aligned} T_{\text{rad}}(n) = T_{\text{rad}}(0) &\cdot \frac{xV_1 \cdot \lambda_1^n - xV_2 \cdot \lambda_2^n}{xV_1 - xV_2} \\ + T_{\text{air}}(0) &\cdot \frac{xV_1 \cdot xV_2 \cdot (\lambda_2^n - \lambda_1^n)}{xV_1 - xV_2} \\ + P_{\text{rad}} &\cdot \frac{s_{1,1}}{ac \cdot (xV_1 - xV_2) \cdot C_{\text{rad}}} \\ + T_{\text{out}} &\cdot \frac{s_{1,2}}{ac \cdot (xV_1 - xV_2) \cdot R_{\text{air}} \cdot C_{\text{air}}} \end{aligned} \tag{4.16}$$

$$\begin{aligned}T_{\text{air}}(n) = T_{\text{rad}}(0) &\cdot \frac{\lambda_1^n - \lambda_2^n}{xV_1 - xV_2}\\ + T_{\text{air}}(0) &\cdot \frac{xV_1 \cdot \lambda_2^n - xV_2 \cdot \lambda_1^n}{xV_1 - xV_2}\\ + P_{\text{rad}} &\cdot \frac{s_{2,1}}{ac \cdot (xV_1 - xV_2) \cdot C_{\text{rad}}}\\ + T_{\text{out}} &\cdot \frac{s_{2,2}}{ac \cdot (xV_1 - xV_2) \cdot R_{\text{air}} \cdot C_{\text{air}}}\end{aligned} \tag{4.17}$$

4.4.4 Multiple Sources Extension

We propose in this section a simple extension of the iterative and linear algebra approaches for temperature prediction with multiple heat sources in the room. The objective is to predict the temperature of the ambient air and all the radiators after a given time period based on their initial temperature, their constant power consumption and the temperature of the outside world.

Consider $K \geq 2$ radiators with, for each radiator $k \in [1, K]$, a temperature T_k, a power consumption P_k, a thermal resistance R_k and a thermal capacitance C_k. Considering that there is a heat transfer from each radiator to the ambient air, the equations for each step of the iterative approach of Section 4.4.2 can be extended as:

$$\begin{aligned}E_{\text{gained\_}k} &= P_k && \forall k \in [1, K]\\ E_{\text{lost\_}k} &= \frac{T_k(t) - T_{\text{air}}(t)}{R_k} && \forall k \in [1, K]\\ E_{\text{lost\_air}} &= \frac{T_{\text{air}}(t) - T_{\text{out}}}{R_{\text{air}}}\\ T_k(t+1) &= T_k(t) + \frac{E_{\text{gained\_}k} - E_{lost\_k}}{C_k} && \forall k \in [1, K]\\ T_{\text{air}}(t+1) &= T_{\text{air}}(t) + \frac{\sum_{k\in[1,K]} E_{\text{lost\_}k} - E_{\text{lost\_air}}}{C_{\text{air}}}\end{aligned}$$

Similarly, Equations (4.7) and (4.8) are extended to:

$$\begin{aligned}T_k(n) &= T_k(n-1) - \frac{T_k(n-1) - T_{\text{air}}(n-1)}{R_k \cdot C_k} + \frac{P_k}{C_k} && \forall k \in [1, K]\\ T_{\text{air}}(n) &= T_{\text{air}}(n-1) + \sum_{k\in[1,K]} \left(\frac{T_k(n-1) - T_{\text{air}}(n-1)}{R_k \cdot C_{\text{air}}}\right) - \frac{T_{\text{air}}(n-1) - T_{\text{out}}}{R_{\text{air}} \cdot C_{\text{air}}}\end{aligned}$$

The linear algebra approach is also extended by dealing with square matrices and vectors of dimension $K+1$. In a similar reasoning as for Equation (4.9) we can group the $K+1$ previous sequences into

$$\begin{pmatrix} T_1(n) \\ \vdots \\ T_K(n) \\ T_{\text{air}}(n) \end{pmatrix} = A_{(K)}^n \cdot \begin{pmatrix} T_1(0) \\ \vdots \\ T_K(0) \\ T_{\text{air}}(0) \end{pmatrix} + S_{(K),n} \cdot \begin{pmatrix} \frac{P_1}{C_1} \\ \vdots \\ \frac{P_K}{C_K} \\ \frac{T_{\text{out}}}{R_{\text{air}}C_{\text{air}}} \end{pmatrix}$$

where $A_{(K)}$ is the extension of matrix A with $K+1$ dimensions expressed as

$$A_{(K)} = \begin{bmatrix} 1 - \frac{1}{R_1 \cdot C_1} & 0 & \dots & 0 & \frac{1}{R_1 \cdot C_1} \\ 0 & 1 - \frac{1}{R_2 \cdot C_2} & & \vdots & \frac{1}{R_2 \cdot C_2} \\ \vdots & 0 & \ddots & 0 & \vdots \\ 0 & \dots & & 1 - \frac{1}{R_K \cdot C_K} & \frac{1}{R_K \cdot C_K} \\ b & b & \dots & b & 1 - \frac{1}{R_{\text{air}} \cdot C_{\text{air}}} - \sum_{k \in [1,K]} \frac{1}{R_k \cdot C_k} \end{bmatrix}$$

and

$$S_{(K),n} = (Id_{K+1} + A_{(K)} + A_{(K)}^2 + \cdots + A_{(K)}^{n-1}).$$

Note that it can easily be shown that $(Id_{K+1} - A_{(K)})$ is invertible and, thus, similarly to Equation (4.11), we can rewrite $S_{(K),n}$ as:

$$S_{(K),n} = (Id_K - A_{(K)})^{-1} \cdot (Id_K - A_{(K)}^n)$$

The diagonalisation of $A_{(K)}$ is however not as simple as in two dimensions and, for the sake of maintaining the reader's sanity, we will stop our exhibit of complicated formulae here.

4.5 Experiments

To assess our temperature prediction model, we conducted a set of experiments to compare the accuracy of the different prediction methods presented in the previous section, and the time the methods take, to predict the temperatures of a Qarnot Computing QRad and its ambient air.

4.5.1 Preparation of the Data

The data we used for the experiments are taken from 18 QRads of the Qarnot Computing platform deployed in the city of Bordeaux in France. We chose only a subset of the whole Qarnot platform because each of these QRads was alone in its room and embedding a single processor.
For the same privacy reasons as for the data used in the experiments of Chapter 3, we cannot provide access to the data used in this chapter.

We retrieved logs for almost 5 months between October 8th, 2019 and February 25th, 2020 containing data from different sensors that were logged every 10 seconds when the QRad was turned ON. In particular, for each timestamp in the logs, we were interested in the temperature of the ambient air and the QRad, as well as the power consumption of the whole QRad. The power consumption of the whole QRad takes into account both the consumption of the auxiliary resistors and of the processing components of the QRad. We also retrieved from the weather station of *Bordeaux Mérignac* the outside temperature taken every hour for this 5-month period.

From these data, for each QRad we cut the sequence of timestamps into subsets of contiguous timestamps with near-constant power consumption and constant outside temperature. A cut was performed if the power consumption absolute difference between the current timestamp and the first of the sequence was strictly greater than 15 W, or if the outside temperature changed. Then, for each sequence of timestamp we created an entry containing: the timestamp, the QRad and the air temperatures at the beginning and at the end of the sequence, the outside temperature and the average power consumption of the QRad during the sequence, and the length (in seconds) of the sequence.

In the logs, the air temperature is updated every 5 minutes. Thus, we only kept entries whose length was more than 300 seconds (5 minutes) to be sure a temperature difference was observed between the beginning and the end of a time period. We also completely discarded two QRads which had about 50 and 100 entries, that we judged insufficient to present relevant results.

The number of entries for the 16 remaining QRads were between 320 and 2,071. The first half of entries for each QRad was used to learn the thermal resistances of the QRad and its ambient air while the second half was used to compare the different prediction methods.

4.5.2 Learning of Physical Constants

There are multiple physical constants involved in our temperature prediction methods, namely the thermal resistance and capacitance of the radiator and of the ambient air. For the 16 QRads selected for our experiments, we know the volume of the room each QRad is placed in and we can compute the thermal capacitance C_{air} for the air in the room by multiplying the volume, density and specific heat of air, considered constant in our range of temperatures.
We took the values for the air density and the specific heat at $20\ °C$, which are respectively $1,200\ g.m^{-3}$ and $1.005\ J.g^{-1}.K^{-1}$, and computed the corresponding thermal capacitance of the air in the room of each QRad accordingly.

A QRad is an object of 28 kilograms and its main mass comes from the aluminum heat-sink. Thus, we computed the thermal capacitance C_{rad} of a QRad as the product of its mass, $28,000\ g$, and specific heat, $0.9\ J.g^{-1}.K^{-1}$.

The thermal resistance values are more difficult to compute and we used a machine learning technique to find these values. We used the non-linear regression function *curve_fit* of the *scipy optimize* Python package to learn the values of R_{rad} and R_{air} that best fit our closed-form temperature prediction function to the data, with respect to both the radiator and the air temperatures.
We learned the thermal resistance values for each QRad, and used the first 50% of entries in our prepared data.

Without surprise, we noticed that the learned values of R_{rad} were similar for all QRads, as all radiators are identical, with an average value of 0.05. Thus, we used this as a constant value for all QRad thermal resistances and ran again the curve fit optimisation to learn the R_{air} value for each QRad with this new fixed value for R_{rad}. The obtained values for the R_{air} ranged between 0.01 and 0.27.
It is worth noticing that, contrarily to that of the QRads, the thermal resistance of the air across all QRads is highly variable as it depends on many parameters, such as the configuration of the room, its orientation and sun exposure, the number of doors and windows, etc.

4.5.3 Evaluated Prediction Methods

During our experiments we compared different prediction methods. We used as input of the prediction methods the same 9 inputs as described at the beginning of Section 4.4, namely the duration of the time period, the power consumption of the radiator during the time period, the temperatures of the radiator, the air and the outside at the beginning of the time period as well as the thermal resistances and capacitances of the radiator and the air.

The first method is our proposed theoretical method introduced in Section 4.4 and is declined in four flavours:

- **Iterative:** This is the naive iterative method presented in Section 4.4.2 that uses Equations (4.5) and (4.6) to predict the radiator and air temperatures at the end of the time period.
- **Matrix:** This is the first version of the linear algebra approach presented in Section 4.4.3 and uses Equation (4.9) to predict the temperatures.
- **Diagonal:** This is the second version of the linear algebra approach leveraging the diagonalisation of matrix A. The decomposition $A^n = PD^nP^{-1}$ is plugged into Equations (4.9) and (4.11) to predict the temperatures.
- **ClosedForm:** This is the last version of the linear algebra approach where all matrix multiplications are unrolled to use the two closed-form equations (4.16) and (4.17) to predict the temperatures.

As these four versions are different representations of the same set of equations, they give the same temperature predictions. Thus, we will only use the *ClosedForm* version to compare the temperature prediction accuracy with the other methods in Section 4.5.4. A comparison of the computation time to perform the predictions of these four versions, along with the other prediction methods, will be presented in Section 4.5.5.

The second prediction method, denoted by **Lumped**, uses the *Lumped Thermal Model* presented in Section 4.4.1 to predict the temperatures. As underlined in the section, this prediction method supposes a constant temperature of the surroundings of the studied body and is not well-suited for our temperature problem as we want to predict both the temperatures of the heater and the air. However, we will use this method as a baseline since it is widely used in other temperature prediction problems.
To predict the temperature of the radiator, the power consumption of the radiator and the initial temperatures of the radiator and of the air are used as inputs of Equation (4.4). Similarly, the power consumption of the radiator, the initial temperature of the air and the temperature of the outside world are used as input of the same

equation to predict the temperature of the air.

When dealing with problems that are difficult to model and study theoretically, or with parameters that are hard to find, it is common to use machine learning approaches to try and have good approximations of the solution that can sometimes outperform methods based on real theoretical models. Our problem is not an exception. Thus, our third prediction method, denoted by **Keras**, uses neural networks to predict the temperatures.

We designed two very simple neural networks, one for the radiator and one for the air temperature, with one layer to capture the 9 inputs, no hidden layer and one layer for the output. A second version of the neural networks, denoted by **Keras2**, was designed, in which we added two dense hidden layers with 18 nodes each.

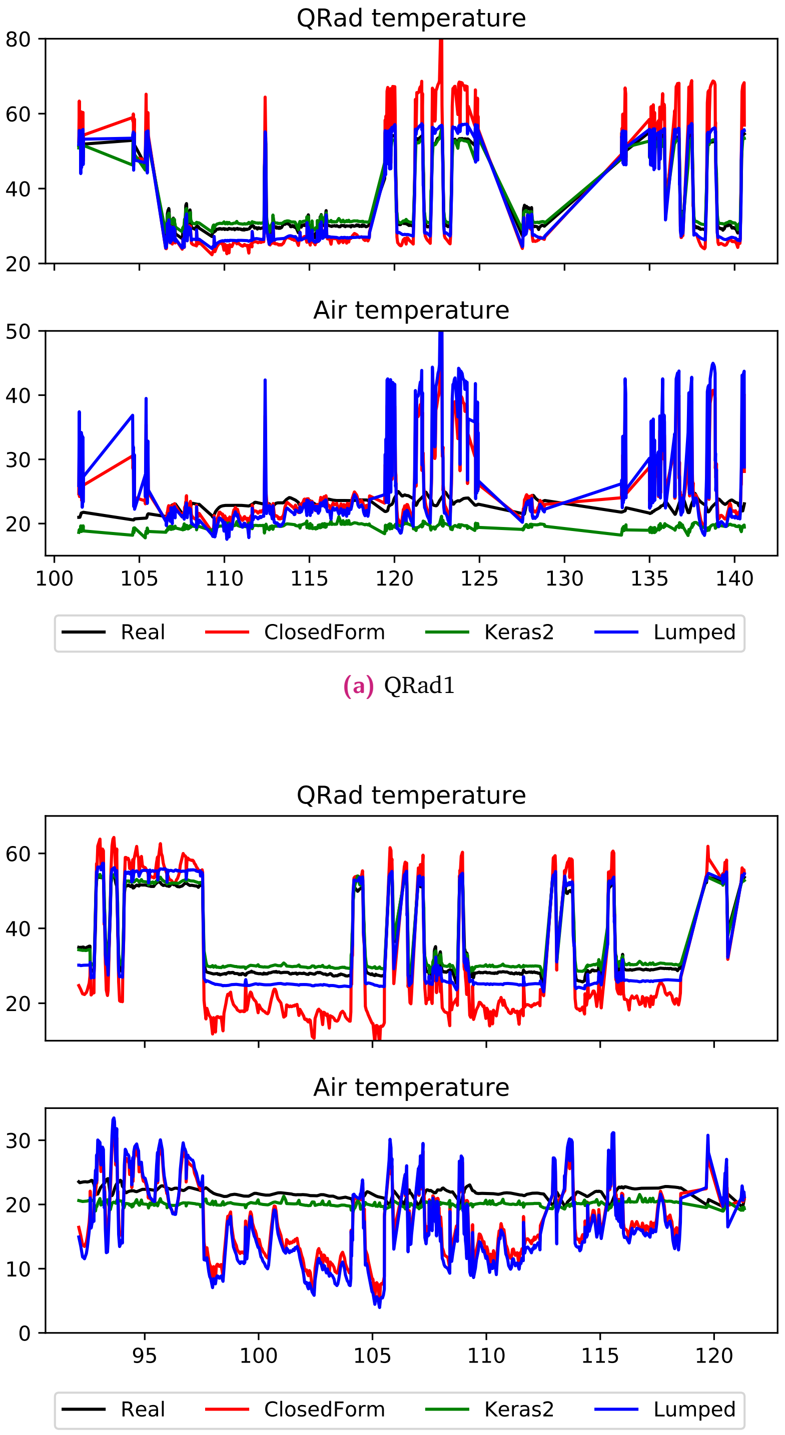
These neural networks were trained with the aggregation of the first half of entries for all QRads and the optimisation metric was the minimisation of the mean squared error between the temperature predictions and the real values, which is the most classical optimisation metric used.

We implemented all these prediction methods in modern Python (version 3.7.6), relying on the *Numpy* package (version 1.18.1) to perform the vector and matrix computations and on the *Keras* package (version 2.3.1) for our neural network solution.

4.5.4 Temperature Prediction Accuracy

We are interested in methods that predict the temperatures as close as possible to what will be the real temperatures. In other words, we want to minimise the distance between the predicted and the real temperatures for both the radiator and the ambient air. Thus, we compared the accuracy of the prediction methods using the *mean absolute error* (MAE) metric.

Figure 4.2 shows the radiator and air temperature predictions of *ClosedForm* (in red), *Keras2* (green) and *Lumped* (blue) along with the real temperature (black) for two QRads, denoted by *QRad1* and *QRad2*. We selected these two QRads as they contained long periods of activities and presented the best results for the MAE metric. The temperature predictions for *Keras* are not shown and discussed as the method was consistently outperformed by *Keras2*.



(a) QRad1

(b) QRad2

Figure 4.2: Real temperature (black) and temperature prediction of *ClosedForm* (red), *Keras2* (green) and *Lumped* (blue) for two QRads. The x-axis shows the day number since the beginning of the data acquisition (October 8th, 2019). The y-axis shows the temperature in Celsius degree.

As one can observe, all methods are able to predict the general behaviour of the QRad temperature, but with large errors for *ClosedForm* and *Lumped*. The average MAE of the prediction of all QRads for *ClosedForm* and *Lumped* are respectively of 5.61 and 2.87, with maximum values at 10.84 and 3.47, which cannot be considered as accurate for a temperature range of 20-70 $°C$. On the contrary, *Keras2* is able to predict the radiator temperature with a good accuracy with an average MAE of 0.85 with a maximum at 1.45.

For the prediction of the air temperature, the results are similar with an average MAE of 5.54, 6.8 and 1.06 for *ClosedForm*, *Lumped* and *Keras2*, with maximum values at 9.66, 11.75 and 2.15, respectively.
These results show that a simple neural network is able to predict temperatures with an average precision of almost one degree with limited amount of training data, while the two analytical methods fail to predict the temperatures with inaccuracies that can go up to 13 degrees for the air temperature in the case of QRad2.

The relative better performance of *Lumped* compared to *ClosedForm* for the prediction of the radiator temperature may appear surprising, as the Lumped Thermal Model is a simplification of our temperature model that considers a constant temperature of the air to compute the temperature of the radiator. In theory, with variations of a few degrees, or even half a degree, *ClosedForm* should provide better predictions over *Lumped*. However, the air temperature is pretty stable over the time periods of the data, with a maximum absolute difference between the air temperature at the beginning and the end of a time period of 0.17 degree over all QRads, which makes the assumption of a constant surrounding temperature for the Lumped Thermal Model appropriate.

4.5.5 Computation Time Evaluation

Having accurate temperature predictions is our main objective, but it is sometimes relevant to trade off a small performance decrease in terms of accuracy of the solution to gain a great improvement in terms of time to find the solution. Indeed, a method that would predict temperatures within 0.1 degree of the real values would be completely inefficient in practice if it always took seconds to compute.

In this section we compare the time taken by the different methods presented in Section 4.5.3 to predict temperatures. More precisely, we compute the mean time, and the *mean absolute percentage error* (MAPE) with respect to the mean time, to

Table 4.2: Computation time comparison of the temperature prediction methods. Values show the mean time in microseconds to make a prediction for a total of 1,000 predictions.

Time Period (s)	Iterative	Matrix	Diagonal	ClosedForm	Lumped	Keras2
10	7.47	205.59	246.55	6.93	3.04	417.55
60	22.79	227.46	234.98	6.89	3.06	417.52
300	97.06	244.17	237.04	6.93	3.04	420.7
600	194.16	250.35	236.24	6.99	3.01	419.0
1800	579.29	258.15	235.46	7.06	2.99	418.16
3600	1155.31	264.07	236.47	6.98	3.04	440.46
7200	2325.72	275.47	236.02	6.95	3.02	417.43
18000	5762.66	287.31	235.65	7.04	3.02	419.2
36000	11516.18	305.37	239.86	7.11	3.12	422.0

perform a temperature prediction for a total of 1,000 predictions with time periods ranging from 10 seconds to 10 hours.

Table 4.2 shows the timing results with values in microseconds. Notice that, as *Keras* and *Keras2* use the same approach with different neural networks, we will only present one result for both of them.

Without surprise, we first observe that the time to make a prediction for the iterative method is linear with the length of the time period. *Iterative* is faster than *Matrix* and *Diagonal* for time periods smaller than 600 seconds, mostly due to the static cost of creating matrices, but is outperformed by all other methods for time periods beyond $1,200$ seconds.

On the contrary, the computation time of all other methods does not seem to depend on the length of the time period, except for *Matrix* which has a slightly positive slope.

A second observation to make is that the diagonalisation of the matrix A does not seem to bring big performance increases compared to the version without diagonalisation. As Table 4.2 witnesses, the gain of the diagonalisation only appears for time periods longer than 300 seconds. A 10% speedup is achieved for time periods of one hour and a 20% of speedup is achieved only for time periods longer than 10 hours, with prediction time between 200 and 300 microseconds. This small gain in performance is probably due to the efficient implementation of matrix operations performed by the *Numpy* package. An implementation of these prediction methods in other programming languages may prove *Diagonal* much more efficient than *Matrix*.

