Scheduling in cloud data center powered by renewable energy only with mixed phases-based workload

Gustavo Rostirolla

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par

Gustavo ROSTIROLLA

Le 25 novembre 2019

Ordonnancement dans un centre de calculs alimenté par des sources d'énergie renouvelables sans connexion au réseau avec une charge de travail mixte basée sur des phases

---

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Patricia STOLF et Stéphane CAUX

Jury

Mme Ivona Brandic, Rapporteure
M. Laurent George, Rapporteur
Mme Anne-Cécile Orgerie, Examinatrice
M. Benjamin Depardon, Examinateur
M. Paul Renaud-Goud, Examinateur
M. Pierre Lopez, Examinateur
Mme Patricia STOLF, Directrice de thèse
M. Stéphane Caux, Co-directeur de thèse

Abstract

Due to the increase of cloud, web-services and high performance computing demands all over the world, datacenters are now known to be one of the biggest actors when talking about energy consumption. In 2006 alone, datacenters were responsible for consuming 61.4 billion kWh in the United States. When looking at the global scenario, datacenters are currently consuming more energy than the entire United Kingdom, representing about 1.3% of world's electricity consumption, and being even called the factories of the digital age.

Supplying datacenters with clean-to-use renewable energy is therefore essential to help mitigate climate change. The vast majority of cloud provider companies that claim to use green energy supply on their datacenters consider the classical grid, and deploy the solar panels/wind turbines somewhere else and sell the energy to electricity companies, which incurs in energy losses when the electricity travels throughout the grid. Even though several efforts have been conducted at the computing level in datacenters partially powered by renewable energy sources, the scheduling considering on site renewable energy sources and its variations, without connection to the grid can still be widely explored.

Since energy efficiency in datacenters is directly related to the resource consumption of the computing nodes, performance optimization and an efficient load scheduling are essential for energy saving. Today, we observe the use of cloud computing as the basis of datacenters, either in a public or private fashion. The main particularity of our approach is that we consider a power envelope composed only by renewable energy as a constraint, hence with a variable amount of power available at each moment. The scheduling under this kind of constraint becomes more complex: without further checks, we are not ensured that a running task will run until completion.

We start by addressing the IT load scheduling of batch tasks, which are characterized by their release time, due date and resource demand, in a cloud datacenter while respecting the aforementioned power envelope. The data utilized for the batch tasks comes from datacenter traces, containing CPU, memory and network values. The power envelopes considered, represent an

estimation which would be provided by a power decision module and is the expected power production based on weather forecasts. The aim is to maximize the Quality of Service with a variable constraint on electrical power.

Furthermore, we explore a workload composed by batch and services, where the resources consumption varies over time. The traces utilized for the service tasks originate from business critical datacenter. In this case we rely on the concept of phases, where each significant resource change in the resources consumption constitutes a new phase of the given task. In this task model phases could also receive less resources than requested. The reduction of resources can impact the QoS and consequently the datacenter profit. In this approach we also include the concept of cross-correlation to evaluate where to place a task under a power curve, and what is the best node to place tasks together (i.e. sharing resources).

Finally, considering the previous workload of batch tasks and services, we present an approach towards handling unexpected events in the datacenter. More specifically we focus on IT related events such as tasks arriving at any given time, demanding more or less resources than expected, or having a different finish time than what was initially expected. We adapt the proposed algorithms to take actions depending on which event occurs, e.g. task degradation to reduce the impact on the datacenter profit.

Résumé

Les centres de données sont reconnus pour être l'un des principaux acteurs en matière de consommation d'énergie du fait de l'augmentation de l'utilisation du cloud, des services web et des applications de calcul haute performance dans le monde entier. En 2006, les centres de données ont consommé 61,4 milliards de kWh aux états-Unis. Au niveau mondial, les centres de données consomment actuellement plus d'énergie que l'ensemble du Royaume-Uni, c'est-à-dire environ 1,3% de la consommation électrique mondiale, et ils sont de fait appelés les usines de l'ère numérique.