Regarding the three methods used during the prediction accuracy comparison of the previous section, the results confirm what we guessed: having prediction methods with good accuracy has a cost. *Keras2* performs temperature predictions in about 450 microseconds with accuracy in the predictions within a single degree, while *ClosedForm* and *Lumped* perform predictions in less than 10 microseconds at a cost of greatly decreasing the temperature accuracy.
It is worth to mention that the transformation from matrices computations to closed-form formulae introduced in Section 4.4.3 enabled the temperature prediction times to decrease from about 250 microseconds to 7 microseconds with *ClosedForm*, achieving similar timing results than *Lumped*.

The MAPE metric can be interpreted as *how stable is the time required to make a prediction?* The lower MAPE was achieved by *Iterative* with values around 0.01, followed by *Matrix*, *Diagonal* and *Keras* with values around 0.03, and *ClosedForm* and *Lumped* with values between 0.1 and 0.16. It is not surprising to see such high variability for the timings of *ClosedForm* and *Lumped* as one prediction takes only a few microseconds, compared to the other methods taking more than 200 microseconds for a single prediction.

4.5.6 Results Discussion

We presented in this section preliminary experiments to compare the accuracy of the various temperature prediction methods presented in the chapter. As results showed, the two analytical methods (*ClosedForm* and *Lumped*) completely failed to predict temperatures within acceptable error ranges.

We believe that the nature of the data had a great impact on these results.
First, the data comes from QRads in the production platform, where inhabitants' actions impacted the behaviour of the QRad and the ambient temperature. The heat flow and the thermal resistance between the room and the outside world may be highly variable due to external parameters, such as someone entering the room, openning the window, or a time period when the sun directly hit the windows of the room, thus rendering the room temperature variations difficult to predict.
Second, the observed temperatures in the data are very stable. The average absolute difference of temperatures between the beginning and the end of a time period for the 16 QRads was of 0.08 degrees for the air temperature, and 1.07 for the radiator. These stable temperatures are not well-suited for the learning of the thermal resistances and capacitances, as these physical constants should reflect the behaviour of the energy transfers and temperature *variations* of the bodies. Moreover, such

stable data do not seem well-suited for the comparison of temperature prediction methods. In fact, a naive prediction method that considers the temperature as constant for the whole time period would greatly outperform the other prediction methods with these data, but would absolutely not be relevant in practice.

For these reasons, we are currently working with the Qarnot team to try and collect data of QRads in a more controlled environment with different heating and cooling phases with an amplitude of between 2 and 10 degrees, taken at different points in time with different outside temperatures. We strongly believe that, with this new data, we will be able to retrieve the physical constants by calculus and not from machine learning, and that the overall accuracy of the prediction methods will be improved, even for long time periods.
In addition, we also would like to refine our prediction models to include more semantic information about the room and the outside, such as the sun exposure or the number and area of windows. Such information could be computed in some weights to factor the inputs of the analytical prediction methods, or directly fed as inputs for the neural network solution.

5 Overview of Scheduling on Two Types of Resources

Taking a step away from the world of connected devices and Edge Computing, we will focus now on a particular resource management problem in the domain of High Performance Computing.

This chapter introduces the problem of efficiently scheduling an application represented by a precedence task graph on a parallel machine composed of two different types of computing resources to minimise the overall execution time of the application. First, we give a formal definition of the addressed scheduling problem and its variants, and expose related work. Second, we highlight the current best known algorithms for the different variants of this scheduling problem, as well as the current lower bounds on the performance guarantees for these solutions.

The work presented in this chapter is part of a collaboration with Olivier BEAUMONT, Louis-Claude CANON, Lionel EYRAUD-DUBOIS, Giorgio LUCARELLI, Loris MARCHAL, Bertrand SIMON and Denis TRYSTRAM, and led to a publication in *ACM Computing Surveys 2020* [Bea+20].

5.1 Problem Statement

The problem we tackle in this chapter is the efficient scheduling of a parallel application on a hybrid platform with the objective of minimising its completion time. The considered platform is a multi-core machine composed of two sets of resources, where each set contains identical processors.

A parallel application consists of a set of tasks that are linked by precedence relations. Each task is characterised by two processing times depending on which type of processor it is assigned to. We assume that an exact estimation of both these processing times is available to the scheduler. This assumption can be justified by several existing models to estimate the execution times of tasks [Ama+16].
In several applications, we always observe an acceleration of the tasks if they are

executed on a GPU compared to their execution on a CPU. However, we consider the more general case where the relation between the two processing times can differ for different tasks.

For this problem, we want to focus on the analysis of the qualitative behavior induced by heterogeneity since it may be assumed that the computations dominate local shared memory costs. Thus, no memory assignment or overhead for data management are considered, nor communication times between the shared memory and the processors, or between two processors of different types. Without loss of generality, we denote in the following by CPU and GPU the two types of processors, and we will assume a greater number of CPUs compared to the number of GPUs.

The main goal of this problem is, given an application to be executed on a hybrid machine, to construct a schedule that decides **when** each task of the application will be executed, and **on which** computing unit, to optimise an overall objective. As the application developers are mainly looking for performance, the objective is usually to minimise the completion time of the last finishing task, denoted by *makespan*, which is one of the most commonly studied objectives [Dro09]. In a heterogeneous context, minimising the makespan of an application corresponds to minimising the maximum between the makespan of the tasks assigned on each set of processors.

5.2 Problem Variants

We present in this section different variants of the tackled problem regarding the submission and execution of the tasks composing the parallel application to be scheduled. Two orthogonal variants are proposed, which can be combined to make 4 different version of the scheduling algorithm.

5.2.1 Dependent and Independent Tasks

The original scheduling problem considers a parallel application composed of a set of tasks that are linked by **precedence constraints**. A relation of precedence between a task A and a task B simply means that the task B cannot start its execution strictly before the task A has completed. Such constraints can model for example a data dependency between two tasks, where the task B requires data inputs that are generated by the task A.

The first variant of the problem that we consider is to relax all the precedence constraints between the tasks of the application. In such a case, all tasks are **independent** to each other and can be executed in any order.

5.2.2 Off-line and On-line Settings

Usually, scheduling problems consider that the whole set of tasks of an application and their processing times are known in advance, prior to any scheduling decision. We say in that case that the problem is **off-line**.

A second variant to consider is the **on-line** setting where the tasks arrive one by one and there is no *a priori* knowledge of the upcoming tasks. In this case, we assume that: (i) a task arrives in the system when it becomes ready, i.e., when all its predecessors have been processed; (ii) when a task arrives, its processing time on any type of resource is known to the scheduler; (iii) the scheduler must schedule a task immediately and irrevocably upon its arrival. This corresponds to the *clairvoyant on-line* context as defined by Leung [Leu04, Chapter 15]. In the particular case when multiple tasks become ready at the same time, we consider that they arrive in the system in any order. This is the case for independent tasks that are all ready at time 0.

The on-line setting is more difficult than the off-line one, but it is of particular interest in the context of large-scale heterogeneous platforms. Indeed, in the context of heterogeneous platforms, programmers of parallel applications, including for regular applications such as dense linear algebra, rely of runtime dynamic systems. These schedulers, such as Quark [YKD11], ParSeC [Bos+13], StarSs [Pla+09] and StarPU [Aug+11], make all their allocation and scheduling decisions at runtime. These on-line decisions are based on the state of the platform, the set of available (ready) tasks, and possibly on static pre-allocation strategies and task priorities that have been computed offline.

5.3 Performance Guarantees

In general, due to combinatorial explosion, it is NP-hard to find an optimal solution of an (interesting) optimisation problem. For this reason, researchers are designing algorithms to **approximate** as much as possible an optimal solution of a problem in polynomial time, instead of trying to find an optimal solution that can take an exponential time.

The scheduling theory is not an exception, and this problem is NP-hard since it is a generalisation of the scheduling problem on identical machines. Thus, we are interested in designing and studying generic algorithms that generate schedules for hybrid computing systems that provide guaranteed performance and behave well *in practice*, i.e., when they are actually used to run parallel applications.

To compare the theoretical guarantees of different algorithms, we rely on the notions of **approximation ratio** and **competitive ratio** presented by Hochbaum [HS87]. The approximation ratio ρ_A of an off-line algorithm A is defined as the maximum, over all possible instances[1] I of the considered problem, of the ratio $\frac{C_{max}(I)}{C^*_{max}(I)}$, where $C_{max}(I)$ denotes the makespan of A on the instance I and $C^*_{max}(I)$ is the optimal makespan on this instance. The competitive ratio of an on-line algorithm is defined in a similar way, comparing the algorithm makespan with the optimal *off-line* makespan. The approximation/competitive ratio of an algorithm denotes an *upper bound* on the objective value for a given instance compared to the optimal value. For example, a 3-competitive algorithm will produce schedules with a makespan being at most 3 times the makespan of an optimal schedule.
However, as we will show in Chapter 6, this upper bound is only *theoretical* and achieved for rare *worst-case* instances, while in practice the algorithms succeed in generating schedules with makespans close to the optimal ones.

5.4 Related Work

Most papers of the huge existing literature about scheduling using GPUs concern specific applications. There are only few papers dealing with generic scheduling in mixed CPU/GPU architectures, and very few of them consider precedence constraints.

Using the three-field notation for scheduling problems introduced by Graham et al. [Gra+79], our addressed problem can be denoted as $(Pm, Pk) \mid prec \mid C_{max}$, where m and k respectively denote the number of CPUs and GPUs of the machine, while the variant with independent tasks can be denoted as $(Pm, Pk) \mid\mid C_{max}$. From a theoretical perspective, these problems are harder to solve than the problems on parallel identical machines, denoted by $P \mid\mid C_{max}$ and $P \mid prec \mid C_{max}$, which are known to be NP-hard [GJ90] but admit Polynomial Time Approximation Schemes (PTAS) [HS87; HS89]. However, they are easier to solve than scheduling problems

[1] Instances of the scheduling problem, not to be confused with the Qarnot-specific QTask instances of the previous chapters

on unrelated machines ($R \,||\, C_{max}$ and $R \mid prec \mid C_{max}$) since we consider only two types of resources. Although several approximation algorithms and PTAS have been proposed for these scheduling problems on unrelated machines [LST90; ST93; SV05; GMW07], their cost makes them impractical for runtime schedulers. Moreover, PTAS have been proposed for the problems we tackle when considering a constant number Q of processor types [BW12; Geh+16], but the cost of these approaches is however prohibitive even with $Q = 2$.

A closely related problem, in which the architecture consists of $Q \geq 2$ different types of resources and each task can be executed only on some of them, has also been studied in the literature. This problem generalises the dedicated processors case if each processor consists of several identical cores, while a $(Q + 1)$-approximation algorithm has been proposed for it [LL78]. Note that, given an allocation, the problem of scheduling in hybrid machines reduces to the above generalised dedicated processors problem.

On a more practical side, there exist some work about off-line scheduling, such as the well-known algorithm HEFT introduced by Topcuoglu et al. [THW99], which has been implemented on the run-time system StarPU [Aug+11].

In the case of independent tasks, a systematic comparison of various heuristics has been performed by Braun et al. [Bra+01]. Specifically, the authors examined 11 different heuristics and provided a good basis for comparison and insights on circumstances why a technique outperforms another. Finally, experimental comparisons of the algorithms in the more recent literature for independent tasks can be found [Ble+15; BEK17; CMV17].

5.5 Best Known Solutions

We present in this section the algorithms achieving the best known performance guarantees for the 4 versions of the addressed scheduling problem.

In addition to the design of algorithms providing good approximation/competitive ratios, researchers in scheduling theory are also interested in finding **lower bounds** of these ratios. For certain scheduling problems, it is possible to design an instance that is considered as *hard* to solve, i.e., an instance for which *any* polynomial algorithm will fail to produce a schedule whose makespan is within a given factor away from the optimal makespan. For example, to prove a lower bound of 2 for a

Table 5.1: Summary of lower bounds and best known approximation or competitive ratios.

Setting		Off-line		On-line	
Independent tasks	Lower bound	-		2	[CYZ14]
	Best known	$1+\epsilon$	[Ked+18]	3.85	[CYZ14]
Tasks with precedences	Lower bound	3	[Fag+19]	$\sqrt{\frac{m}{k}}$	[Can+19]
	Best known	$3+2\sqrt{2}$	[Fag+19]	$2\sqrt{\frac{m}{k}}+1-\frac{1}{\sqrt{mk}}$	[Can+19]

given scheduling problem, one must find an instance of the problem for which all schedules constructed by an algorithm will have a makespan of at least twice the optimal makespan for this instance. Finding a lower bound on a performance ratio is a strong result, as it proves that it is not possible to design a scheduling algorithm achieving a performance ratio smaller than this lower bound.

Table 5.1 summarises the approximation ratio of the best known solution, as well as the current lower bound on this ratio, for the different versions of our scheduling problem.

First of all, since this scheduling problem is NP-hard, a trivial lower bound on the approximation ratio is 1 excluded, as an approximation ratio of 1 would mean that the algorithm is able to find an optimal solution in polynomial time.

For the simplest version of our problem, namely the off-line case with independent tasks, Kedad-Sidhoum et al. [Ked+18] proposed a methodology combining dual approximation techniques [HS87] and dynamic programming to construct algorithms of approximation ratio $1+\epsilon$, with $\epsilon > 0$ as small as desired. Thus, this problem is said to be *closed*, as there is no place for improvement in a theoretical point of view between the lower bound and the approximation ratio of the best solution.
However, although such algorithms, called PTAS, are able to produce near-optimal schedules, their time complexity is exponential in the inverse of ϵ. For this reason, other algorithms of much lower complexity have been designed at the expense of a reduced performance guarantee with an approximation ratio of 2. We think for example of the algorithms *BalancedEstimate* and *BalancedMakespan* of Canon et al. [CMV17], *CLB2C* of Cheriere and Saule [CS15] or *HeteroPrio* of Beaumont et al. [BEK17]. Notice that, to achieve an approximation ratio of 2, the two latter assume that the largest processing time of any task of the application is smaller than the optimal makespan, which is a fair assumption.

Taking a look at the problem with precedence constraints, a recent work from Fagnon et al. [Fag+19] proved a lower bound of 3 on the approximation ratio and extended the linear program-based algorithm of Kedad-Sidhoum et al. [KMT15] with a new rounding technique, improving its approximation ratio in the general case from 6 to $3 + 2\sqrt{2}$ and tending to 3 when the ratio m/k between the number of CPUs and GPUs in the machine tends to 1. The original algorithm will be further detailed and discussed in Chapter 6.

In the on-line context when tasks arrive one by one, Chen et al. [CYZ14] were the first to propose algorithms to schedule independent tasks on a hybrid machine with a 3.85-competitive algorithm in the general case with an arbitrary number of CPUs and GPUs, and a $1 + \sqrt{3}$-competitive algorithm in the balanced case where $k = m$. They also showed a simple lower bound of 2 for the competitive ratio of such algorithms.

When it comes to tasks with precedences, the on-line setting seems to become much more difficult as the performance guarantees are no longer constant values. In fact, Canon et al. [Can+19] prove that no algorithm can achieve a competitive ratio lower than $\sqrt{m/k}$ and extend the algorithm of Amaris et al. [Ama+18] to improve the competitive ratio from $4\sqrt{m/k}$ to $2\sqrt{m/k} + 1 - \frac{1}{\sqrt{mk}}$. The original algorithm for this problem will also be further detailed in Chapter 6.

For a more complete review of existing algorithms for these 4 versions of scheduling on hybrid machines, we strongly recommend the reader to read our survey on this matter [Bea+20].

6 Generic Algorithms for Scheduling Applications on Heterogeneous Platforms

This chapter details our own contributions regarding the problem of scheduling parallel applications on two resources types introduced in Chapter 5.

This work was done in collaboration with Marcos AMARIS, Giorgio LUCARELLI and Denis TRYSTRAM and was published in the *Concurrency and Computations: Practice and Experience 2020* journal [Ama+18]. A preliminary version was published in the *EuroPar 2017* conference [Ama+17].

6.1 Contribution Summary

We study both the off-line and the on-line settings of the problem of scheduling a parallel application with precedence constraints on a hybrid machine, and our goal is to design algorithms through a solid theoretical analysis that can be practically implemented in actual heterogeneous systems.

Contrarily to most existing approaches (e.g., [THW99]), we address the problem by separately focusing on the following two phases:

- **allocation**: each task is assigned to a type of resources, either CPU or GPU;
- **scheduling**: each task is assigned to a specific pair of resource and time interval respecting the decided allocation as well as the precedence constraints.

In the off-line mode, we aim to study the two phases separately motivated by the fact that there are strong lower bounds on the approximability of known single-phase algorithms. On the one hand, the approximation ratio of the well-known *Heterogeneous Earliest Finish Time* (HEFT) algorithm [THW99] cannot be better than $\Omega(\frac{m}{k^2})$ when $k \leq \sqrt{m}$ (Section 6.3). On the other hand, it can be easily shown that List Scheduling policies have arbitrarily large approximation ratios, even if we consider some enhanced order of tasks, like prioritising the task of the largest acceleration, i.e., the ratio between the processing time on a CPU and on a GPU.

The two-phase approach has been used by Kedad-Sidhoum et al. [KMT15], where a linear program (which we call *Heterogeneous Linear Program* or simply HLP) in conjunction with a rounding have been proposed for the allocation phase, while the greedy *Earliest Starting Time* (EST) policy has been applied to schedule the tasks. This algorithm, called HLP-EST, achieves an approximation ratio of 6 and we show in Section 6.3 that this ratio is tight. In fact, our worst-case example does not depend on the scheduling policy applied in the second phase.

Based on this negative result, we propose to revisit both phases. In Section 6.4.1, we replace the EST policy in HLP-EST by a specific ordering of tasks combined to a classical List Scheduling. The task ordering is based on both the allocation decisions taken during the first phase (linear program) and the critical path. This refined algorithm, denoted by HLP-OLS, preserves the tight approximation ratio of 6 and achieves good practical performances.

In Section 6.4.2 we study the on-line version of the problem, where tasks arrive in any order that respects the precedence constraints, and the scheduler has to take irrevocable decisions for their execution at the time of their arrival. We present a combination of low-complexity rules for deciding the allocation of a task upon its arrival, which takes into account the actual schedule and the relation between its processing times on both types of resources. We show that these rules, combined with List Scheduling, lead to an algorithm of competitive ratio $\Theta(\sqrt{\frac{m}{k}})$. This is the first on-line algorithm when precedence constraints are considered in the hybrid context.

In Section 6.5 we propose an extension of HLP-EST and HLP-OLS and their analysis for the case where $Q \geq 2$ types of identical processors are available. We show that both algorithms achieve a tight approximation ratio of $Q(Q+1)$.

Section 6.6 describes the generation of a benchmark used to perform the experimental evaluation of the discussed off-line and on-line algorithms, whose results are presented in Section 6.7. In the off-line setting, experiments showed that the new scheduling method based on HLP (HLP-OLS) outperforms HLP-EST on both contexts with 2 or 3 resource types with an average improvement in the makespan of 10%. Comparisons with HEFT showed that HLP-OLS offered similar performances with an improvement of 2% on average for 2 resource types, but a decrease of 4% on average for 3 resource types. In the on-line setting, results showed that our proposed algorithm outperformed a greedy policy by an average improvement of 16% but was outperformed by the Earliest Finish Time (EFT) policy by 10% on average.

6.2 Definitions and Notations

We consider a parallel application that should be scheduled on m identical CPUs and k identical GPUs. Without loss of generality, we assume that $m \geq k$. The application is represented by a *Directed Acyclic Graph* $G = (V, E)$ whose nodes correspond to sequential tasks and arcs correspond to precedence relations among the tasks. We denote by $\mathcal{T}$ the set of all tasks. By slightly abusing the notation, we can write $\mathcal{T} \equiv V$. The execution of a task needs a different amount of time if it is performed by a CPU or by a GPU. Let $\overline{p_j}$ (resp. $\underline{p_j}$) be the processing time of a task T_j if it is executed on any CPU (resp. GPU). Given a schedule S, we denote by C_j the completion time of a task T_j in S. In any feasible schedule, for each arc $(i, j) \in E$, the task T_j cannot start its execution before the completion of T_i. We say that T_i is a **predecessor** of T_j and we denote by $\Gamma^-(T_j)$ the set of all predecessors of T_j. Similarly, we say that T_j is a **successor** of T_i and we denote by $\Gamma^+(T_i)$ the set of all successors of T_i. We call **descendant** of T_j each task T_i for which there is a path from j to i in G.

The objective is to create a feasible non-preemptive schedule of minimum makespan. In other words, we seek a schedule that respects the precedence constraints among tasks, does not interrupt their execution and minimises the completion time of the last finishing task, i.e., $C_{\max} = \max_{T_j}\{C_j\}$. Extending the three-fields notation for scheduling problems introduced by Graham, this problem can be denoted as $(CPU, GPU) \mid prec \mid C_{\max}$.

6.3 Preliminaries

In this section, we briefly present the two basic existing approaches for scheduling on heterogeneous/hybrid platforms and we discuss their theoretical efficiency by presenting lower bounds on their performance guarantee.

The first approach is the scheduling-oriented algorithm HEFT [THW99]. According to HEFT, the tasks are initially prioritised with respect to their precedence relations and their average processing times. Then, following this priority, tasks are scheduled with possible back-filling on the available pair of processor and time interval in

Table 6.1: Sets of tasks composing the instance for which HEFT achieves an approximation ratio of $\frac{m+k}{k^2}\left(1-\frac{1}{e^k}\right)$.

Set of tasks	# tasks per set	$\overline{p_j}$	$\underline{p_j}$
A_i, $1 \leq i \leq m$	k	$\left(\frac{m}{m+k}\right)^i$	$\left(\frac{m}{m+k}\right)^i$
B_i, $1 \leq i \leq m$	m	$\left(\frac{m}{m+k}\right)^i$	$\frac{k}{m^2}\left(\frac{m}{m+k}\right)^m$

which they feasibly complete as early as possible. Note that HEFT is a heuristic that works for platforms with several heterogeneous resources and also takes into account possible communication costs. However, even for the simpler setting without communication costs, with only two types of resources and $k = 1$, HEFT cannot have a worst-case approximation guarantee better than $\frac{m}{2}$ [Ble+15]. This result depends only on the number of CPUs, since the example provided uses just one GPU. The following theorem slightly improves the above result for the case of a single GPU. More interestingly, it expresses the lower bound to the approximation ratio of HEFT using both the number of CPUs and of GPUs.

Theorem 1. *For any $k \leq \sqrt{m}$, the worst-case approximation ratio for HEFT is at least $\frac{m+k}{k^2}\left(1-\frac{1}{e^k}\right)$, even in the hybrid model with independent tasks.*

Proof. We describe an instance that consists of independent tasks, and hence no communication costs are defined. We also consider the hybrid platform model where we only have a set of m identical CPUs and a set of k identical GPUs. Then, the rank of each task $T_j \in \mathcal{T}$ computed by HEFT is simplified as follows:

$$rank(T_j) = \frac{m\overline{p_j} + k\underline{p_j}}{m+k}$$

HEFT considers the tasks in decreasing order with respect to their rank and assigns each task to the CPU or GPU where its completion time is minimised. In case of ties, we assume, without loss of generality, that HEFT prefers to assign the task to a GPU, while it chooses arbitrarily between CPUs or GPUs. Notice that, since all tasks are independent, no idle times are introduced in the schedule.

Our instance consists of $2m$ sets of $km + m^2$ tasks in total, as shown in Table 6.1. The rank of each task $T_j \in A_i$, $1 \leq i \leq m$, is

$$rank(T_j) = \frac{(m+k)\left(\frac{m}{m+k}\right)^i}{m+k}$$

while the rank of each task $T_j \in B_i$, $1 \leq i \leq m$, is

$$rank(T_j) = \frac{m\left(\frac{m}{m+k}\right)^i + \frac{k^2}{m^2}\left(\frac{m}{m+k}\right)^m}{m+k}.$$

According to the above ranks, HEFT will schedule all tasks in A_{i+1} (resp. B_{i+1}) after all tasks in A_i (resp. B_i), $1 \leq i \leq m-1$.
Moreover, for any $T_j \in A_i$ and $T_{j'} \in B_i$, $1 \leq i \leq m$, we have:

$$\begin{aligned}(m+k)\left(rank(T_j) - rank(T_{j'})\right) &= (m+k)\left(\frac{m}{m+k}\right)^i - m\left(\frac{m}{m+k}\right)^i - \frac{k^2}{m^2}\left(\frac{m}{m+k}\right)^m \\ &= k\left(\left(\frac{m}{m+k}\right)^i - \frac{k}{m^2}\left(\frac{m}{m+k}\right)^m\right) \\ &\geq k\left(\left(\frac{m}{m+k}\right)^m - \frac{k}{m^2}\left(\frac{m}{m+k}\right)^m\right) \\ &> 0\end{aligned}$$

where the last inequality holds since $k \leq m$.
For any $T_j \in B_i$ and $T_{j'} \in A_{i+1}$, $1 \leq i \leq m-1$, we have:

$$\begin{aligned}(m+k)\left(rank(T_j) - rank(T_{j'})\right) &= m\left(\frac{m}{m+k}\right)^i + \frac{k^2}{m^2}\left(\frac{m}{m+k}\right)^m - (m+k)\left(\frac{m}{m+k}\right)^{i+1} \\ &> \left(\frac{m}{m+k}\right)^i\left(m - (m+k)\frac{m}{m+k}\right) \\ &= 0\end{aligned}$$

Based on the above equations, HEFT will consider the sets of tasks according to the following order:

$$A_1 \prec B_1 \prec A_2 \prec B_2 \prec \cdots \prec A_i \prec B_i \prec A_{i+1} \prec \cdots \prec A_m \prec B_m$$

Initially, HEFT will schedule the k tasks in A_1 in a different GPU. Hence, to minimise the completion times of the m tasks in B_1, each one should be scheduled on a different CPU. Note that, all tasks in $A_1 \cup B_1$ finish at the same time, i.e., at time $\frac{m}{m+k}$. Similarly, the tasks in A_2 will be scheduled on a different GPU, the tasks in B_2 on a different CPU, and all of them will finish at the same time, i.e., at time $\frac{m}{m+k} + \left(\frac{m}{m+k}\right)^2$.

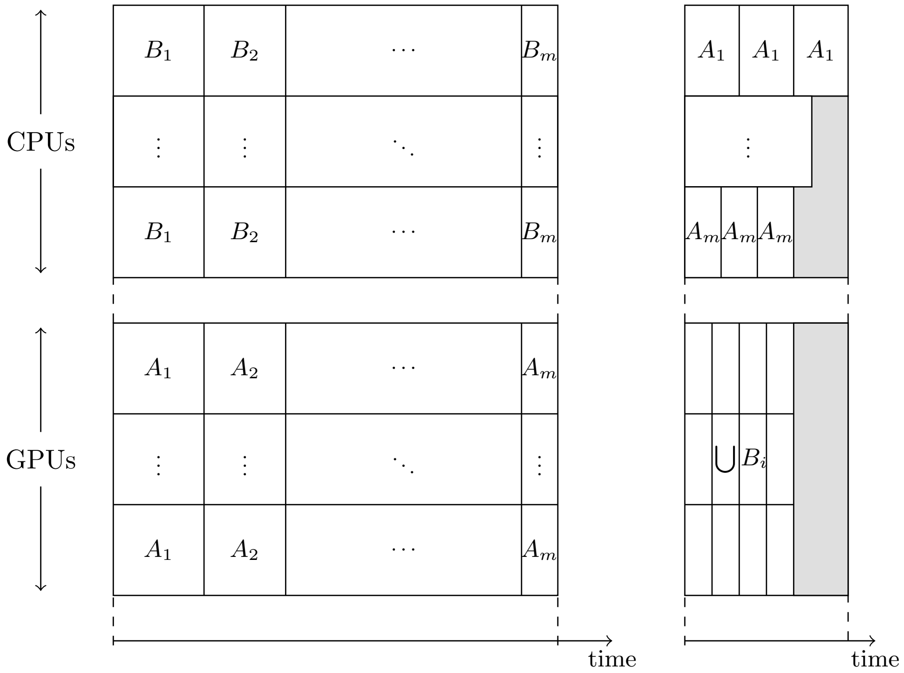


Figure 6.1: Possible schedule of HEFT (left) and optimal schedule (right). Notice that the gray area represents idle times.

The scheduling procedure continues in the same way for the tasks in the remaining sets. The left-hand side of Figure 6.1 shows a schedule produced by HEFT. In this schedule, all machines finish their execution at time

$$\begin{aligned}
\sum_{i=1}^{m}\left(\frac{m}{m+k}\right)^{i} &= \frac{1-\left(\frac{m}{m+k}\right)^{m+1}}{1-\frac{m}{m+k}} - 1 \\
&\simeq \frac{1-\frac{m}{m+k}\frac{1}{e^k}}{1-\frac{m}{m+k}} - 1 \\
&= \frac{(m+k)e^k - m}{ke^k} - 1 \\
&= \frac{me^k - m}{ke^k} \\
&= \frac{m}{k}\left(1-\frac{1}{e^k}\right).
\end{aligned}$$

On the other hand, we can create a schedule of makespan at most $\frac{km}{m+k}$. To see this, we assign all tasks of A_i, $1 \le i \le m$, on CPU i, while to each of the k GPUs we assign $\frac{m^2}{k}$ different tasks of $\bigcup_{i=1}^{m} B_i$. The right-hand side of Figure 6.1 visualises such a schedule, whose makespan is dominated either by the load of CPU 1 or by the load of any of the GPUs. Specifically, the makespan will be equal to:

$$\max\left\{k\left(\frac{m}{m+k}\right), \frac{m^2}{k}\frac{k}{m^2}\left(\frac{m}{m+k}\right)^m\right\} \leq \frac{km}{m+k}$$

Since an optimal schedule could have an even smaller makespan the theorem follows. □

The second approach for scheduling on hybrid platforms is proposed by Kedad-Sidhoum et al. [KMT15] and distinguishes the allocation and the scheduling decisions. For the allocation phase, an integer linear program is proposed which decides the allocation of tasks to the CPU or GPU side by optimising the standard lower bounds for the makespan of a schedule which are proposed by Graham [Gra69], namely the critical path and the load. To present this integer linear program, let x_j be a binary variable which is equal to 1 if a task T_j is assigned to the CPU side, and zero otherwise. Let also C_j be a variable that indicates the completion time of T_j and λ the variable that corresponds to the maximum over all lower bounds used, i.e., to a lower bound of the makespan. Then, the Heterogeneous Linear Program (HLP) is as follows:

$$\begin{aligned}
&\text{minimise } \lambda \text{ subject to:} && \\
&C_i + \overline{p_j}x_j + \underline{p_j}(1-x_j) \leq C_j && \forall T_j \in \mathcal{T}, T_i \in \Gamma^-(T_j) && (6.1)\\
&\overline{p_j}x_j + \underline{p_j}(1-x_j) \leq C_j && \forall T_j \in \mathcal{T} : \Gamma^-(T_j) = \emptyset && (6.2)\\
&C_j \leq \lambda && \forall T_j \in \mathcal{T} && (6.3)\\
&\frac{1}{m}\sum_{T_j\in\mathcal{T}} \overline{p_j}x_j \leq \lambda && && (6.4)\\
&\frac{1}{k}\sum_{T_j\in\mathcal{T}} \underline{p_j}(1-x_j) \leq \lambda && && (6.5)\\
&x_j \in \{0,1\} && \forall T_j \in \mathcal{T} && (6.6)\\
&C_j \geq 0 && \forall T_j \in \mathcal{T} &&
\end{aligned}$$

Constraints (6.1), (6.2) and (6.3) describe the critical path, while Constraints (6.4) and (6.5) impose that the makespan cannot be smaller than the average load on CPU and GPU sides. Note that the particular problem of deciding the allocation to minimise the maximum over the three lower bounds is NP-hard, since it is a generalisation of the PARTITION problem to which it reduces if all tasks are independent, $m = k$, and $\overline{p_j} = \underline{p_j}$ for each T_j.

Table 6.2: Sets of tasks composing the instance for which HLP-EST achieves an approximation ratio of $6 - O(\frac{1}{m})$.

Set of tasks	# tasks per set	$\overline{p_j}$	$\underline{p_j}$
A	1	$\frac{m(2m+1)}{m-1}$	∞
B_1	$2m+1$	$2m-1$	1
B_2	$2m+1$	1	$2m-1$

After relaxing the integrality Constraint (6.6), a fractional allocation can be found in polynomial time. To get an integral solution, the variables x_j are rounded as follows: if $x_j \geq \frac{1}{2}$ then T_j is assigned to the CPU side, otherwise T_j is assigned to the GPU side.

Finally, the Earliest Starting Time (EST) policy is applied for scheduling the tasks: at each step, the ready task with the earliest possible starting time is scheduled respecting the precedence relations and the decided allocation. We call this algorithm HLP-EST.

HLP-EST achieves an approximation ratio of 6 [KMT15]. The following theorem shows that this ratio is tight.

Theorem 2. *There is an instance for which HLP-EST achieves an approximation ratio of* $6 - O(\frac{1}{m})$. *Hence, the ratio for HLP-EST is tight.*

Proof. Consider a hybrid system with an equal number of CPUs and GPUs, i.e, $m = k$. The instance consists of $2m + 3$ tasks that are partitioned into 3 sets as shown in Table 6.2.

The only precedence relations exist between tasks of B_1 and B_2. Specifically, for each task $T_j \in B_2$ we have that $\Gamma^-(T_j) = B_1$, that is no task in B_2 can be executed before the completion of all tasks in B_1. Note that there are no precedences between tasks of the same set.

Any optimal solution of the relaxed HLP for the above instance will assign the task T_A on a CPU, i.e., $x_A = 1$. Hence, the objective value of any optimal solution will be at least $\frac{m(2m+1)}{m-1}$ due to Constraints (6.2) and (6.3). The following technical proposition shows that an optimal solution for the relaxed HLP has exactly this objective value, by describing a feasible fractional assignment for the remaining tasks.

Proposition 1. *There is a small constant $\epsilon > 0$ for which the assignment $x_A = 1$, $x_j = \frac{1}{2}$ for each $T_j \in B_1$, $x_j = \frac{1}{2} - \epsilon$ for each $T_j \in B_2$, and $\lambda = \frac{m(2m+1)}{m-1}$ corresponds to a feasible solution for the relaxed HLP.*

Proof. We will show that every constraint of the relaxed HLP is satisfied by the assignment of the binary variables x_j proposed in the statement and by setting $\lambda = \frac{m(2m+1)}{m-1}$. But before this, we need to give feasibly value for C_j, for each $T_j \in \mathcal{T}$, based on Constraints (6.1) and (6.2).

For the task T_A, we set

$$C_A = \frac{m(2m+1)}{m-1},$$

for each task $T_j \in B_1$, we set

$$C_j = \frac{1}{2}(2m-1) + \frac{1}{2} = m,$$

and for each task $T_j \in B_2$, we set

$$C_j = m + (\frac{1}{2} - \epsilon) + (\frac{1}{2} + \epsilon)(2m-1) = 2m + 2\epsilon(m-1),$$

satisfying by definition Constraints (6.1) and (6.2).

To show the feasibility of Constraint (6.3), it suffices to prove it for T_A as well as for a task $T_j \in B_2$. For these cases, we have

$$C_A = \frac{m(2m+1)}{m-1} = \lambda \text{ and}$$
$$C_j = 2m + 2\epsilon(m-1) \leq \lambda,$$

where the last inequality holds for arbitrarily small ϵ, and hence Constraint (6.3) is satisfied.

For Constraint (6.4), we have

$$\begin{aligned}\sum_{T_j \in \mathcal{T}} \overline{p_j} x_j &= \overline{p_A} x_A + \sum_{T_j \in B_1 \cup B_2} \overline{p_j} x_j \\ &= \frac{m(2m+1)}{m-1} + (2m+1)\frac{2m-1}{2} + (2m+1)(\frac{1}{2} - \epsilon) \\ &< \frac{m(2m+1)}{m-1} + m(2m+1) \\ &= m\frac{m(2m+1)}{m-1} \\ &= m\lambda\end{aligned}$$

and hence it is satisfied.

For Constraint (6.5), we have

$$\begin{aligned}\sum_{T_j \in \mathcal{T}} \underline{p_j}(1 - x_j) &= \underline{p_A}(1 - x_A) + \sum_{T_j \in B_1 \cup B_2} \underline{p_j}(1 - x_j) \\ &= 0 + (2m+1)\frac{1}{2} + (2m+1)(2m-1)(\frac{1}{2} + \epsilon) \\ &< m(2m+1) + \epsilon(4m^2 - 1) \\ &\leq m\lambda \\ &= k\lambda\end{aligned}$$

where the last inequality is true for an arbitrarily small ϵ, and hence the constraint is satisfied.

Concluding, all constraints are satisfied with $\lambda = \frac{m(2m+1)}{m-1}$, and thus the proposition holds. □

Given the optimal fractional assignment proposed above, HLP-EST will round the fractional variables and allocate the tasks as follows: the task T_A is assigned to the CPU side, each task $T_j \in B_1$ is assigned to the CPU side, and each task $T_j \in B_2$ is assigned to the GPU side. Then, HLP-EST schedules the tasks according to the EST policy. However, we will argue here for any scheduling policy and thus any possible schedule.

Assuming that an algorithm has scheduled the task T_A on any CPU during any interval $[t, t + \overline{p_A})$ and $m \geq 3$, there is only one meaningful family of schedules for the tasks in $B_1 \cup B_2$. Specifically, the $2m + 1$ tasks of B_1 will be scheduled during

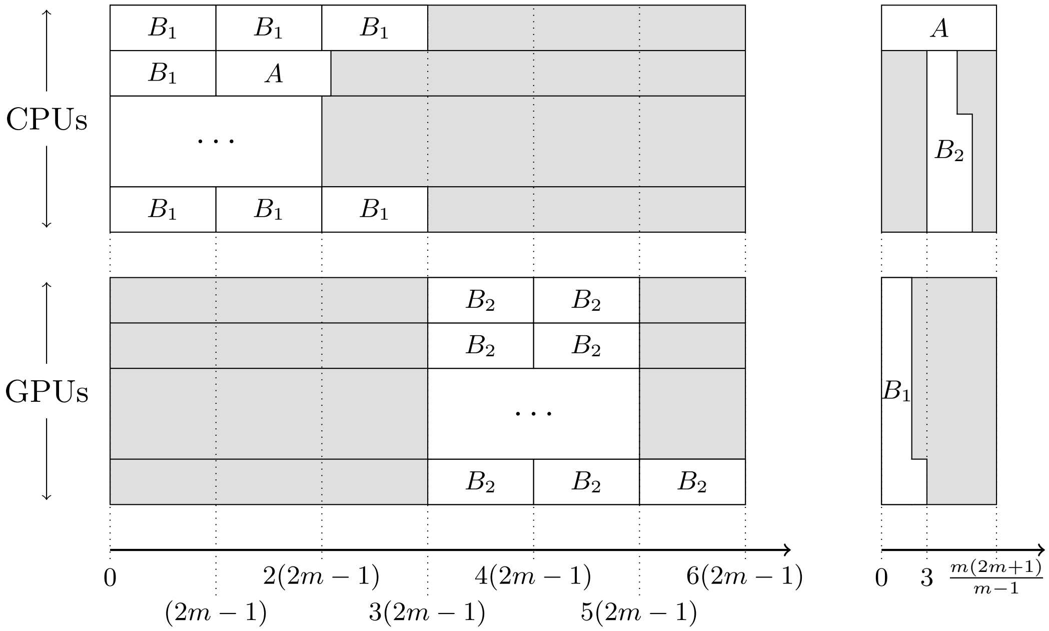


Figure 6.2: Resulting schedule of HLP-EST (left) and optimal schedule (right) for the proposed instance. Notice that the gray areas represent idle times.

the interval $[0, 3(2m-1))$ on the m CPUs, while at least one of them completes at time $3(2m-1)$. Then, the $2m+1$ tasks of B_2 will be scheduled during the interval $[3(2m-1), 6(2m-1))$ on the $k=m$ GPUs, while at least one of them completes at time $6(2m-1)$. Clearly, we should define t such that $t + \overline{p_A} \leq 6(2m-1)$. An illustration of the above schedule is given in the left-hand side of Figure 6.2.

The makespan of the created schedule is equal to $6(2m-1)$, while an optimal schedule, depicted in the right-hand side of Figure 6.2, places T_A on a CPU, all tasks of B_1 on the $k=m$ GPUs and all tasks of B_2 on the remaining $m-1$ CPUs, achieving a makespan of $\frac{m(2m+1)}{m-1}$. Hence, the approximation ratio achieved for this instance is

$$\frac{6(2m-1)}{\frac{m(2m+1)}{m-1}} = 6 - O\left(\frac{1}{m}\right)$$

and the theorem follows. □

Note that the proof of the previous theorem implies a stronger result since the worst case example does not depend on which scheduling policy will be applied after the allocation step, and hence the following corollary holds.

Corollary 1. *Any scheduling policy which is applied after the allocation decisions taken by the rounding of an optimal solution of the relaxed HLP leads to an approximation algorithm of ratio at least* $6 - O(\frac{1}{m})$.

6.4 Algorithms

In this section, we propose algorithms for both the off-line and on-line settings of the addressed problem.

6.4.1 Off-line Setting

We propose in the following a new scheduling policy, which prioritises the tasks based on the solution obtained for HLP after the rounding step. The motivation of assigning priorities to the tasks is for taking into account the precedence relations between them. More specifically, we want to prioritise the scheduling of *critical tasks*, i.e., the tasks on the critical path, before the remaining (less critical) tasks.