Un des moyens d'atténuer le changement climatique est d'alimenter les centres de données en énergie renouvelable (énergie propre). La grande majorité des fournisseurs de cloud computing qui prétendent alimenter leurs centres de données en énergie verte sont en fait connectés au réseau classique et déploient des panneaux solaires et des éoliennes ailleurs puis vendent l'électricité produite aux compagnies d'électricité. Cette approche entraîne des pertes d'énergie lorsque l'électricité traverse le réseau. Même si différents efforts ont été réalisés au niveau informatique dans les centres de données partiellement alimentés par des énergies renouvelables, des améliorations sont encore possibles notamment concernant l'ordonnancement prenant en compte les sources d'énergie renouvelables sur site sans connexion au réseau et leur intermittence. C'est le but du projet ANR DataZERO, dans le cadre duquel cette thèse a été réalisée.

L'efficacité énergétique dans les centres de données étant directement liée à la consommation de ressources d'un nœud de calcul, l'optimisation des performances et un ordonnancement efficace des calculs sont essentiels pour économiser l'énergie. La spécificité principale de notre approche est de placer le centre de données sous une contrainte de puissance, provenant entièrement d'énergies renouvelables : la puissance disponible peut ainsi varier au cours du temps. L'ordonnancement de tâches sous ce genre de contrainte rend le problème plus difficile, puisqu'on doit notamment s'assurer qu'une tâche qui commence aura assez d'énergie pour aller jusqu'à son terme.

Dans cette thèse, nous commençons par proposer une planification de tâches de type "batch" qui se caractérisent par leur instant d'arrivée, leur date

d'échéance et leurs demandes de ressources tout en respectant une contrainte de puissance. Les données utilisées pour les tâches de type batch viennent de traces de centres de données et contiennent des mesures de consommation CPU, mémoire et réseau. Quant aux enveloppes de puissance considérées, elles représentent ce que pourrait fournir un module de décision électrique, c'est-à-dire la production d'énergie prévue (énergie renouvelable seulement) basée sur les prévisions météorologiques. L'objectif est de maximiser la Qualité de Service avec une contrainte sur la puissance électrique.

Par la suite, nous examinons une charge de travail composée de tâches de type "batch" et de services, où la consommation des ressources varie au cours du temps. Les tracecs utilisées pour les services proviennent d'une centre de données à "business critique". Dans ce cadre, nous envisageons le concpet de phases, dans lequel les changements significatifs de consommation de resources à l'intérieur d'une même tâche marquent le début d'une nouvelle phase. Nous considérons également un modèle de tâches pouvant recevoir moins de ressources que demandées. Nous étudions l'impact de ce modèle sur le profit du centre de donnèes pour chaque type de tâche. Nous intégrons aussi le concept de "corrélation croisée" pour évaluer où placer une tâche selon une courbe de puissance afin de trouver le meilleur nœud pour placer plusieurs tâches (c.-à-d. Partager les ressources).

Enfin, nous présentons une approche pour faire face à des événements inattendus tels que des tâches exigeant plus ou moins de ressources que prévu, ou ayant une date de fin différente, et étudions les actions possibles, compte tenu de la dégradation des tâches et de l'impact sur le profit du centre de données.

Acknowledgments

First I would like to express my deepest gratitude to my advisors Patricia Stolf and Stéphane Caux for their excellent guidance and patience, creating a great atmosphere to pursue my scientific goals. I am also very grateful to Paul Renaud-Goud, who also guided me from the early stages of this research. I could not have imagined having better advisors and mentors for my Ph.D studies.

I would also like to thank the remainder of the thesis committee: Anne-Cécile Orgerie, Benjamin Depardon, Ivona Brandic, Laurent George and Pierre Lopez for their valuable time reviewing this dissertation and the questions that will incentive me to widen my research from various perspectives.