To do this, we define for each task T_j a rank $Rank(T_j)$ in the same sense as in HEFT. However, in our case, the rank of each task depends on the allocation given by HLP, while in HEFT it depends on the average processing time of the task. Specifically, the rank of each task T_j is computed after the rounding operation of the assignment variable x_j and corresponds to the length, in the sense of processing time, of the longest path between this task and its last descendant in the precedence graph. Thus, each task will have a larger rank than all its descendants. The rank of the task T_j is recursively defined as follows:

$$Rank(T_j) = \overline{p_j} x_j + \underline{p_j}(1 - x_j) + \max_{i \in \Gamma^+(T_j)} \{Rank(T_i)\}$$

After ordering the tasks in decreasing order with respect to their ranks, we apply the standard List Scheduling algorithm adapted to two types of resources and taking into account the rounding of the assignment variables x_j. We call the above described policy Ordered List Scheduling (OLS), while the newly defined algorithm (including the allocation) is denoted by HLP-OLS.

Although this policy performs well in practice, as we will see in Section 6.7, its approximation ratio cannot be better than 6 due to the lower bound presented in Theorem 2. On the other hand, it is quite easy to see that HLP-EST and HLP-OLS have the same approximation ratio by following the same reasoning as in Lemmas 4 and 5 of Kedad-Sidhoum et al. [KMT15].

Consider the schedule produced by HLP-OLS and partition the time interval $I = [0, C_{max})$ into two subsets I_{CP} and I_W. The set I_{CP} contains every time slot where at least one processor of each type is idle, while the set I_W consists of the remaining time slots in I, i.e., $I_W = I \backslash I_{CP}$. We then can divide the set I_W into two possibly non-disjoint subsets I_{CPU} (resp. I_{GPU}) containing the time slots where all the CPUs (resp. GPUs) are busy. Denoting by $|I|$ the number of unitary time slots in an interval I we have

$$\begin{aligned} |I_{CP}| &\leq CP, \\ |I_{CPU}| &\leq \frac{W_{CPU}}{m} \text{ and} \\ |I_{GPU}| &\leq \frac{W_{GPU}}{k} \end{aligned}$$

where CP, W_{CPU} and W_{GPU} denote respectively the length of the critical path, the total work load on all CPUs and the total work load on all GPUs based on the assignment decided by the rounding of the fractional solution of HLP. Because of the rounding, each of these three quantities are bounded above by twice the objective value λ of the optimal solution of HLP. Since λ is smaller than the feasible optimal makespan C^*_{max}, we deduce the following bound for the makespan of HLP-OLS:

$$\begin{aligned} C_{max} &\leq |I_{CPU}| + |I_{GPU}| + |I_{CP}| \\ &\leq \frac{W_{CPU}}{m} + \frac{W_{GPU}}{k} + CP \\ &\leq 6C^*_{max} \end{aligned}$$

Corollary 2. *HLP-OLS achieves an approximation ratio of 6. This ratio is tight.*

6.4.2 On-line Setting

In the HLP-EST algorithm, an integer linear program is used to find an efficient allocation of each task to the CPU or GPU side. Although this program optimises the classical lower bounds for the makespan, and hence informally optimises the allocation, the resolution of its relaxation has a high complexity in practice and cannot be used in an on-line setting.

In what follows, we present an algorithm for the on-line scheduling problem in the hybrid context. Our algorithm first decides the allocation of a task to a resource type upon its arrival by using a set of rules. These rules take into account both the actual schedule and the relation between the processing times of a task, in a similar way as in the 4-competitive algorithm proposed by Chen et al. [CYZ14] for the on-line problem with independent tasks. Then, a List Scheduling is applied to schedule each task respecting the decided allocation and precedence relations.

To describe the new rules, we define τ_{gpu} to be the earliest time when at least one GPU is idle. Let also $R_{j,gpu} = \max\{\tau_{gpu}, \max_{T_i \in \Gamma^-(T_j)}\{C_i\}\}$ be the *ready time* of T_j for GPUs, i.e., the earliest time at which T_j can be executed on a GPU. Then, the set of rules is defined as follows:

Step 1: If $\overline{p_j} \geq R_{j,gpu} + \underline{p_j}$ then assign T_j to the GPU side;

Step 2: Else if $\frac{\overline{p_j}}{\sqrt{m}} \leq \frac{\underline{p_j}}{\sqrt{k}}$ then assign T_j to the CPU side. Otherwise assign T_j to the GPU side.

Intuitively, the rule of Step 2 tries to balance the load between the two types of resources, while Step 1 ensures that the GPU side is favoured for the tasks that are greatly accelerated on GPUs.

This set of rules is combined with a greedy List Scheduling policy that schedules each task as early as possible on the CPU or GPU side already decided by the rules. We call the algorithm obtained by this combination ER-LS (Enhanced Rules - List Scheduling). In the following, we give upper and lower bounds for the competitive ratio of ER-LS.

Theorem 3. *ER-LS achieves a competitive ratio of* $4\sqrt{\frac{m}{k}}$.

Proof. Let W_{CPU}, W_{GPU} and CP be the total load on all CPUs, the total load on all GPUs and the length of the critical path produced by the allocation of tasks decided by the rules, respectively. With the same reasoning as in Section 6.4.1, for the makespan of the schedule produced by ER-LS we can observe that

$$C_{max} \leq \frac{W_{CPU}}{m} + \frac{W_{GPU}}{k} + CP. \qquad (6.7)$$

In the following, we bound the average load of both sides ($\frac{W_{CPU}}{m} + \frac{W_{GPU}}{k}$) by $3\sqrt{\frac{m}{k}}C^*_{max}$ and the length of the critical path by $\sqrt{\frac{m}{k}}C^*_{max}$. Recall that C^*_{max} denotes the makespan of the optimal off-line solution of the instance.

We denote by $\mathcal{SA}_{cpu}$ (resp. $\mathcal{SA}_{gpu}$) the set containing the tasks placed on the CPU (resp. GPU) side in both a solution of the algorithm and the optimal solution, by $\mathcal{SB}_{gpu}$ the set containing tasks placed by Step 1 on the GPU side in a solution of the algorithm but on the CPU side in the optimal solution, and by $\mathcal{SC}_{cpu}$ (resp. $\mathcal{SC}_{gpu}$) the set containing tasks placed by Step 2 on the CPU (resp. GPU) side in a solution of the algorithm but on the GPU (resp. CPU) side in the optimal solution.

We also denote by sa_{cpu}, sa_{gpu}, sb_{gpu}, sc_{cpu} and sc_{gpu} the sum of processing times of all tasks in the sets $\mathcal{SA}_{cpu}$, $\mathcal{SA}_{gpu}$, $\mathcal{SB}_{gpu}$, $\mathcal{SC}_{cpu}$ and $\mathcal{SC}_{gpu}$, respectively. Note that we use here the processing times according to the allocation of ER-LS.

Bounding the loads.
Consider T_{j_0} to be the last finishing task in $\mathcal{SB}_{gpu}$. Since the task is scheduled according to Step 1, we know that $\overline{p_{j_0}} \geq R_{j_0,gpu} + \underline{p_{j_0}} \geq \frac{sb_{gpu}}{k}$. We also know that T_{j_0} is scheduled on a CPU in the optimal solution so we have $\overline{p_{j_0}} \leq C^*_{max}$ and thus: $\frac{sb_{gpu}}{k} \leq C^*_{max}$.

Each task in $\mathcal{SC}_{gpu}$ is scheduled on the CPU side in the optimal solution. According to Step 2, the total processing time of tasks in $\mathcal{SC}_{gpu}$ in the optimal solution is at least $\sqrt{\frac{m}{k}}sc_{gpu}$, so we have for the cpu side $\frac{sa_{cpu}+\sqrt{\frac{m}{k}}sc_{gpu}}{m} \leq C^*_{max}$. The same reasoning for the GPU side gives $\frac{sa_{gpu}+\sqrt{\frac{k}{m}}sc_{gpu}}{k} \leq C^*_{max}$.

By adding the three inequalities we have the following:

$$\frac{sb_{gpu}}{k} + \frac{sa_{cpu} + \sqrt{\frac{m}{k}}sc_{gpu}}{m} + \frac{sa_{gpu} + \sqrt{\frac{k}{m}}sc_{cpu}}{k} \leq 3C^*_{max}$$

By separating the loads on CPU and on GPU on the left-hand side of the above inequality and taking into account that $m \geq k$ we have

$$\frac{sa_{cpu}}{m} + \frac{sc_{cpu}}{\sqrt{mk}} \geq \frac{sa_{cpu} + sc_{cpu}}{m} \geq \sqrt{\frac{k}{m}}\frac{sa_{cpu} + sc_{cpu}}{m}$$

and

$$\frac{sa_{gpu} + sb_{gpu}}{k} + \frac{sc_{gpu}}{\sqrt{mk}} = \frac{sa_{gpu} + sb_{gpu}}{k} + \frac{sc_{gpu}}{k}\sqrt{\frac{k}{m}} \geq \sqrt{\frac{k}{m}}\frac{sa_{gpu} + sb_{gpu} + sc_{gpu}}{k}.$$

Summing these two bounds we finally obtain

$$\sqrt{\frac{k}{m}}(\frac{sa_{cpu} + sc_{cpu}}{m} + \frac{sa_{gpu} + sb_{gpu} + sc_{gpu}}{k}) \leq 3C^*_{max}$$

and thus

$$\frac{W_{CPU}}{m} + \frac{W_{GPU}}{k} \leq 3\sqrt{\frac{m}{k}}C^*_{max}. \tag{6.8}$$

Bounding the critical path.

Consider the sets $\mathcal{SA}^{CP}_{cpu} \subseteq \mathcal{SA}_{cpu}$, $\mathcal{SA}^{CP}_{gpu} \subseteq \mathcal{SA}_{gpu}$, $\mathcal{SB}^{CP}_{gpu} \subseteq \mathcal{SB}_{gpu}$, $\mathcal{SC}^{CP}_{cpu} \subseteq \mathcal{SC}_{cpu}$ and $\mathcal{SC}^{CP}_{gpu} \subseteq \mathcal{SC}_{gpu}$ to be the sets containing only the tasks belonging to the critical path obtained by the algorithm, with the same notation in lower case for the sum of processing times of all tasks in each set and the same notation with a star * for the sum of processing times of all tasks in the optimal solution.

For the sets $\mathcal{SA}^{CP}_{cpu}$ and $\mathcal{SA}^{CP}_{gpu}$, by definition, as placement is the same, we have:

$$sa^{CP}_{cpu} + sa^{CP}_{gpu} = sa^{CP^*}_{cpu} + sa^{CP^*}_{gpu}$$

According to Step 1, every task in $\mathcal{SB}^{CP}_{gpu}$ has a processing time smaller than in the optimal solution, so $sb^{CP}_{gpu} \leq sb^{CP^*}_{gpu}$. According to Step 2, every task T_j in $\mathcal{SC}^{CP}_{cpu}$ (resp. $\mathcal{SC}^{CP}_{gpu}$) verifies $\overline{p_j} \leq \sqrt{\frac{m}{k}}\underline{p_j}$ (resp. $\underline{p_j} \leq \sqrt{\frac{k}{m}}\overline{p_j}$), so we have $sc^{CP}_{cpu} \leq \sqrt{\frac{m}{k}}sc^{CP^*}_{cpu}$ and $sc^{CP}_{gpu} \leq \sqrt{\frac{m}{k}}sc^{CP^*}_{gpu}$.

By summing the previous inequalities for the critical path we get:

$$\begin{aligned} CP &= sa^{CP}_{cpu} + sa^{CP}_{gpu} + sb^{CP}_{gpu} + sc^{CP}_{cpu} + sc^{CP}_{gpu} \\ &\leq \sqrt{\frac{m}{k}}(sa^{CP^*}_{cpu} + sa^{CP^*}_{gpu} + sb^{CP^*}_{gpu} + sc^{CP^*}_{cpu} + sc^{CP^*}_{gpu}) \\ &\leq \sqrt{\frac{m}{k}}CP^* \end{aligned}$$

Since $CP^* \leq C^*_{max}$, we have $CP \leq \sqrt{\frac{m}{k}}C^*_{max}$ and, combining this inequality with Equations (6.7) and (6.8), the theorem follows. □

Theorem 4. *There is an instance for which ER-LS achieves a competitive ratio of* $\sqrt{\frac{m}{k}}$.

Proof. Consider a hybrid system with m CPUs and $k \leq m$ GPUs. The instance consists of $m + k$ tasks that are partitioned into 2 sets as shown in Table 6.3.

Table 6.3: Sets of tasks composing the instance for which ER-LS achieves a competitive ratio of $\sqrt{\frac{m}{k}}$.

Set of tasks	# tasks per set	$\overline{p_j}$	$\underline{p_j}$
A	k	$\sqrt{m}$	$\sqrt{m}$
B	m	$\sqrt{m}$	$\sqrt{k}$

The k tasks of type A are independent to each other and the m tasks of type B are with precedence constraints as follows:

$$B_1 \prec B_2 \prec \cdots \prec B_m$$

The tasks are ordered in a list by first taking all tasks of type A and then the tasks of type B respecting the precedences.

The ER-LS algorithm will first place the k tasks of type A on a GPU according to Step 1. The completion time of these tasks is then $\sqrt{m}$. Then, since $\sqrt{m} \leq \sqrt{m} + \sqrt{k}$, the task B_1 will be placed on a CPU according to Step 2, with completion time $\sqrt{m}$. The task B_2 will also be placed on a CPU according to Step 2, starting at time $\sqrt{m}$ and completing at time $2\sqrt{m}$. With the same reasoning, each task B_i, $i \in \{1, m\}$ is placed on a CPU according to Step 2 starting at time $(i-1)\sqrt{m}$ and completing at time $i\sqrt{m}$. Thus, the schedule produced by ER-LS for this instance has a makespan of $C_{max} = m\sqrt{m}$.

An optimal schedule would have all tasks of type A placed on the CPU side with a completion time for each task of $\sqrt{m}$. The tasks of type B would be placed on the GPU side with a completion time for each task B_i, $i \in \{1, m\}$, of $i\sqrt{k}$. Thus, the optimal makespan is $C^*_{max} = m\sqrt{k}$.

Hence, ER-LS achieves a competitive ratio $\frac{C_{max}}{C^*_{max}} = \sqrt{\frac{m}{k}}$ for this instance and the theorem holds. □

As Theorems 3 and 4 showed, there is a gap between the lower and upper bound on the competitive ratio of ER-LS. In fact, as mentioned in Chapter 5, Canon et al. [Can+19] showed that the upper bound could be reduced to $2\sqrt{m/k} + 1 - \frac{1}{\sqrt{mk}}$ by removing Step 1 of the algorithm, and that the ratio of this new algorithm was almost tight.
Note also that if the number of CPUs and GPUs are close, then the competitive ratio of ER-LS becomes constant.

6.5 Generalisation on Q Resource Types

In this section, we generalise the addressed scheduling problem for $Q \geq 2$ different types of identical processors. Specifically, we explain how to extend HLP-EST and HLP-OLS, which are designed for the case $Q = 2$, to handle more types of computing resources. In this direction, we first present a modified linear program and a new rounding method to decide the allocation for $Q \geq 2$. Then, we explain the modifications made on the scheduling policy OLS. Note that the EST policy is directly generalised for multiple resource types. Finally, we give an upper bound on the approximation ratio of these two new algorithms.

Before continuing, we need some additional notations. Let M_q be the set of processors of type q, $1 \leq q \leq Q$, and $m_q = |M_q|$ its size. The execution of a task $T_j \in \mathcal{T}$ on a processor of type q, $1 \leq q \leq Q$, takes $p_{j,q}$ time units.

In what follows, we extend the linear program HLP to take into account more resource types. To do this, we replace the binary variable x_j by $x_{j,q}$ which indicates if the task $T_j \in \mathcal{T}$ is assigned to the resource type q. As before, let C_j be a variable corresponding to the completion time of T_j and λ be the variable that represents a lower bound to the makespan. Then, we consider the following modification of HLP, which we call QHLP:

minimise λ subject to:

$$C_i + \sum_{q=1}^{Q} p_{j,q} x_{j,q} \leq C_j \qquad \forall T_j \in \mathcal{T}, T_i \in \Gamma^-(T_j) \qquad (6.9)$$

$$\sum_{q=1}^{Q} p_{j,q} x_{j,q} \leq C_j \qquad \forall T_j \in \mathcal{T} : \Gamma^-(T_j) = \emptyset \qquad (6.10)$$

$$C_j \leq \lambda \qquad \forall T_j \in \mathcal{T} \qquad (6.11)$$

$$\frac{1}{m_q} \sum_{T_j \in \mathcal{T}} p_{j,q} x_{j,q} \leq \lambda \qquad 1 \leq q \leq Q \qquad (6.12)$$

$$\sum_{q=1}^{Q} x_{j,q} = 1 \qquad \forall T_j \in \mathcal{T} \qquad (6.13)$$

$$x_{j,q} \in \{0,1\} \qquad \forall T_j \in \mathcal{T},\ 1 \leq q \leq Q \qquad (6.14)$$

$$C_j \geq 0 \qquad \forall T_j \in \mathcal{T}$$

$$\lambda \geq 0$$

The main difference here concerns Constraint (6.13), which assures that each task is integrally assigned to exactly one type of resources.

After relaxing the integrality Constraint (6.14) of QHLP, we can solve in polynomial time the obtained relaxation. To get an integral allocation, we assign each task T_j to the resource type q' for which the assignment variable $x_{j,q}$ has the greatest value, i.e., $q' = \text{argmax}_{1 \leq q \leq Q}\{x_{j,q}\}$. In other words, for such q' we set $x_{j,q'} = 1$ and $x_{j,q} = 0$ for any $q \neq q'$. In case of ties, we give priority to the resource type for which T_j has the smallest processing time. Once the assignment step is done, we use the Earliest Starting Time policy taking into account the precedence constraints as well as the allocation provided by the rounding of $x_{j,q}$ variables. We call this algorithm QHLP-EST.

For the scheduling policy OLS, we modify the computation of the rank for each task as follows:

$$Rank(T_j) = \sum_{q=1}^{Q} p_{j,q} x_{j,q} + \max_{i \in \Gamma^+(T_j)} \{Rank(T_i)\}$$

The algorithm HLP-OLS can be extended using the linear program QHLP and the new rounding method to get an integral allocation of the tasks. We then obtain the algorithm QHLP-OLS using the modified OLS policy.

In the following, we show that the approximation ratio of QHLP-EST is $Q(Q+1)$ and that this ratio is tight. Note that, as in the case $Q = 2$ presented in Section 6.4.1, the same reasoning can be applied for QHLP-OLS leading to the same result.

Theorem 5. *QHLP-EST achieves an approximation ratio of $Q(Q+1)$. This ratio is tight.*

Proof. We analyse the structure of a schedule produced by the algorithm to give an upper bound on the approximation ratio.

We denote by W_q, $1 \leq q \leq Q$, the total load on all processors of type q in the obtained schedule. We also denote by C^R_{max}, W^R_q and L^R the objective value, the total load on all processors of type q and the length of the longest path in the fractional optimal solution of the relaxed QHLP, respectively. Finally, we define by

$C^*_{\max}$ the optimal makespan over all feasible schedules for our problem. Then, the following inequalities hold:

$$L^R \leq C^R_{\max} \leq C^*_{\max} \tag{6.15}$$

$$\frac{W^R_q}{m_q} \leq C^R_{\max} \leq C^*_{\max},\ 1 \leq q \leq Q \tag{6.16}$$

To analyse the structure of the schedule, we partition the time interval of the schedule $I = [0, C_{max})$ into two disjoint subsets of intervals I_{CP} and I_W. The set I_{CP} contains every time slot where at least one processor of each type is idle, while the set I_W consists of the remaining time slots in I, i.e., $I_W = I \backslash I_{CP}$. We then can divide the set I_W into Q, possibly non-disjoint, subsets I_q, $1 \leq q \leq Q$, which contain respectively every time slot where all processors of type q are busy. Henceforth, we denote by $|I|$ the length of I, i.e. the number of unitary time slots in I. Then, we have:

$$C_{\max} = |I| \leq |I_{CP}| + \sum_{q=1}^{Q} |I_q|$$

In the following, we will bound above by QC^*_{max} the length of the subset I_{CP} and each subset I_i, $1 \leq i \leq Q$.