This research has been carried out with the support of the DataZERO project as well as Laplace and IRIT laboratories. I would like to thank all the members of the project for the valuable discussions during the group meetings. I would also like to express my gratitude for the SEPIA team, specially Jean-Marc Pierson, Georges da Costa and Amal Sayah.

For my co-workers and the frequent office visitors Guo Chaopeng, Léo Grange, Tanissia Djemai, Zong Yi Liu, Minh-Thuyen Thi, Berk Celik, Malik Irain, Tristan Salord, Ophélie Fraisier, Bilal Fakih, Morgan Seguela, Florent Dubois, Meryem Zaid and the brazilian friends. Thank you for all the coffees and beers shared, and specially the discussions that broadened my point of view in several subjects. I'm also thankful to Pedro Velho and Rodrigo da Rosa Righi who influenced me to pursue the scientific path through their enthusiasm and love for teaching.

Finally, I would like to thank my family, who taught me the value of life and always encouraged me to move forward, while providing me a safe place to call home.

When we least expect it, life sets us a challenge to test our courage and willingness to change; at such a moment, there is no point in pretending that nothing has happened or in saying that we are not yet ready. The challenge will not wait. Life does not look back.

— Paulo Coelho.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Nowadays, data centers are one of the most energy consuming facilities, being even called the factories of the digital age. Data centers are large computing facilities where several devices work to meet users demand. Due to the increase of data processing needs and the significant growth [42] of platforms such as Google, Amazon and Facebook to cite a few, data centers have to continue increasing in size and processing capability.

The majority of data centers rely in the cloud computing model [42] due to its advantages. Among these advantages we can cite the main ones (i) virtualization of computing resources and (ii) the pay-as-you-go billing model [60]. These features reduce of entry cost for smaller business trying to perform compute-intensive activities where an almost immediate access to hardware resources can be achieved without upfront capital. Also, the cloud model makes easier for enterprises to scale their services as the resources requirement grow. This computing model enables the access to computing resources in a transparent way for several applications and users, occasioning a significant growth on the cloud data centers. In 2006 the energy consumption of data centers reached 61.4 billion kWh only in the United States [78], and estimations of being in charge of consuming about 1.3% of world's electricity [72]. In 2019 90% of companies have some process going through cloud computing and by 2021 cloud data centers are expected to process 94% of the computing workload [96]. These high values lead to several researches such as the ones presented in Kong and Liu [71], Orgerie et al. [97] and Mastelic et al. [89] to cite a few, where the aim is to improve the energy efficiency of data centers, in some of cases considering the integration of renewable energy sources.

Large size cloud data centers such as Amazon EC2 [6] (47.8% of the market

share [42]) are already taking steps to improve the energy efficiency of its data centers, with other providers going in the same direction. The final goal is to have efficient data centers and increase the part of the energy that comes from renewable energy sources [54]. Nevertheless, because of the intermittency of renewable energy sources, all the typical approaches consider that the cloud data center is always connected to the power grid, and the usage of renewable energies come with several challenges and constraints.

To tackle some of the challenges, funds have been allocated to projects such as DataZERO, where the aim is to investigate the possible solutions to design and operate a datacenter powered only by renewable energy. Moreover the project focuses on a distributed approach where a negotiation occurs among the different actors (data center and energy supplier) in order to find an agreement. In addition to the design of an efficient negotiation process, the objectives also include to efficiently control the power coming from different sources of energy. On the IT side, the main problem is the scheduling of tasks on the IT servers under the constraint of the power production over time. In this thesis we focus on the latter problem, concerning the IT scheduling. The goal is to design and prove the efficiency of a novel approach on how to schedule a mix of applications in cloud data centers without any connection to the grid, while respecting this variable energy production during time.