Due to the rounding policy, we know that if $x_{j,q} = 1$ then $x^R_{j,q} \geq \frac{1}{Q}$. Hence, we have:

$$x_{j,q} \leq Q \cdot x^R_{j,q} \quad \forall T_j \in \mathcal{T}, 1 \leq q \leq Q \tag{6.17}$$

Consider first the subset of intervals I_{CP}. There is a directed path $\mathcal{P}$ of tasks being executed during any time slot in I_{CP}. The construction of $\mathcal{P}$ is the same as described by Graham [Gra69] and Kedad-Sidhoum *et al.* [KMT15]. Since the directed path $\mathcal{P}$ covers every time slot in I_{CP}, the length of I_{CP} is smaller than the length of $\mathcal{P}$

and the length of $\mathcal{P}$ in the optimal solution of $QHLP$, noted $\mathcal{P}^R$, is smaller than L^R. Thus, using the inequalities (6.15) and (6.17), we have the following bound:

$$\begin{aligned}
|I_{CP}| &\leq |\mathcal{P}| \\
&\leq \sum_{j\in\mathcal{P}} \sum_{q=1}^{Q} p_{j,q} x_{j,q} \\
&\leq Q \sum_{j\in\mathcal{P}} \sum_{q=1}^{Q} p_{j,q} x^R_{j,q} \\
&= Q \cdot |\mathcal{P}^R| \\
&\leq Q \cdot L^R \\
&\leq Q \cdot C^*_{\max}
\end{aligned}$$

Consider now each subset I_q, $1 \leq q \leq Q$. For each time slot in I_q all processors of type q are busy, so $|I_q|$ is smaller than the average load on all the processors of type q. Using the inequalities (6.16) and (6.17), we have the following bounds:

$$\begin{aligned}
|I_q| &\leq \frac{W_q}{m_q} \\
&\leq \frac{1}{m_q} \sum_{x_{j,q}=1} p_{j,q} \\
&\leq \frac{Q}{m_q} \sum_{j\in V} p_{j,q} x^R_{j,q} \\
&\leq Q \cdot \frac{W^R_q}{m_q} \\
&\leq Q \cdot C^*_{\max}
\end{aligned}$$

Thus, by combining the calculated bounds we get:

$$\begin{aligned}
C_{\max} &= |I| \\
&\leq |I_{CP}| + \sum_{q=1}^{Q} |I_q| \\
&\leq Q(Q+1)C^*_{max}
\end{aligned}$$

The tightness for $Q = 2$ comes directly from Theorem 2. For any value $Q > 2$, the instance given in Theorem 2 can be easily extended to prove that the approximation ratio given above is tight. Hence, the theorem follows. □

6.6 Benchmark Creation

In this section, we describe the generation of a general benchmark for evaluating scheduling algorithms for the addressed problem. The benchmark is composed of 5 applications generated by *Chameleon*, a dense linear algebra software which is part of the MORSE project [Agu+12], and a more irregular application (*fork-join*) generated using *GGen*, a library for generating directed acyclic graphs [Cor+10].

The applications of *Chameleon*, named *getrf*, *posv*, *potrf*, *potri* and *potrs*, are composed of multiple sequential basic tasks of linear algebra. Different numbers, denoted by nb_blocks, and sizes, denoted by $block\_size$, of sub-matrices have been used for the applications; specifically we set $nb\_blocks \in \{5, 10, 20\}$ and $block\_size \in \{64, 128, 320, 512, 768, 960\}$, for a total of 18 instances per application. Table 6.4 shows the total number of tasks for each application and each value of nb_blocks. Notice that the value of $block\_size$ does not impact the number of tasks.

For the setting with 2 resource types, the applications were executed with the runtime StarPU [Aug+11] on a machine with two Dual core Xeon E7 v2 with a total of 10 physical cores with hyper-threading of 3 GHz and 256 GB of RAM. The machine had 4 GPUs NVIDIA Tesla K20 (Kepler architecture) with each 5 GB of memory and 200 GB/s of bandwidth.

For 3 resource types, the applications were executed with the runtime StarPU on an Intel Dual core i7-5930k machine with a total of 6 physical cores with hyper-threading of 3.5 GHz and 12 GB of RAM. This machine had 2 NVIDIA GPUs: a GeForce GTX-970 (Maxwell architecture) with 4 GB of memory and 224 GB/s of bandwidth; and a Quadro K5200 (Kepler architecture) with 8 GB of memory and 192 GB/s of bandwidth. We forced each task to run first on CPU and then on GPU (or on both GPUs for the case with 3 resource) and stored the processing times for each type of resource.

The *fork-join* application corresponds to a real situation where the execution starts sequentially and then forks to $width$ parallel tasks. The results are aggregated by performing a join operation, completing a phase. This procedure can be repeated p times, the number of phases. For this benchmark, we set $p \in \{2, 5, 10\}$ and $width \in \{100, 200, 300, 400, 500\}$, for a total of 15 instances.

The processing time of each task on CPU was computed using a Gaussian distribution with center p and standard deviation $\frac{p}{4}$. To generate the processing times of a task on each of the two types of GPUs, we define the notion of *acceleration factor* with

Table 6.4: Number of tasks for each instance of the Chameleon applications.

# blocks	Application getrf_nopiv	posv	potrf	potri	potrs
5	55	65	35	105	30
10	385	330	220	660	110
20	2870	1960	1540	4620	420

Table 6.5: Number of tasks for each instance of the fork-join application.

# phases	Width 100	200	300	400	500
2	203	403	603	803	1003
5	506	1006	1506	2006	2506
10	1011	2011	3011	4011	5011

respect to the processing time on CPU already computed. Our goal is to create a more irregular application than the 5 from Chameleon to study the importance of the allocation decision. For this reason, 5% of the parallel tasks of each phase are highly decelerated when executed on a GPU by choosing uniformly at random an acceleration factor for each of them in $[0.1, 0.5]$. For each of the remaining tasks, we randomly chose an acceleration factor in $[0.5, 50]$, which corresponds to the range of acceleration factors observed for the 5 applications of Chameleon. Note that the 5% of highly decelerated tasks are separately selected at random for each of the two types of GPUs. Table 6.5 shows the total number of tasks for each instance of the fork-join application.

The data-sets and other information are available under Creative Commons Public License [@bench].

6.7 Experiments

In this section, we compare the performance of various scheduling algorithms by a simulation campaign using a benchmark composed of 6 parallel applications, presented in Section 6.6. First, we compare the off-line algorithms for the studied problem with 2 and 3 resource types. We then compare the on-line algorithms for 2 resource types.

6.7.1 Off-line Setting

Algorithms and machine configurations.
We compared the performance, in terms of makespan, of HLP-OLS (Section 6.4.1) with HLP-EST (Section 6.3) and HEFT ([THW99]). The algorithms were implemented in Python (v. 2.7.6). The command-line *glpsol* (v. 4.52) solver of the GLPK package was used for the linear program. Each algorithm was implemented with a second version adapted for 3 types of resources, using the generalisation of the algorithms presented in Section 6.5 for the 2 linear program-based algorithms. We denote by QHLP-EST, QHLP-OLS and QHEFT these algorithms for 3 resource types.

For the machine settings, we determined different sets of pairs (Nb_CPUs, Nb_GPUs). Specifically, we used 16, 32, 64 and 128 CPUs with 2, 4, 8 and 16 GPUs for a total of 16 machine settings for the case with 2 resource types. For the case with 3 resource types, we determined different sets of triplets (Nb_CPUs, Nb_GPU1s, Nb_GPU2s) with the same numbers of CPUs and for either types of GPUs, for a total of 64 machine settings.

We define a *configuration* to be a combination of an instance of application and a machine setting. Recall that an instance of an application defines its parameters, such as the number and size of the blocks for a Chameleon application, or the number and width of the phases for the fork-join application. We executed the algorithms only once with each configuration since all algorithms are deterministic. For each run, we stored the optimal objective solution of the linear program, denoted by LP^*, and the makespans of the six algorithms. For the application instance giving the highest number of tasks, solving the linear program took about 100 seconds while the running time of each algorithm took at most 10 seconds, once a solution of HLP was found for the linear program-based algorithms.

Results for 2 resource types.
To study the performance of the 3 algorithms, we computed the ratio between each makespan and the optimal solution LP^* of the linear program HLP, which corresponds to a good lower bound of the optimal makespan. Fig. 6.3 shows the ratio of each configuration. Notice that the bigger dot represents the mean value of the ratio for each application. We can see that HLP-EST is outperformed, on average, by the two other algorithms. The performances of HLP-OLS and HEFT are quite similar, on average, but we observe that HEFT creates more outliers.

Fig. 6.4 compares in more detail the two HLP-based algorithms (top), and the algorithms HLP-OLS and HEFT (bottom), by showing the ratio between the makespans

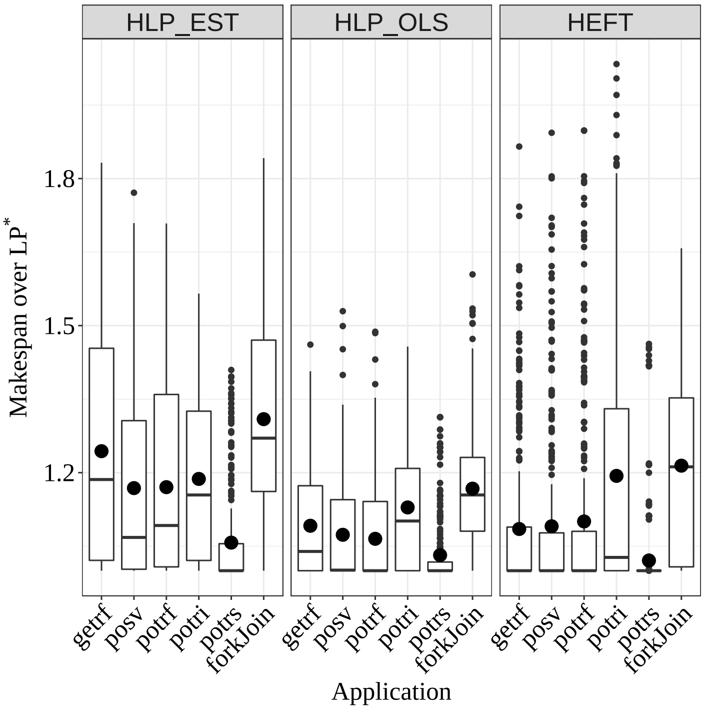


Figure 6.3: Ratio of makespan over LP^* for each configuration grouped by application for the off-line algorithms with 2 resource types. Notice that the big square shows the mean, for each application.

of the two algorithms. We can see that HLP-OLS outperforms HLP-EST, except for a few configurations with the application *potri*, with an improvement close to 8% on average. Comparing HLP-OLS and HEFT we notice that, even if the two algorithms have similar performances, HEFT is on average outperformed by HLP-OLS by 2%, with a maximum of 60% of improvement for HLP-OLS with some configurations of *potri*. Moreover, HEFT has a significantly worse performance than HLP-OLS in strongly heterogeneous applications where there is a bigger perturbation in the (dis-)acceleration of the tasks on the GPU side, like *forkJoin*, since in these irregular cases the allocation problem becomes more critical.

Results for 3 resource types.
To study the performance of the 3 algorithms, we computed the ratio between each makespan and the optimal solution LP^* of the linear program QHLP, which corresponds to a good lower bound of the optimal makespan. Fig. 6.5a shows the ratio of each configuration. Notice that the bigger dot represents the mean value of the ratio for each application. We can see that QHLP-EST is on average outperformed by the two other algorithms. We also observe that even if QHEFT presents many

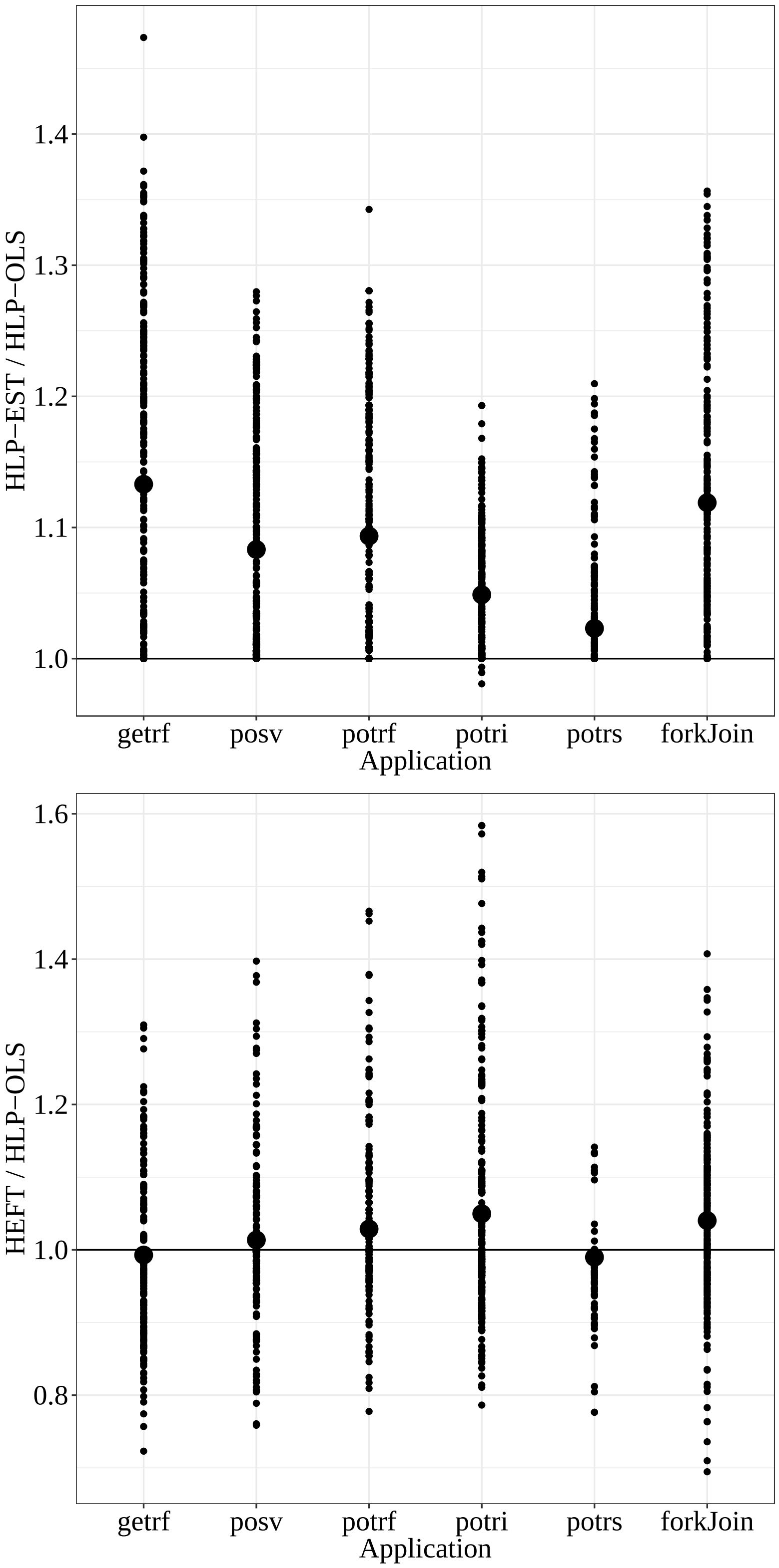
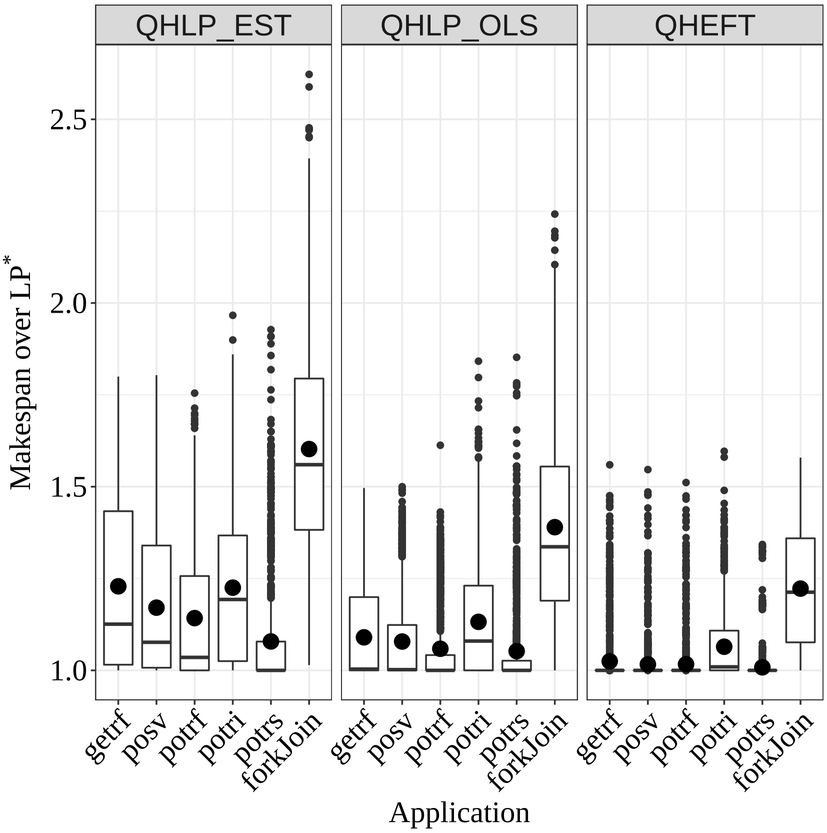
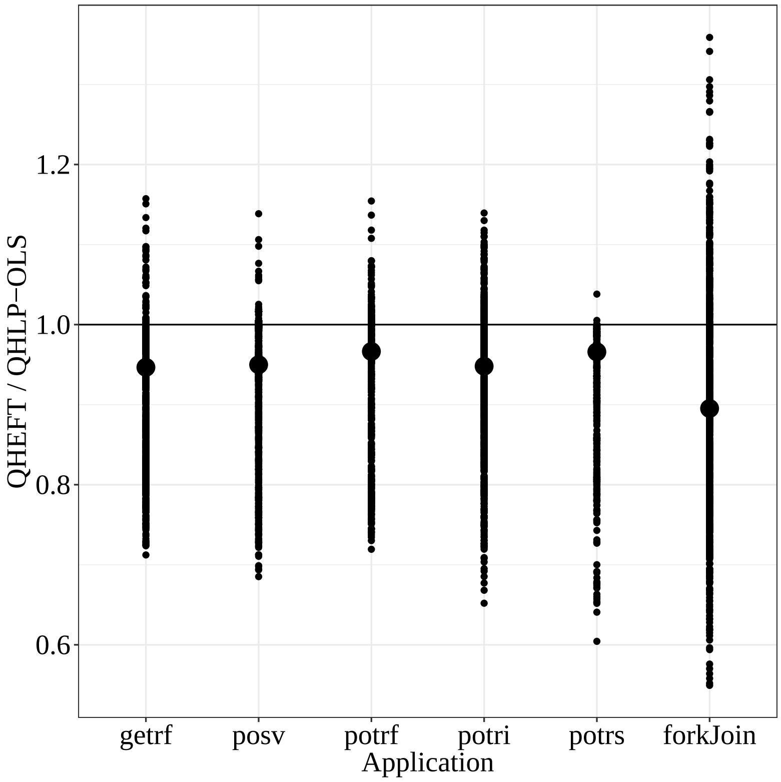


Figure 6.4: Ratio between the makespans of HLP-EST and HLP-OLS (top), and HEFT and HLP-OLS (bottom) for each configuration, grouped by application. Notice that the big square shows the mean, for each application.



(a) Ratio of makespan over LP^* for each configuration, grouped by application for the algorithms generalised for 3 resource types. Notice that the big square shows the mean, for each application.



(b) Ratio between the makespan of QHEFT and QHLP-OLS for each configuration, grouped by application. Notice that the big square shows the mean, for each application.

Figure 6.5

outliers for the applications *getrf*, *posv* and *potrf*, the algorithm outperforms on average QHLP-OLS.

Fig. 6.5b (bottom) compares in more detail QHEFT and QHLP-OLS, by showing the ratio between the makespans of the two algorithms. Comparing QHLP-OLS and QHEFT, we can see that QHEFT presents an improvement over QHLP-OLS of 5% on average. We also observe that the ratios for the most irregular application, *fork-join*, are spread with configurations favorable to QHEFT (up to 45% of improvement) and other configurations favorable to QHLP-OLS (up to 36% of improvement). Similar results were observed between QHLP-EST and QHLP-OLS as for 2 resource types and thus are not showed here.

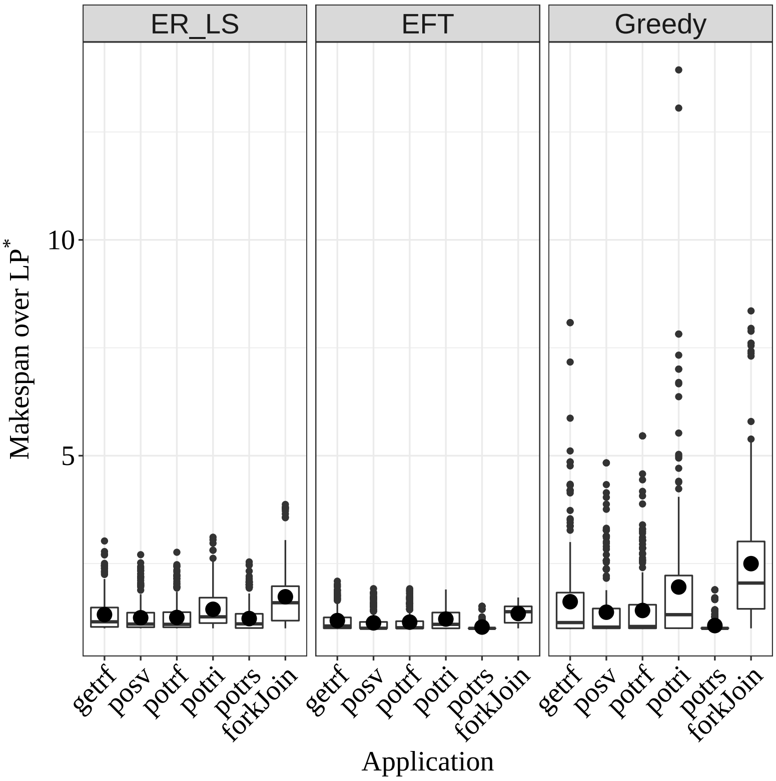
Finally, we notice that the approximation ratios, computed with a lower bound of the optimal makespan, do not exceed 2 and thus are far from the theoretical bounds of the algorithms, even for the case with 3 resource types.

6.7.2 On-line Setting

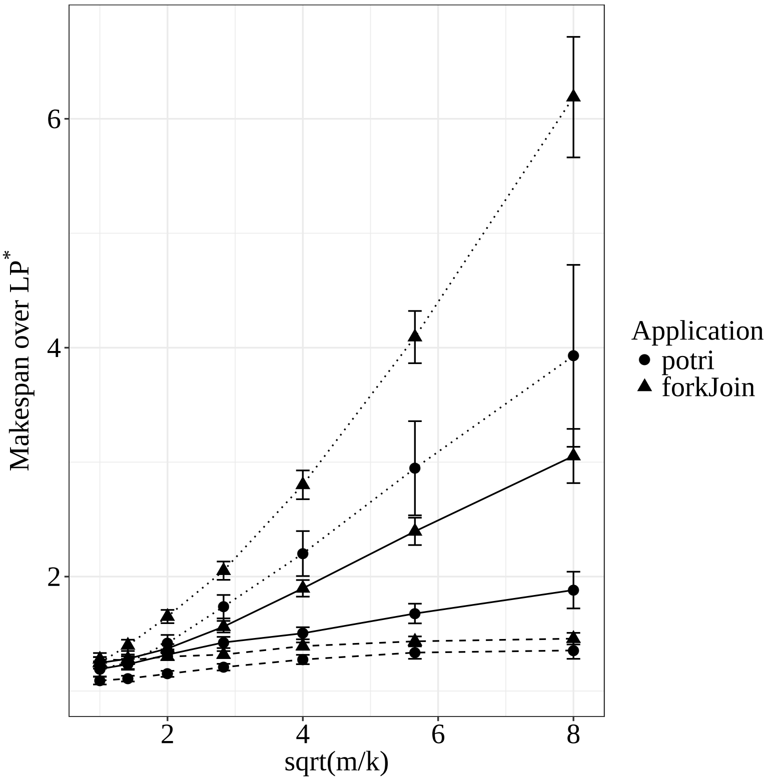
Algorithms.
We compared the performance, in terms of makespan, of the algorithm ER-LS (Section 6.4.2) with 3 baseline algorithms: *EFT*, which schedules a task on the processor which gives the earliest finish time for that task; *Greedy*, which allocates a task on the processor type which has the smallest processing time for that task; and *Random*, which randomly assigns a task to the CPU or GPU side. For the algorithms Greedy and Random, we used a List Scheduling algorithm to schedule the tasks once the allocations have been made. The algorithms were implemented in Python (v. 2.8.6). For the machine settings, we used the same sets of pairs as for the off-line setting with 16, 32, 64 and 128 CPUs, and 2, 4, 8 and 16 GPUs.

We executed the algorithms only once with each configuration since all algorithms are deterministic, except Random. The running times of the algorithms were similar and took at most 5 seconds for the application instance giving the highest number of tasks.

Results.
Fig. 6.6a compares the ratios between the makespan of each of the on-line algorithms and LP^*. Due to large differences between the performances of Random and the 3 other algorithms, we kept only the algorithms ER-LS, EFT and Greedy. Results show that Greedy is on average outperformed by ER-LS and EFT, and that EFT creates



(a) Ratio of makespan over LP^* for each configuration, grouped by application for the on-line algorithms with 2 resource types. Notice that the big square shows the mean, for each application.



(b) Mean competitive ratio of ER-LS (plain), EFT (dashed) and Greedy (dotted) as a function of $\sqrt{\frac{m}{k}}$.
Notice that error bars show the standard error.

Figure 6.6

less outliers than the 2 other algorithms. Fig. 6.7 compares in more detail Greedy and ER-LS (top), and EFT and ER-LS (bottom), by showing the ratio between the makespans of the two algorithms. We can see that ER-LS outperforms Greedy on average, with a maximum for the potri application where ER-LS performs 11 times better than Greedy. More specifically, there is an improvement of between 8% and 36% on average for ER-LS depending on the application considered, except for *potrs* whose makespans are on average 10% greater than for Greedy.

Comparing EFT and ER-LS, we can see that ER-LS is outperformed by EFT with a decrease of 11% on average, and up to 60% for certain configurations of fork-join. However, the worst-case competitive ratio for EFT can be directly obtained from the proof of the worst-case approximation ratio for HEFT, presented in Section 6.3. More specifically, if the adversary presents to EFT the list of tasks ordered as in the counter-example for HEFT (Theorem 1), i.e., by decreasing order of the rank, then we obtain the same lower bound.

We also study the performance of the 3 algorithms with respect to the theoretical upper bound given in Section 6.4.2. Fig. 6.6b shows the mean competitive ratio of ER-LS, EFT and Greedy along with the standard error as a function of $\sqrt{\frac{m}{k}}$ associated to each configuration. To simplify the lecture, we only present the applications *potri* and *fork-join*, since other *Chameleon* applications showed similar results. We observe that the competitive ratio is smaller than $\sqrt{\frac{m}{k}}$ and far from the theoretical upper bound of $4\sqrt{\frac{m}{k}}$ for ER-LS.

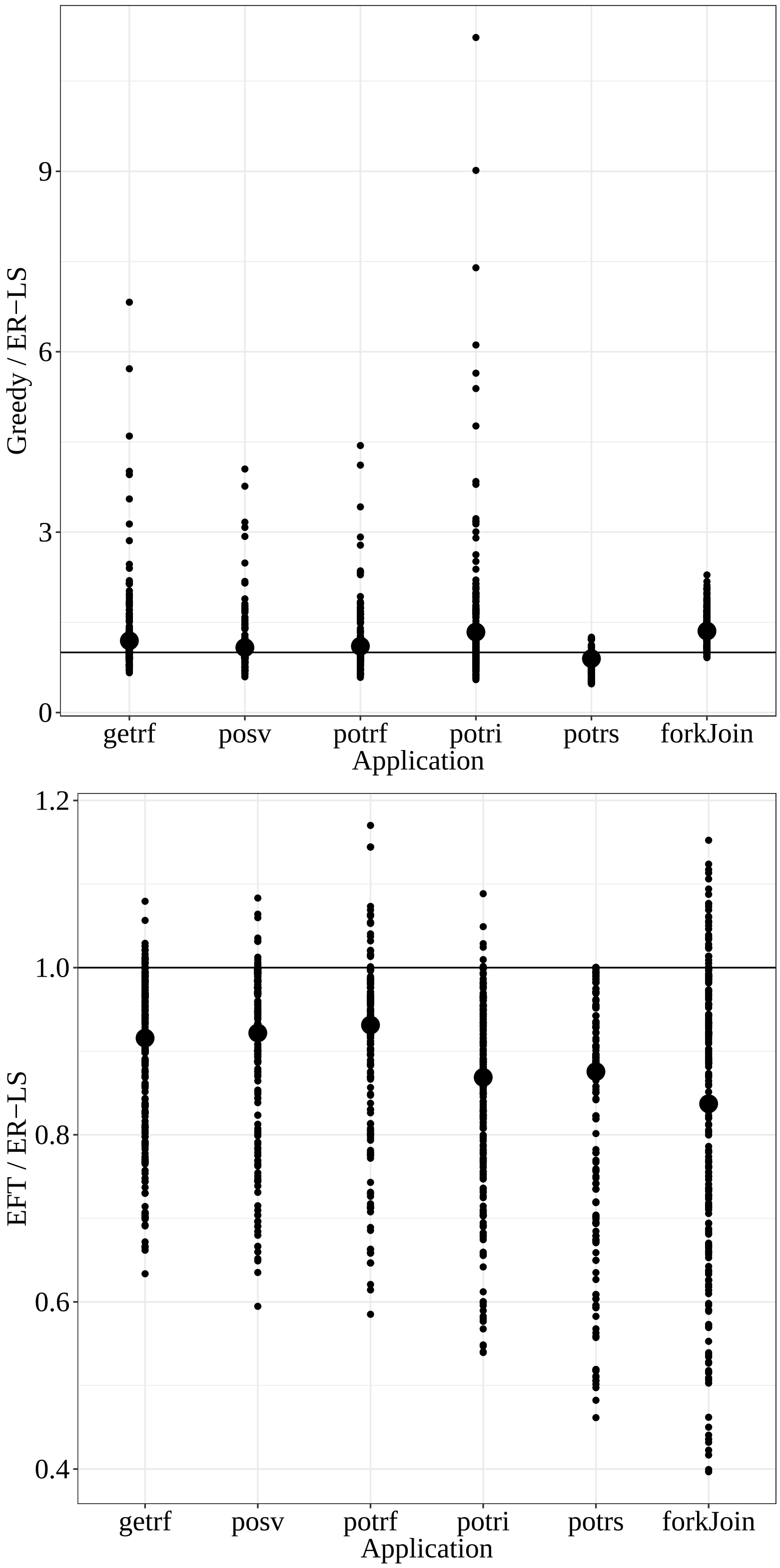


Figure 6.7: Ratio between the makespans of Greedy and ER-LS (top), and EFT and ER-LS (bottom) for each configuration, grouped by application. Notice that the big square shows the mean, for each application.

7 Semi On-Line Two-Agent Scheduling

In this chapter, we propose the formulation and modelling of a particular setting of a two-agent non-preemptive scheduling problem on parallel machines. Since we are still in the early stages in the study of this problem, we will only present the model and related work.

This is an ongoing work in collaboration with Vincent Fagnon, Giorgio Lucarelli and Denis Trystram.

7.1 Context

As opposed to the placement problem discussed in Chapter 3 dealing with a whole distributed infrastructure, we place ourselves in a more *local* environment within an Edge site. In this context, the target platform is a parallel machine composed of identical processors dedicated to service the computing requests originating from the smart building and its vicinity. These computing requests are submitted to the system one by one and should be executed as soon as possible. In addition, when there are no local requests to service, the computing resources are made available for processing *off-loaded* tasks coming from neighbouring Edge sites or Cloud data-centres. Such off-loaded tasks are submitted to the platform by a *superior entity* and are executed in a best effort manner until new requests from the smart building arrive. The superior entity is in charge of managing multiple Edge sites and ensuring a global load balancing of the off-loaded tasks between these sites.
Figure 7.1 summarises this context.

7.2 Related Work

Our problem can be viewed as an extension of the non-preemptive two-agent scheduling problem on a single machine, which has been extensively studied in the

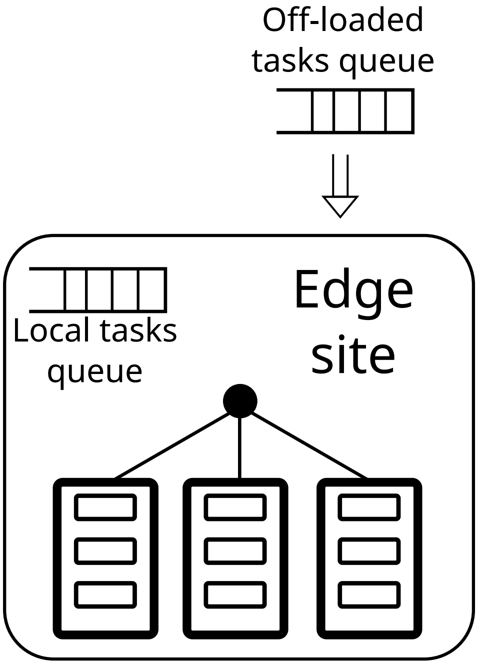


Figure 7.1: Scheme of an Edge site with two tasks queues corresponding to two agents competing for the computing resources.

literature. However, the solutions presented in the following are not suited for our problem as, in our case, the tasks of one agent are submitted *on-line* with release dates.

Agnetis et al. [Agn+04] consider the problem where each agent has its own objective function to minimise, such as the sum of a regular function, the (weighted) sum of tasks completion time, or the sum of tardy tasks. They study multiple combinations of objective functions with two different methods to solve the problem. The first method is to transform the multi-objective into a single-objective problem, by transforming the objective of the second agent into a constraint. The second solution is to find the set of non-dominated pairs of objective values on the Pareto front. This work was further extended to the multi-agent scheduling problem on multiple machines [Agn+14].

Li et al. [LCF16] address a variant of the problem where each agent has a family of tasks to schedule. When the first task of a family is scheduled, or when a task is scheduled after the task of another family, a setup time is appended prior to the starting time of the task. The authors transform the second objective into a constraint and propose polynomial- and pseudo-polynomial-time algorithms to solve the problem with multiple objective functions, such as the sum of completion times, the maximum lateness or the number of tardy tasks.

Cheng et al. [Che+19] study the problem with due dates, where the first agent wants to minimise the sum of lateness while the second agent wants to minimise the number of tardy tasks, and propose a method to find the non-dominated solutions. This problem is further studied by Yin et al. [YWC20], considering more objective functions for the two agents.

Baker and Smith [BS03] study the problem with two or three agents competing for the single resource, and the overall optimisation objective is a weighted combination of the objective of each agent, considering the maximum completion time, the sum of (weighted) completion time or the maximum lateness of the tasks.
On a similar way, Liu et al. [LGL19] study the off-line problem with release dates, where the objective is a linear combination of the maximum tasks completion time of each agent. They propose an algorithm whose approximation ratio depends on the linear coefficients in the objective function, and a fully polynomial time approximation scheme, to solve the considered problem.

One can also find extensions of the scheduling problem with more than two agents and more than one machine. For example, Saule and Trystram [ST09] focus on the problem with arbitrary number of agents competing to schedule tasks on a parallel machine, where the objective of an agent is either the minimisation of the maximum, or the sum, of completion time of their tasks. They propose inapproximability bounds and give algorithms with approximation ratio depending on the number of agents.

Another scheduling problem of interest is the problem of multi-organisation scheduling [Coh+11; Dut+11], where each organisation has its own computing resources and its own set of tasks to execute. The objective is to minimise the overall completion time of all organisations with the ability to off-load tasks from one organisation to another. However, this corresponds to a broader view of our addressed problem that considers multiple Edge sites at once.

7.3 Problem Modelling

We consider in this work a particular setting of the problem of two-agent non-preemptive scheduling, and model the problem as follows. The target platform is a parallel machine composed of m identical processors. Two agents compete for the resources of the platform to execute their own set of non-preemptive sequential tasks and optimise their own objective.

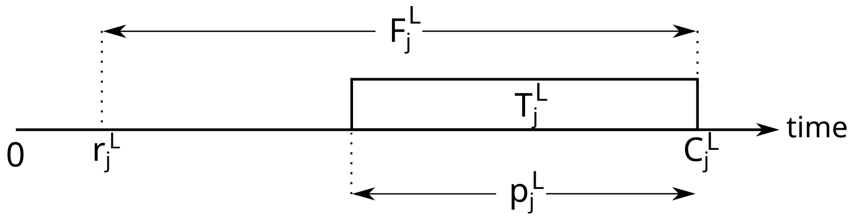


Figure 7.2: Summary of notations for the execution of a local task.

The first agent corresponds to the grouping of all users submitting computing requests from the smart building. We denote by $\mathcal{S}_\mathcal{L}$ the set of all n_L **local tasks** T_j^L, with $j \in [1, n_L]$. Local tasks are submitted in an **on-line** fashion with release dates, meaning that a task T_j^L and its processing time p_j^L becomes known to the system only at its release time r_j^L. Given a schedule, we denote by C_j^L the completion time of a local task and by $F_j^L = C_j^L - r_j^L$ its **flow-time**, which corresponds to the time spent by the task in the system. Figure 7.2 summarises all these notations.

Similarly, the tasks from the second agent correspond to the tasks off-loaded to this platform by the superior entity. We denote by $\mathcal{S}_\mathcal{G}$ the set of all n_G **global tasks** T_j^G, with $j \in [1, n_G]$. These global tasks are submitted **off-line** as a batch, all ready at time 0 and with known processing time p_j^G, $\forall j \in [1, n_G]$. We also denote by C_j^G the completion time of a global task in a given schedule.

The addressed problem is bi-objective and each agent has its own objective to optimise. The objective of the local agent is the minimisation of the sum flow-time of its tasks, denoted by $\sum F^L = \sum_{j \in [1, n_L]} F_j^L$, while the objective of the global agent is the minimisation of the maximum completion time of its tasks, denoted by $C_{max}^G = \max_{j \in [1, n_G]} C_j^G$.

7.4 Resource Augmentation

Resource augmentation is a well-known technique to help break theoretical lower bounds on the competitive ratio of scheduling algorithms in the on-line setting by giving more power to the algorithm when comparing to the optimal.

Different resource augmentation techniques have been introduced in the past years, from *machine augmentation* [Phi+97] – the algorithm is given more machines to schedule the tasks – and *speed augmentation* [KP00] – the algorithm is given machines that execute tasks faster – to *task rejection*[Cho+15] – the algorithm may decide to reject some tasks and not execute them at all.

A lower bound on the competitive ratio of an algorithm reveals its weaknesses by presenting a worst-case, or *pathological*, instance for which the algorithm fails to produce a good solution compared to the optimal. In such a case, simply endowing the algorithm a little bit more power could help cope with the problematic instance and reduce drastically the gap between the algorithm solution and the optimal one, reducing the competitive ratio from an unbounded value depending on some parameters of the instance input to a constant performance guarantee. For example, Lucarelli et al. applied different resource augmentation techniques to the online problem of scheduling non-preemptive tasks on unrelated machines with the objective of minimising the total weighted flow-time of the tasks [Luc+16]. Despite the strong lower bound of $\Omega(\sqrt{n})$ for the simplified off-line problem on a single machine, where n is the number of tasks, they showed that a combination of speed augmentation and rejection permitted to design an algorithm with a competitive ratio depending on the speed increase and the rejection allowed to the algorithm.

In our context, global tasks are off-loaded to the platform to take benefit from the idle times when there are no local tasks to execute. At times, it may happen that there is a burst of local task submissions but there are not enough available machines due to the execution of global tasks. In fact, it can be shown in an example, with a long global task and a short local task to be executed on a single machine, that any scheduling decision could lead to arbitrarily bad objective value for the local task. It simply consists in submitting the local task right after the time when the algorithm decided the global task to start. In this case, the achieved competitive ratio for the local task is the ratio of processing time between the global and the local tasks, which can be arbitrarily large.

For this reason, to reduce the impact of scheduling global tasks over local ones, we introduce in our model the rejection of tasks, which was already studied in the off-line problem of two-agent scheduling on a single machine by Feng et al. [Fen+14], where the rejection of a task induces a penalty on the objective function of the corresponding agent. However, we restrict in our case the algorithm to reject only an ϵ-fraction of the global tasks, in terms of number of tasks, and thus do not consider any penalty on the objective function.

The reason why we allow the rejection of only the global tasks, and not the local ones, is to comply with our application of the problem in practice, where global tasks are off-loaded by the *superior entity* and executed in a best effort manner, while local tasks originates from the local environment and must be executed on place. Moreover, when a global task is rejected, it is sent back to the superior entity that will re-submit this task in a future batch of global task off-loaded to another Edge site.

Notice that the rejection of tasks should not be mixed up with the notion of preemption. The preemption of a task consists in stopping its execution on a processor for a given time and resume the execution afterwards, to leave space and execute a task of higher priority for example. In our addressed problem, the preemption of tasks is not allowed, as it requires time and memory to save the state of tasks, which is costly and even may not be feasible in practice.

8 Conclusion

Through the different chapters of this dissertation, we studied both theoretical and practical aspects of resource management problems in heterogeneous platforms.

First, we discussed challenges raised by the evolution of Edge Computing platforms and took a focus on the production platform of the *Qarnot Computing* company as a case study. The implementation of new features in Batsim, a simulator originally designed for High Performance Computing infrastructures, allowed us to perform a complete simulation of the Qarnot platform. Although not fully validated, our simulated platform enabled us to quickly design and test variants of data and job placement policies, corresponding to the scheduling problem at the top-level of the Qarnot platform. We hope that such an example of how to leverage the Batsim/SimGrid toolkit to simulate a complex Edge Computing platform and its internal mechanisms will encourage researchers to move away from the development of *ad hoc* simulators for the purpose of a single experiment/publication, and widely adopt a common simulation toolkit like Batsim.

Second, at the border between theory and practice, we introduced a model to predict the temperature of heaters and their ambient air in rooms and offices. We designed a prediction method based on thermodynamics formulae and compared it with a simpler prediction method widely used in the literature, and a machine learning approach using a simplistic neural network. Using real logs from the heaters of the Qarnot platform, we observed very low accuracy for the two analytical prediction methods compared to the machine learning-based method. We are currently working on retrieving a new set of data and refining our prediction method to try and improve the results.

Third, we theoretically studied the problem of scheduling a parallel application on hybrid multi-core machines. We detailed the problem and several setting variants and exposed, for each variant, the current algorithms achieving the best known performance guarantees and their lower bounds. We also proposed two new algorithms and gave proofs of their performance guarantees. Then, we circled back in the practical aspect of the work by comparing the performances between our scheduling algorithms and baselines with applications derived from linear algebra kernels.
Taking a step further, one great improvement for this work would be to introduce in

the model a communication cost between two tasks of the application when both tasks are executed on different resource types. One can find in the literature some work on that matter considering a hybrid machine with CPUs and GPUs [AZM18; AMP19]. However, a stronger model introducing also communication costs between two GPUs should be considered, to bring the theory closer to what is happening in practice.