1.2 Motivation

Manufacture and powering of Information and Communication Technologies (ICT) devices was estimated to consume over 7% of global electricity in 2012 [54]. Projections anticipate that in 2030 data centers electricity demand alone can reach 13% of global electricity consumption [113]. This increase is in its biggest part due to the fact that cloud data centers are a fundamental part of ubiquitous [126] and pervasive [110] computing, where the access to information anytime and everywhere is an indispensable feature of everyday lives. More recent reports [25, 31] show that globally, devices and connections are growing faster (10% Compound Annual Growth Rate - CAGR) than the population. The same report also presents a growing number of M2M (Machine-to-Machine) applications, such as smart meters, video surveillance, healthcare monitoring, transportation, and package or asset tracking all of which rely on services running in a cloud data center.

This significant increase in the energy consumption of cloud data centers leads to a number of works that aim to improve the energy efficiency in these systems [3, 11, 45, 47] through leverages such as DVFS (Dynamic Voltage and Frequency Scaling), Software and Hardware improvements and workload

consolidation. While the improvements in efficiency are promising, they are yet to catch up with the growth in data traffic and consumption of cloud services [103].

In the power production side, we observe an increase in the global total capacity of renewable power of 9% between the years of 2015 and 2016 [107] with continuous additions points towards greener data centers approaches. The regular infrastructure of a cloud data center, from the power point of view, consists in 2 circuits [118] capable of supplying power to the data center alone. With the additions in renewable energy sources, some approaches summarized in Kong and Liu [71] change one of these energy supply circuits for a renewable one. Nevertheless, all the works evaluated by the authors still consider the grid (brown energy) as a backup when not enough renewable energy is produced.

This absence of this brown energy as backup brings new challenges from the IT perspective in how to handle the intermittent nature of the renewable energy sources. Some directions on how to manage the workload when these facilities are powered only with renewable energy and no connection to the grid were presented in Sharma et al. [112], the only work outside of DataZERO project that evaluates such a scenario. In this case, the authors focus mainly on deactivating server when there is no energy. The key point is that this type of action is not realistic for the vast majority of applications running in cloud platforms, since the authors assume that all the applications would instantaneously stop and resume their execution when energy is available. The authors also state that application modifications are necessary to adapt traditional server-based applications for "blinking", since these applications implicitly assume always-on or mostly-on servers. Overall, the reliability deterioration when utilizing up to 100% of renewable energy sources need further studies [40, 74].

In our case none of the power supply branches will be connected to the grid, since DataZERO proposes a "Breakthrough" data center architecture [101]. Instead of a regular UPS we have a green energy sources only connection, which comes in line with the new goals for greener data center [54]. The complete removal of the connection to the power grid imposes several new challenges, such as the dimensioning of the renewable energy sources and storage to provide enough energy to the data center, specially in the hours of peak operation. Another challenge is the adaptation of the IT workload to fit under the amount of energy that is produced.

In this thesis we focus mainly on the IT challengers, where the power available for a given time interval (hours/days/weeks) would come as an input. In other words, contrary to the works previously evaluated, and the ones that will be further presented, we focus on adapting the workload to the power

available and not the contrary. Additionally, the power would come strictly from renewable energy sources, and the idea is to provide feasible scheduling solutions with reductions in applications performance if necessary, associated with a price (profit to the data center) and quality of service metric.

1.3 Problem Statement

Recently, there have been a number of works that investigate the management of data center power consumption. As examples we can cite job scheduling, virtual machine migration, service rate allocation, software controlling, shifting demand in time, resources consolidation and DVFS. Such approaches have mainly been explored in a context where the data center is connected to the power grid [71], and removing this connection can impose several challenges [101] specially on how to handle the servers and applications on the data center side.

A first challenge is the classical problem of applications placement. The cloud provider needs to decide in which one of the cloud resources the application will be executed and when. One of the applications (here also called tasks) considered in this thesis can be defined by a start time and expected finish time between which an amount of computation needs to be done. In our case we also consider that this amount of computation have variations over time and that it can be executed according to what the user specified or with a smaller performance. In the case of smaller performance an increase in the execution time would occur risking a violation of the expected finish time. We call this specific type of application batch tasks.