Finally, on a more open perspective, we proposed the formulation of a new setting of the two-agent scheduling problem on parallel machines. This problem corresponds to the resource management problem of an Edge site and is related to the bottom-level scheduling problem of the Qarnot platform. We decided to include this formulation in the manuscript, even if it is still an on-going work without results yet, because it completes our tour of the Qarnot use-case by addressing another resource management problem at the bottom-level of the platform.
Furthermore, Chapter 7 embodies a major step in any theoretical analysis, which is the modelling and formulation of the optimisation problem under investigation. Such a step is difficult and it may take several iterations to refine the model from a real-world problem to a sound formulation that can be analysed theoretically. Thus, the reader should not be mistaken by the rather small size of the chapter, as it summarises several months of thinking and modelling choices to shape up the proposed formulation. Yet, the study of this scheduling problem, and all of its possible variants with different machine settings and objectives, is left as an open door, as there is enough space for a whole thesis.

On a personal note, all this work around a production platform and its simulation greatly improved my practical background and helped me better grasp the links between theoretical and practical problems. I believe that an effort should be made to reduce the gap between theoretical optimisation problems and their application in practice. In fact, linking the theoretical, possibly simplified, aspect of a problem to its practical counterpart gives *meaning* to the solution produced, as opposed to solutions using simple heuristics or coming from pure *black-box* techniques such as with deep learning. This is why we chose to develop a new temperature prediction method based on thermodynamics formulae and use it in our simulations, instead of relying on classical machine learning solutions. Unfortunately, our experiments showed that it was not that simple.

Yet, we saw that it was sometimes difficult to have a good theoretical analysis of optimisation problems. This brings us to ask ourselves what will be the future of scheduling and more generally resource management problems. As the next generation of computing platforms will continue to grow in complexity, the related optimisation problems may require even more assumptions and simplifications to have theoretical analysis proving constant approximation guarantees. One solution would be to break down and tackle smaller, more localised, problems that could be combined together to bring solutions to the whole picture. This is what we intended to do with the Qarnot use-case by distinguishing the *global* and *local* placement problems to replace the actual *all-in-one* solution. However, such solutions would have to compete with more simple – low-complexity – heuristics that are usually preferred in practice, even if they do not provide any performance guarantees.

Furthermore, one can find in the literature many works based on machine learning techniques, such as a methodology to derive scheduling policies for High Performance Computing platforms [CD17] or temperature prediction methods, as we saw in Chapter 4. With the constant progress of neural networks and data analytics, one can simply wonder *“will these techniques become the new standard for solving optimisation problems?”*

Bibliography

[@batgit] *Batsim Github Repository*. URL: `https://github.com/oar-team/batsim` (visited on Sept. 25, 2020). cit. on pp. 27, 28

[@batrust] *Batsim-rust Gitlab Repository*. URL: `https://gitlab.inria.fr/batsim/batsim-rust` (visited on Sept. 25, 2020). cit. on p. 27

[@batsched] *Batsched Gitlab Repository*. URL: `https://gitlab.inria.fr/batsim/batsched` (visited on Sept. 25, 2020). cit. on p. 27

[@battemp] *Batsim Temperature Branch*. URL: `https://gitlab.inria.fr/batsim/batsim/tree/temperature-sbac-2020` (visited on Sept. 25, 2020). cit. on p. 31

[@bench] *Heterogeneous SWF GitHub Repository*. URL: `https://github.com/marcosamaris/heterogeneous-SWF` (visited on Sept. 25, 2020). cit. on p. 97

[@cisco] *Cisco Annual Internet Report (2018-2023) White Paper*. URL: `https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html` (visited on Sept. 25, 2020). cit. on p. 2

[@cloudheat] *Cloud&Heat Website*. URL: `https://www.cloudandheat.com` (visited on Sept. 25, 2020). cit. on pp. 3, 20

[@CSU] *Computer Science Unplugged*. URL: `https://csunplugged.org/en/` (visited on Sept. 25, 2020). cit. on p. 6

[@digiplex] *DigiPlex Website*. (Visited on Sept. 25, 2020). cit. on p. 3

[@evalys] *Evalys Github Repository*. URL: `https://github.com/oar-team/evalys` (visited on Sept. 25, 2020). cit. on p. 27

[@lancey] *Lancey Website*. URL: `https://www.lancey.fr/en/` (visited on Sept. 25, 2020). cit. on p. 45

[@omnet] *OMNet++ Github Repository*. URL: `https://github.com/omnetpp/omnetpp` (visited on Sept. 25, 2020). cit. on p. 25

[@pybatsim] *Pybatsim Gitlab Repository*. URL: `https://gitlab.inria.fr/batsim/pybatsim` (visited on Sept. 25, 2020). cit. on pp. 27, 28, 35

[@pytemp] *Pybatsim Temperature Branch*. URL: `https://gitlab.inria.fr/batsim/pybatsim/tree/temperature-sbac-2020` (visited on Sept. 25, 2020). cit. on p. 36

[@qarnot] *Qarnot Computing Website*. URL: `https://www.qarnot.com/en/home` (visited on Sept. 25, 2020). cit. on pp. 3, 4

[@robin] *Batexpe Gitlab Repository*. URL: `https://gitlab.inria.fr/batsim/batexpe` (visited on Sept. 25, 2020). cit. on p. 27

[@SGgit] *SimGrid Github Repository*. URL: `https://github.com/simgrid/simgrid` (visited on Sept. 25, 2020). cit. on p. 26

[@SGpublis] *SimGrid Publications*. URL: `http://simgrid.org/publications.html` (visited on Sept. 25, 2020). cit. on pp. 23, 26

[@SGtemp] *SimGrid Temperature Branch*. URL: `https://github.com/Mommessc/simgrid/tree/temperature-sbac-2020` (visited on Sept. 25, 2020). cit. on p. 31

[@simpleIoTsim] *SimpleIoTSimulator Web Site*. URL: `http://www.smplsft.com/SimpleIoTSimulator.html` (visited on Sept. 25, 2020). cit. on p. 24

[@top500] *TOP500 Website*. URL: `https://www.top500.org/` (visited on Sept. 25, 2020). cit. on p. 1

[AA16] Arif Ahmed and Ejaz Ahmed. "A Survey on Mobile Edge Computing". In: *2016 10th International Conference on Intelligent Systems and Control (ISCO)*. Jan. 2016, pp. 1–8. cit. on p. 3

[ADL19] Farah Ait Salaht, Frédéric Desprez, and Adrien Lebre. "An Overview of Service Placement Problem in Fog and Edge Computing". In: RR-9295 (Oct. 2019), pp. 1–43. cit. on p. 5

[Agn+04] Alessandro Agnetis, Pitu B. Mirchandani, Dario Pacciarelli, and Andrea Pacifici. "Scheduling Problems with Two Competing Agents". In: *Operations Research* 52.2 (2004), pp. 229–242. cit. on p. 108

[Agn+14] Alessandro Agnetis, Jean-Charles Billaut, Stanislaw Gawiejnowicz, Dario Pacciarelli, and Ameur Soukhal. *Multiagent Scheduling - Models and Algorithms*. Springer, 2014. cit. on p. 108

[Agu+12] Emmanuel Agullo, George Bosilca, Bérenger Bramas, et al. "Poster: Matrices Over Runtime Systems at Exascale". In: *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*. Nov. 2012, pp. 1330–1331. cit. on p. 96

[AIM10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. "The Internet of Things: A survey". In: *Computer networks* 54.15 (2010), pp. 2787–2805. cit. on p. 3

[Ait+19] 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*. IEEE, July 2019, pp. 1–9. cit. on p. 5

[All+02] Bill Allcock, Joe Bester, John Bresnahan, et al. “Data Management and Transfer in High-Performance Computational Grid Environments”. In: *Parallel Computing* 28.5 (2002), pp. 749–771. cit. on p. 24

[Ama+16] Marcos Amaris, Raphael Y. de Camargo, Mohamed Dyab, Alfredo Goldman, and Denis Trystram. “A comparison of GPU execution time prediction using machine learning and analytical modeling”. In: *IEEE 15th International Symposium on Network Computing and Applications*. Oct. 2016, pp. 326–333. cit. on p. 67

[Ama+17] Marcos Amaris, Giorgio Lucarelli, Clément Mommessin, and Denis Trystram. “Generic Algorithms for Scheduling Applications on Hybrid Multicore Machines”. In: *Euro-Par: International Conference on Parallel and Distributed Computing*. 2017, pp. 220–231. cit. on p. 75

[Ama+18] Marcos Amaris, Giorgio Lucarelli, Clément Mommessin, and Denis Trystram. “Generic Algorithms for Scheduling Applications on Heterogeneous Platforms”. In: *Concurrency and Computation: Practice and Experience* (July 2018), pp. 1–29. cit. on pp. 73, 75

[AMP19] Massinissa Ait Aba, Alix Munier-Kordon, and Guillaume Pallez. “Scheduling on Two Unbounded Resources with Communication Costs”. In: *Euro-Par - European Conference on Parallel Processing*. Gottingen, Germany, Aug. 2019. cit. on p. 114

[Aug+11] Cédric Augonnet, Samuel Thibault, Raymond Namyst, and Pierre-André Wacrenier. “StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures”. In: *Concurrency and Computation: Practice and Experience* 23.2 (2011), pp. 187–198. cit. on pp. 69, 71, 96

[AZM18] Massinissa Ait Aba, Lilia Zaourar, and Alix Munier-Kordon. “Approximation algorithm for scheduling applications on hybrid multi-core machines with communications delays”. In: *IPDPS Workshops 2018, 2018 IEEE International Parallel and Distributed Processing Symposium Workshops*. Vancouver, Canada: IEEE, May 2018, pp. 36–45. cit. on p. 114

[Bau+20a] Adwait Bauskar, Anderson da Silva, Adrien Lebre, et al. “Investigating Placement Challenges in Edge Infrastructures through a Common Simulator”. In: *SBAC-PAD* (2020). cit. on p. 24

[Bau+20b] Adwait Bauskar, Anderson da Silva, Adrien Lebre, et al. “Investigating Placement Challenges in Edge Infrastructures through a Common Simulator”. HAL Technical Report. Feb. 2020. cit. on p. 24

[Bea+20] Olivier Beaumont, Louis-Claude Canon, Lionel Eyraud-Dubois, et al. “Scheduling on Two Types of Resources: A Survey”. In: *ACM Comput. Surv.* 53.3 (May 2020). cit. on pp. 67, 73

[BEK17] Olivier Beaumont, Lionel Eyraud-Dubois, and Suraj Kumar. “Approximation Proofs of a Fast and Efficient List Scheduling Algorithm for Task-Based Runtime Systems on Multicores and GPUs”. In: *IEEE International Parallel & Distributed Processing Symposium (IPDPS)*. Orlando, United States, May 2017. cit. on pp. 71, 72

[BF17] Antonio Brogi and Stefano Forti. “QoS-aware Deployment of IoT Applications Through the Fog”. In: *IEEE Internet of Things Journal* 4.5 (Oct. 2017), pp. 1185–1192. cit. on pp. 5, 24

[Ble+15] Raphael Bleuse, Safia Kedad-Sidhoum, Florence Monna, Grégory Mounié, and Denis Trystram. “Scheduling independent tasks on multi-cores with GPU accelerators”. In: *Concurrency and Computation: Practice and Experience* 27.6 (2015), pp. 1625–1638. cit. on pp. 71, 78

[Bos+13] George Bosilca, Aurélien Bouteiller, Anthony Danalis, et al. “PaRSEC: A programming paradigm exploiting heterogeneity for enhancing scalability”. In: *Computing in Science and Engineering* 15.6 (Nov. 2013), pp. 36–45. cit. on p. 69

[Bra+01] Tracy D. Braun, Howard Jay Siegel, Noah Beck, et al. “A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems”. In: *Journal of Parallel and Distributed Computing* 61.6 (2001), pp. 810–837. cit. on p. 71

[BS03] Kenneth R. Baker and J. Cole Smith. “A Multiple-Criterion Model for Machine Scheduling”. In: *J. Scheduling* 6.1 (2003), pp. 7–16. cit. on p. 109

[BW12] Vincenzo Bonifaci and Andreas Wiese. “Scheduling unrelated machines of few different types”. In: *arXiv preprint arXiv:1205.0974* (2012). cit. on p. 71

[Cal+11] Rodrigo Calheiros, R. Ranjan, Anton Beloglazov, Cesar De Rose, and Rajkumar Buyya. “CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms”. In: *Software Practice and Experience* 41 (Jan. 2011), pp. 23–50. cit. on p. 25

[Can+19] Louis-Claude Canon, Loris Marchal, Bertrand Simon, and Frédéric Vivien. “Online Scheduling of Task Graphs on Heterogeneous Platforms”. In: *IEEE Transactions on Parallel and Distributed Systems* (2019). cit. on pp. 72, 73, 91

[Cas+14] Henri Casanova, Arnaud Giersch, Arnaud Legrand, Martin Quinson, and Frédéric Suter. “Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms”. In: *Journal of Parallel and Distributed Computing* 74.10 (June 2014), pp. 2899–2917. cit. on pp. 6, 23, 25

[CD17] Danilo Carastan-Santos and Raphael Y. De Camargo. “Obtaining Dynamic Scheduling Policies with Simulation and Machine Learning”. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC ’17. Denver, Colorado: Association for Computing Machinery, 2017. cit. on p. 115

[Che+19] Chen-Yang Cheng, Shu-Fen Li, Kuo-Ching Ying, and Yu-Hsi Liu. “Scheduling Jobs of Two Competing Agents on a Single Machine”. In: *IEEE Access* 7 (2019). cit. on p. 109

[Cho+07] Jeonghwan Choi, Youngjae Kim, Anand Sivasubramaniam, et al. “Modeling and Managing Thermal Profiles of Rack-mounted Servers with ThermoStat”. In: *2007 IEEE 13th International Symposium on High Performance Computer Architecture*. 2007, pp. 205–215. cit. on p. 44

[Cho+15] Anamitra Roy Choudhury, Syamantak Das, Naveen Garg, and Amit Kumar. “Rejecting jobs to Minimize Load and Maximum Flow-time”. In: *Proceedings of the 2015 Annual ACM-SIAM Symposium on Discrete Algorithms*. 2015, pp. 1114–1133. cit. on p. 110

[CMV17] Louis-Claude Canon, Loris Marchal, and Frédéric Vivien. “Low-Cost Approximation Algorithms for Scheduling Independent Tasks on Hybrid Platforms”. In: *Euro-Par 2017: 23rd International European Conference on Parallel and Distributed Computing*. Santiago de Compostela, Spain: Springer, Aug. 2017. cit. on pp. 71, 72

[Coh+11] Johanne Cohen, Daniel Cordeiro, Denis Trystram, and Frédéric Wagner. “Multi-Organization Scheduling Approximation Algorithms”. In: *Concurrency and Computation: Practice and Experience* 23.17 (2011), pp. 2220–2234. cit. on p. 109

[Cor+10] Daniel Cordeiro, Grégory Mounié, Swan Perarnau, et al. “Random Graph Generation for Scheduling Simulations”. In: *ICST (SIMUTools)*. 2010. cit. on p. 96

[CS15] Nathanael Cheriere and Erik Saule. “Considerations on Distributed Load Balancing for Fully Heterogeneous Machines: Two Particular Cases”. In: *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*. May 2015, pp. 6–16. cit. on p. 72

[CYZ14] Lin Chen, Deshi Ye, and Guochuan Zhang. “Online Scheduling Of Mixed CPU-GPU Jobs”. In: *International Journal of Foundations of Computer Science* 25.06 (2014), pp. 745–761. cit. on pp. 72, 73, 88

[Dav04] David P. Anderson. “BOINC: a System for Public-Resource Computing and Storage”. In: *Fifth IEEE/ACM International Workshop on Grid Computing*. 2004, pp. 4–10. cit. on pp. 13, 24

[Deg+17] Augustin Degomme, Arnaud Legrand, Georges Markomanolis, et al. “Simulating MPI applications: the SMPI approach”. In: *IEEE Transactions on Parallel and Distributed Systems* 28.8 (Aug. 2017), p. 14. cit. on p. 23

[Dem17] Wolfgang Demtröder. “Thermodynamics”. In: *Mechanics and Thermodynamics*. Cham: Springer International Publishing, 2017, pp. 253–319. cit. on p. 46

[DNC17] Amaury Durand, Yanik Ngoko, and Christophe Cérin. “Distributed and In-Situ Machine Learning for Smart-Homes and Buildings: Application to Alarm Sounds Detection”. In: *2017 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPS Workshops 2017*. 2017, pp. 429–432. cit. on p. 16

[Don+19] Bruno Donassolo, Ilhem Fajjari, Arnaud Legrand, and Panayotis Mertikopoulos. “Fog Based Framework for IoT Service Provisioning”. In: *IEEE CCNC*. Jan. 2019. cit. on p. 5

[Dro09] Maciej Drozdowski. *Scheduling for Parallel Processing*. Springer Publishing Company, 2009. cit. on pp. 2, 68

[Dut+11] Pierre-François Dutot, Fanny Pascual, Krzysztof Rzadca, and Denis Trystram. “Approximation Algorithms for the Multiorganization Scheduling Problem”. In: *IEEE Transactions on Parallel and Distributed Systems* 22.11 (2011), pp. 1888–1895. cit. on p. 109

[Dut+16] 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*. May 2016. cit. on pp. 6, 23, 25, 26

[Fag+19] Vincent Fagnon, Imed Kacem, Giorgio Lucarelli, and Bertrand Simon. *Scheduling on Hybrid Platforms: Improved Approximability Window*. 2019. arXiv: `1912.03088 [cs.DS]`. cit. on pp. 72, 73

[Fen+14] Qi Feng, Baoqiang Fan, Shisheng Li, and Weiping Shang. “Two-agent scheduling with rejection on a single machine”. In: *Applied Mathematical Modelling* 39.3 (2014), pp. 1183–1193. cit. on p. 111

[FMO07] Alexandre P. Ferreira, Daniel Mossé, and Jan C. Oh. “Thermal Faults Modeling Using a RC Model with an Application to Web Farms”. In: *19th Euromicro Conference on Real-Time Systems (ECRTS’07)*. 2007, pp. 113–124. cit. on p. 45

[Geh+16] Jan Clemens Gehrke, Klaus Jansen, Stefan EJ Kraft, and Jakob Schikowski. “A PTAS for Scheduling Unrelated Machines of Few Different Types”. In: *SOSFEM: Theory and Practice of Computer Science*. Springer, 2016, pp. 290–301. cit. on p. 71

[Get+15] Vladimir Getov, Darren J. Kerbyson, Matt Macduff, and Adolfy Hoisie. “Towards an Application-Specific Thermal Energy Model of Current Processors”. In: *Proceedings of the 3rd International Workshop on Energy Efficient Supercomputing*. E2SC ’15. Austin, Texas: Association for Computing Machinery, 2015. cit. on p. 45

[GJ90] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990. cit. on p. 70

[GMW07] Martin Gairing, Burkhard Monien, and Andreas Woclaw. "A faster combinatorial approximation algorithm for scheduling unrelated parallel machines". In: *Theoretical Computer Science* 380.1 (2007), pp. 87–99. cit. on p. 71

[Gra+79] Ronald L. Graham, Eugene L. Lawler, Jan K. Lenstra, and Alexander H. G. Rinnooy Kan. "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey". In: *Discrete Optimization II*. Vol. 5. Annals of Discrete Mathematics. Elsevier, 1979, pp. 287–326. cit. on p. 70

[Gra69] Ronald. L. Graham. "Bounds on Multiprocessing Timing Anomalies". In: *SIAM Journal On Applied Mathematics* 17.2 (1969), pp. 416–429. cit. on pp. 81, 94

[Gup+16] 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". In: *Software: Practice and Experience* (June 2016). cit. on p. 25

[GV13] Gene H. Golub and Charles F. Van Loan. "Matrix Computations, 4th". In: 2013. cit. on p. 52

[Han+14] Son N. Han, Gyu M. Lee, Noel Crespi, et al. "DPWSim: A Simulation Toolkit for IoT Applications Using Devices Profile for Web Services". In: *2014 IEEE World Forum on Internet of Things (WF-IoT)*. 2014, pp. 544–547. cit. on p. 24

[Hea+06] Taliver Heath, Ana Paula Centeno, Pradeep George, et al. "Mercury and Freon: Temperature Emulation and Management for Server Systems". In: *SIGPLAN Not.* 41.11 (Oct. 2006), pp. 106–116. cit. on p. 44

[Hei+17] Franz C. Heinrich, Tom Cornebize, Augustin Degomme, et al. "Predicting the Energy Consumption of MPI Applications at Scale Using a Single Node". In: *Cluster 2017*. IEEE. Sept. 2017. cit. on pp. 27, 34

[Hei19] Franz Christian Heinrich. "Modeling, Prediction and Optimization of Energy Consumption of MPI Applications Using SimGrid". PhD thesis. Univ. Grenoble Alpes, France, 2019. cit. on p. 25

[HRL18] Petri Hietaharju, Mika Ruusunen, and Kauko Leiviskä. "A Dynamic Model for Indoor Temperature Prediction in Buildings". In: *Energies* 11 (June 2018), p. 1477. cit. on p. 45

[HS87] Dorit S. Hochbaum and David B. Shmoys. "Using Dual Approximation Algorithms for Scheduling Problems Theoretical and Practical Results". In: *J. ACM* 34.1 (Jan. 1987), pp. 144–162. cit. on pp. 70, 72

[HS89] Leslie A. Hall and David B. Shmoys. “Approximation schemes for constrained scheduling problems”. In: *30th Annual Symposium on Foundations of Computer Science*. Oct. 1989, pp. 134–139. cit. on p. 70

[Hua+06] Wei Huang, Shougata Ghosh, Sivakumar Velusamy, et al. “HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design”. In: *IEEE Transactions on Very Large Scale Integration Systems - VLSI* 14 (May 2006), pp. 501–513. cit. on p. 45

[JM08] Ramkumar Jayaseelan and Tulika Mitra. “Temperature Aware Task Sequencing and Voltage Scaling”. In: *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*. ICCAD ’08. San Jose, California: IEEE Press, 2008, pp. 618–623. cit. on p. 45

[Ked+18] Safia Kedad-Sidhoum, Florence Monna, Grégory Mounié, and Denis Trystram. “A Family of Scheduling Algorithms for Hybrid Parallel Platforms”. In: *International Journal of Foundations of Computer Science* 29.1 (2018), pp. 63–90. cit. on p. 72

[KMT15] Safia Kedad-Sidhoum, Florence Monna, and Denis Trystram. “Scheduling tasks with precedence constraints on hybrid multi-core machines”. In: *HCW - IPDPS Workshops*. 2015, pp. 27–33. cit. on pp. 73, 76, 81, 82, 87, 94

[KN12] Rini T. Kaushik and Klara Nahrstedt. “T*: A Data-Centric Cooling Energy Costs Reduction Approach for Big Data Analytics Cloud”. In: *SC ’12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. 2012, pp. 1–11. cit. on p. 45

[KP00] Bala Kalyanasundaram and Kirk Pruhs. “Speed is as Powerful as Clairvoyance”. In: *J. ACM* 47.4 (July 2000), pp. 617–643. cit. on p. 110

[LCF16] Shi-Sheng Li, Ren-Xia Chen, and Qi Feng. “Scheduling Two Job Families on a Single Machine with Two Competitive Agents”. In: *J. Comb. Optim.* 32.3 (2016), pp. 784–799. cit. on p. 108

[Leb+19] Adrien Lebre, Jonathan Pastor, Anthony Simonet, and Mario Südholt. “Putting the Next 500 VM Placement Algorithms to the Acid Test: The Infrastructure Provider Viewpoint”. In: *IEEE Transactions on Parallel and Distributed Systems* 30.1 (Jan. 2019), pp. 204–217. cit. on p. 5

[Lee+10] Victor W. Lee, Changkyu Kim, Jatin Chhugani, et al. “Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU”. In: *SIGARCH Computer Architecture News* 38 (2010), pp. 451–460. cit. on p. 1

[Leu04] Joseph Y.T. Leung. *Handbook of scheduling: algorithms, models, and performance analysis*. CRC Press, 2004. cit. on p. 69

[LGL19] Peihai Liu, Manzhan Gu, and Ganggang Li. “Two-Agent Scheduling on a Single Machine with Release Dates”. In: *Computers and Operations Research* 111 (2019), pp. 35–42. cit. on p. 109

[LL78] Jane W. Liu and C. L. Liu. "Performance Analysis of Multiprocessor Systems Containing Functionally Dedicated Processors". In: *Acta Informatica* 10.1 (1978), pp. 95–104. cit. on p. 71

[LQ10] Shaobo Liu and Meikang Qiu. "Thermal-Aware Scheduling for Peak Temperature Reduction with Stochastic Workloads". In: *In 16th IEEE Real-Time and Embedded Technology and Applications Symposium*. 2010, pp. 59–62. cit. on p. 45

[LST90] Jan Karel Lenstra, David B Shmoys, and Éva Tardos. "Approximation algorithms for scheduling unrelated parallel machines". In: *Mathematical programming* 46.1-3 (1990), pp. 259–271. cit. on p. 71

[Luc+16] Giorgio Lucarelli, Nguyen Kim Thang, Abhinav Srivastav, and Denis Trystram. "Online Non-preemptive Scheduling in a Resource Augmentation Model based on Duality". In: *European Symposium on Algorithms (ESA 2016)*. Vol. 57. 63. Aarhus, Denmark, Aug. 2016, pp. 1–17. cit. on p. 111

[M60] *Meteo60 Website*. URL: `https://www.meteo60.fr` (visited on Sept. 25, 2020). cit. on p. 35

[Mas+20] Eric Masanet, Arman Shehabi, Nuoa Lei, Sarah Smith, and Jonathan Koomey. "Recalibrating Global Data Center Energy-Use Estimates". In: *Science* 367.6481 (2020), pp. 984–986. eprint: `https://science.sciencemag.org/content/367/6481/984.full.pdf`. cit. on p. 3

[Mij+15] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, et al. "Network Function Virtualization: State-of-the-art and Research Challenges". In: *IEEE Communications Surveys & Tutorials* 18.1 (2015), pp. 236–262. cit. on p. 3

[Moo+05] Justin Moore, Jeff Chase, Parthasarathy Ranganathan, and Ratnesh Sharma. "Making Scheduling "Cool": Temperature-Aware Workload Placement in Data Centers". In: *Proceedings of the Annual Conference on USENIX Annual Technical Conference*. ATEC '05. Anaheim, CA: USENIX Association, 2005, p. 5. cit. on p. 45

[Naa+17] Mohammed Islam Naas, Philippe Raipin Parvedy, Jalil Boukhobza, and Laurent Lemarchand. "iFogStor: An IoT Data Placement Strategy for Fog Infrastructure". In: *ICFEC'17*. 2017, pp. 97–104. cit. on p. 5

[NB17] Jiacheng Ni and Xuelian Bai. "A Review of Air Conditioning Energy Performance in Fata Centers". In: *Renewable and Sustainable Energy Reviews* 67 (2017), pp. 625–640. cit. on pp. 3, 43

[NDT16] Yanik Ngoko, Pierre-François Dutot, and Denis Trystram. "Heating as a Cloud-Service, A Position Paper (Industrial Presentation)". In: *Euro-Par 2016: Parallel Processing*. Springer International Publishing, 2016, pp. 389–401. cit. on p. 7

[Pau+18] Debayan Paul, Tanmay Chakraborty, Soumya Kanti Datta, and Debolina Paul. “IoT and Machine Learning Based Prediction of Smart Building Indoor Temperature”. In: Aug. 2018, pp. 1–6. cit. on p. 45

[Pfl+16] T. Pflanzner, A. Kertesz, B. Spinnewyn, and S. Latré. “MobIoTSim: Towards a Mobile IoT Device Simulator”. In: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. 2016, pp. 21–27. cit. on p. 24

[Phi+97] Cynthia A. Phillips, Cliff Stein, Eric Torng, and Joel Wein. “Optimal Time-Critical Scheduling via Resource Augmentation (Extended Abstract)”. In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*. STOC ’97. El Paso, Texas, USA: Association for Computing Machinery, 1997, pp. 140–149. cit. on p. 110

[Pla+09] Judit Planas, Rosa M. Badia, Eduard Ayguadé, and Jesus Labarta. “Hierarchical task-based programming with StarSs”. In: *International Journal of High Performance Computing Applications* 23.3 (2009), pp. 284–299. cit. on p. 69

[POD15] Wojciech Piatek, Ariel Oleksiak, and Georges Da Costa. “Energy and Thermal Models for Simulation of Workload and Resource Management in Computing Systems”. In: *Simulation Modelling Practice and Theory* 58 (2015). Special Issue on Techniques and Applications for Sustainable Ultrascale Computing Systems, pp. 40–54. cit. on p. 45

[Poq17] Millian Poquet. “Simulation approach for resource management. (Approche par la simulation pour la gestion de ressources)”. PhD thesis. Grenoble Alpes University, France, 2017. cit. on p. 25

[Qay+18] Tariq Qayyum, Asad W. Malik, Muazzam A. Khan Khattak, Osman Khalid, and Samee U. Khan. “FogNetSim++: A Toolkit for Modeling and Simulation of Distributed Fog Environment”. In: *IEEE Access* 6 (2018), pp. 63570–63583. cit. on pp. 23, 25

[RB08] Luiz Ramos and Ricardo Bianchini. “C-Oracle: Predictive thermal management for data centers”. In: *2008 IEEE 14th International Symposium on High Performance Computer Architecture*. 2008, pp. 111–122. cit. on p. 44

[Rua+06] Antonio Ruano, E.M. Crispim, Eusébio Conceição, and Maria Lúcio. “Prediction of Building’s Temperature Using Neural Networks Models”. In: *Energy and Buildings* 38 (June 2006), pp. 682–694. cit. on p. 45

[Shi+16] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. “Edge Computing: Vision and Challenges”. In: *IEEE Internet of Things Journal* 3.5 (Oct. 2016), pp. 637–646. cit. on p. 2

[Ska+04] Kevin Skadron, Mircea R. Stan, Karthik Sankaranarayanan, et al. “Temperature-Aware Microarchitecture: Modeling and Implementation”. In: *ACM Trans. Archit. Code Optim.* 1.1 (Mar. 2004), pp. 94–125. cit. on p. 45

[Ska+17] Olena Skarlat, Matteo Nardelli, Stefan Schulte, Michael Borkowski, and Philipp Leitner. “Optimized IoT Service Placement in the Fog”. In: *SOC* 11.4 (Dec. 2017), pp. 427–443. cit. on p. 5

[SOE17] Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. “EdgeCloudSim: An Environment for Performance Evaluation of Edge Computing Systems”. In: *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*. May 2017, pp. 39–44. cit. on pp. 23, 25

[Sot+14] Stelios Sotiriadis, Nik Bessis, Eleana Asimakopoulou, and Navonil Mustafee. “Towards Simulating the Internet of Things”. In: *2014 28th International Conference on Advanced Information Networking and Applications Workshops*. 2014, pp. 444–448. cit. on p. 24

[ST09] Erik Saule and Denis Trystram. “Multi-users scheduling in parallel systems”. In: *23rd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2009, Rome, Italy, May 23-29, 2009*. 2009, pp. 1–9. cit. on p. 109

[ST93] David B Shmoys and Éva Tardos. “An approximation algorithm for the generalized assignment problem”. In: *Mathematical programming* 62.1-3 (1993), pp. 461–474. cit. on p. 71

[Sub13] R Shankar Subramanian. “"Unsteady Heat Transfer: Lumped Thermal Capacity Model”. In: *Department of Chemical and Biomolecular Engineering, Clarkson University* (2013). cit. on pp. 44, 49

[SV05] Evgeny V. Shchepin and Nodari Vakhania. “An optimal rounding gives a better approximation for scheduling unrelated machines”. In: *Operations Research Letters* 33.2 (2005), pp. 127–133. cit. on p. 71

[THW99] Haluk Topcuoglu, Salim Hariri, and Min-You Wu. “Task scheduling algorithms for heterogeneous processors”. In: *Heterogeneous Computing Workshop (HCW)*. 1999, pp. 3–14. cit. on pp. 71, 75, 77, 98

[TS07] Bertil Thomas and Mohsen Soleimani-Mohseni. “Artificial Neural Network Models for Indoor Temperature Prediction: Investigations in two Buildings”. In: *Neural Computing and Applications* 16 (Jan. 2007), pp. 81–89. cit. on p. 45

[TZQ18] Shubbhi Taneja, Yi Zhou, and Xiao Qin. “Thermal Benchmarking and Modeling for HPC Using Big Data Applications”. In: *Future Generation Computer Systems* 87 (May 2018). cit. on p. 44

[Vel+13] Pedro Velho, Lucas Schnorr, Henri Casanova, and Arnaud Legrand. “On the Validity of Flow-level TCP Network Models for Grid and Cloud Simulations”. In: *ACM Transactions on Modeling and Computer Simulation* 23.4 (Oct. 2013). cit. on p. 23

[WB04] Andreas Weissel and Frank Bellosa. “Dynamic Thermal Management for Distributed Systems”. In: *Workshop on Temperature-Aware Computer Systems (TACS’04)*. 2004. cit. on p. 44

[Woj14] Janusz Wojtkowiak. “Lumped Thermal Capacity Model”. In: *Encyclopedia of Thermal Stresses*. Ed. by Richard B. Hetnarski. Dordrecht: Springer Netherlands, 2014, pp. 2808–2817. cit. on pp. 44, 46, 50

[WZX13] Nan Wang, Jiangfeng Zhang, and Xiaohua Xia. “Energy Consumption of Air Conditioners at Different Temperature Set Points”. In: *Energy and Buildings* 65 (2013), pp. 412–418. cit. on p. 44

[Xia+18] 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*. 2018, pp. 751–760. cit. on p. 5

[Xu+18] Chengliang Xu, Huanxin Chen, Jiangyu Wang, Yabin Guo, and Yue Yuan. “Improving Prediction Performance for Indoor Temperature in Public Buildings Based on a Novel Deep Learning Method”. In: *Building and Environment* 148 (Nov. 2018). cit. on p. 45

[YJG03] Andy B. Yoo, Morris A. Jette, and Mark Grondona. “SLURM: Simple Linux Utility for Resource Management”. In: *Job Scheduling Strategies for Parallel Processing, 9th International Workshop, JSSPP 2003, Seattle, WA, USA, June 24, 2003, Revised Papers*. 2003, pp. 44–60. cit. on p. 2

[YKD11] Azim YarKhan, Jakub Kurzak, and Jack J. Dongarra. *QUARK Users’ Guide: QUeueing And Runtime for Kernels*. UTK ICL. 2011. cit. on p. 69

[You+18] Ashkan Yousefpour, Ashish Patil, Genya Ishigaki, et al. “QoS-aware Dynamic Fog Service Provisioning”. In: *CoRR* abs/1802.00800 (2018). arXiv: `1802.00800`. cit. on p. 5

[YWC20] Yunqiang Yin, Dujuan Wang, and T.C.E. Cheng. *Due Date-Related Scheduling with Two Agents - Models and Algorithms*. Springer, 2020. cit. on p. 109

[Zen+17] Xuezhi Zeng, Saurabh Kumar Garg, Peter Strazdins, et al. “IOTSim: a Cloud Based Simulator for Analysing IoT Applications”. In: *J. Syst. Archit.* 72.C (Jan. 2017), pp. 93–107. cit. on p. 24

[Zha+15] Ben Zhang, Nitesh Mor, John Kolb, et al. “The Cloud is Not Enough: Saving IoT from the Cloud.” In: *HotStorage*. 2015. cit. on p. 2

[Zha+18] Kaicheng Zhang, Akhil Guliani, Seda Ogrenci-Memik, et al. “Machine Learning-Based Temperature Prediction for Runtime Thermal Management Across System Components”. In: *IEEE Transactions on Parallel and Distributed Systems* 29.2 (2018), pp. 405–419. cit. on p. 44

List of Figures

List of Tables

List of Communications

Additionally, the work conducted in preparation of this dissertation led to the following communications.

International journals

- Marcos Amaris, Giorgio Lucarelli, Clément Mommessin, and Denis Trystram. "Generic Algorithms for Scheduling Applications on Heterogeneous Platforms". In: *Concurrency and Computation: Practice and Experience* (July 2018), pp. 1–29.
- Olivier Beaumont, Louis-Claude Canon, Lionel Eyraud-Dubois, et al. "Scheduling on Two Types of Resources: A Survey". In: *ACM Comput. Surv.* 53.3 (May 2020).

International conferences with proceedings

- Marcos Amaris, Giorgio Lucarelli, Clément Mommessin, and Denis Trystram. "Generic Algorithms for Scheduling Applications on Hybrid Multi-core Machines". In: *Euro-Par: International Conference on Parallel and Distributed Computing*. 2017, pp. 220–231.
- Adwait Bauskar, Anderson da Silva, Adrien Lebre, et al. "Investigating Placement Challenges in Edge Infrastructures through a Common Simulator". In: *SBAC-PAD* (2020).
- Clément Mommessin, Matthieu Dreher, Tom Peterka, and Bruno Raffin. "Automatic Data Filtering for In Situ Workflows". In: *IEEE International Conference on Cluster Computing*. Hawai, United States, Sept. 2017.

International workshops (invited)

- Clement Mommessin, Giorgio Lucarelli, and Denis Trystram. "Scheduling at the Edge". In: *14th Scheduling for Large Scale Systems Workshop*. Bordeaux, France, June 2019, pp. 1–28.
- Denis Trystram, Giorgio Lucarelli, Clement Mommessin, and Yanik Ngoko. "Challenges for scheduling at the Edge". In: *MCST 2019 - Workshop on Mathematical Challenges in Scheduling Theory*. Sanya, China, Oct. 2019, pp. 1–33.

Abstract

The world of Information Technology (IT) is in constant evolution. With the explosion of the number of digital and connected devices in our everyday life, the IT infrastructures have to face an ever growing amount of users, computing requests and data generated. The Internet of Things has seen the development of computing platforms at the edge of the network to bridge the gap between the connected devices and the Cloud, called the Edge Computing. In the domain of High Performance Computing, the parallel programs executed on the platforms require always more computing power in a search for improved performances. Besides, we observed in the past years a diversification of the hardware composing these infrastructures. This complexification of the (network of) computing platforms pose several optimisation challenges that can appear at different levels. In particular, it led to a need for better management systems to make an efficient usage of the heterogeneous resources composing these platforms.

The work presented in this thesis focuses on resource optimisation problems for distributed and parallel platforms of the Edge Computing and High Performance Computing domains. In both cases, we study the modelling of the problems and propose methods and algorithms to optimise the resource management for better performance, in terms of quality of the solutions. The problems are studied from both theoretical and practical perspectives. More specifically, we study the resource management problems at multiple levels of the *Qarnot Computing* platform, an Edge Computing production platform mostly composed of computing resources deployed in heaters of smart-buildings. In this regard, we propose extensions to the Batsim simulator to enable the simulation of Edge Computing platforms and ease the design, development and comparison of data and job placement policies in such platforms. Then, we design a new temperature prediction method for smart-buildings and propose a formulation of a new scheduling problem with two agents on multiple machines.

In parallel, we study the problem of scheduling applications on hybrid multi-core machines with the objective of minimising the completion time of the overall application. We survey existing algorithms providing performance guarantees on the constructed schedules and propose two new algorithms for different settings of the problem, proving performance guarantees for both. Then, we conduct an experimental campaign to compare in practice the relative performance of the new algorithms with existing solutions in the literature.

Résumé

Le monde des Technologies de l'Information (IT) est en constante évolution. Avec l'explosion du nombre d'appareils numériques et connectés dans notre vie de tous les jours, les infrastructures IT doivent faire face à une constante augmentation du nombre d'utilisateurs, de requêtes informatiques et de données générées. L'Internet des Objets a vu le développement de plates-formes de calcul en bordure du réseau pour combler l'écart entre les appareils connectés et le Cloud, appelé le Edge Computing. Dans le domaine du Calcul à Haute Performance, les programmes parallèles exécutés sur les plates-formes demandent toujours plus de puissance de calcul à la recherche d'une amélioration des performances. De plus, il a été observé au cours des dernières années une diversification des composants matériels dans ces infrastructures. Cette complexification des (réseaux de) plates-formes de calculs pose plusieurs problèmes d'optimisation qui peuvent apparaître à divers niveaux. En particulier, cela a mené au besoin de meilleurs systèmes de gestion pour une utilisation efficace des ressources hétérogènes qui composent ces plates-formes.

Le travail présenté dans cette thèse se focalise sur des problèmes d'optimisation de ressources pour les plates-formes parallèles et distribuées du Calcul à Haute Performance et du Edge Computing. Dans les deux cas, nous étudions la modélisation des problèmes et nous proposons des méthodes et des algorithmes de gestion de ressources pour de meilleures performances. Les problèmes sont étudiés à la fois sur des plans théoriques et pratiques. Plus spécifiquement, nous étudions les problèmes de gestion de ressources à différents niveaux de la plate-forme *Qarnot Computing*, une plate-forme de production Edge principalement composée de ressources de calculs déployées dans des radiateurs de bâtiments intelligents. Pour cela, nous proposons des extensions au simulateur Batsim pour permettre la simulation de plates-formes Edge et pour faciliter le design, le développement et la comparaison de politiques de placement de données et de tâches sur de telles plates-formes. Ensuite, nous proposons une nouvelle méthode de prédiction de la température pour des bâtiments intelligents et nous formulons un nouveau problème d'ordonnancement à deux agents sur machines multiples.

En parallèle, nous étudions le problème d'ordonnancement d'applications sur machines multi-cœur hybrides dont l'objectif est la minimisation du temps total de complétion de l'application. Nous faisons une revue des algorithmes existants avec des garanties de performance, puis nous proposons deux nouveaux algorithmes pour différentes variantes du problème et nous donnons des preuves de leur garanties de performance. Enfin, nous conduisons une campagne expérimentale pour comparer la performance relative de nos algorithmes avec des solutions existantes de la littérature.