The other type of applications considered are services where there is only a defined start time and there is no knowledge of when the application ends. For this type of applications a reduction in the expected computational resources translates into users not being served, though implying a reduction in the Quality of Service (QoS).

A scenario where the tasks would need to receive less computing resources or change their execution time is presented in Figure 1.1. In the presented case we can see a set of 12 tasks and their corresponding power consumption. As an illustrative example we consider a 1:1 ratio between computing resources requested and power consumption. The red line on top represents the total power $\times$ time that is expected to be produced in that day. In this case, if we try to place with the requested amount of computing resources and in the moment that the users submit each task, the energy produced by the renewable energy sources would not be enough. As we can see in the figure the renewable energy production can be intermittent due to its source (solar or wind turbines) an a high variability can occur.

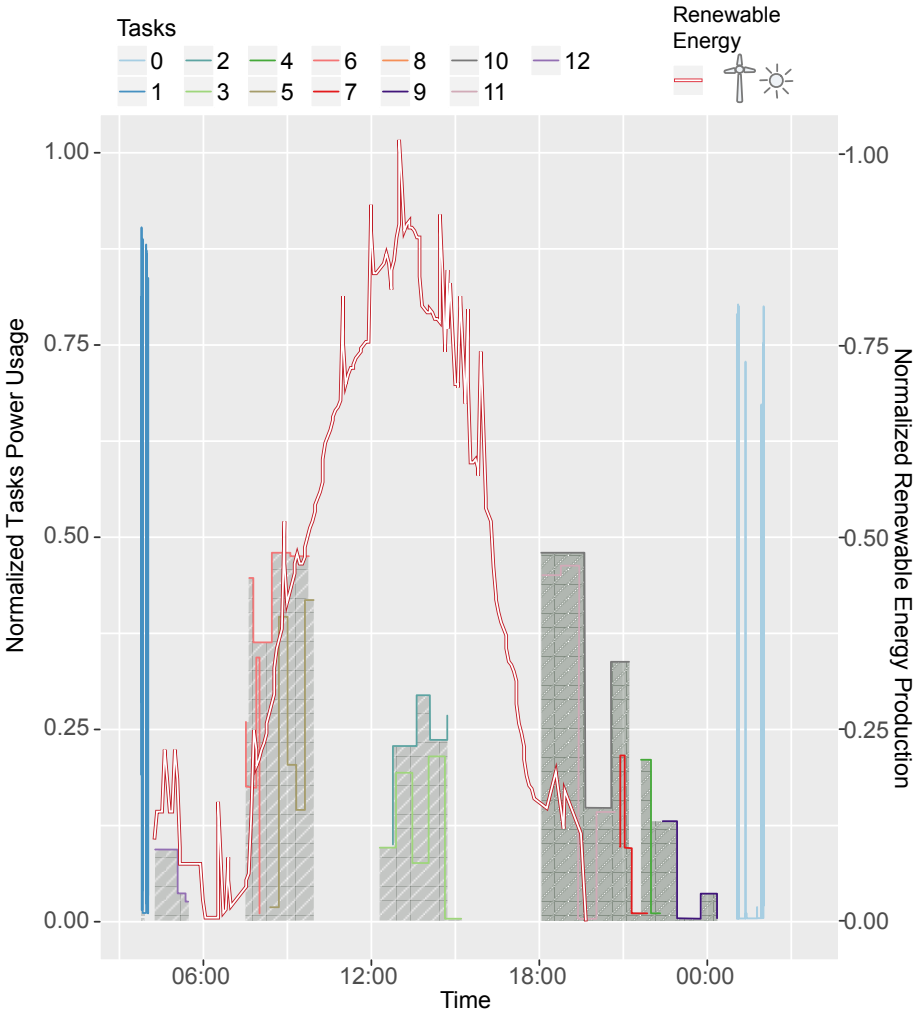


Figure 1.1: Task power consumption and renewable energy production in a data center powered only by renewable energy.

In this scenario, we can highlight the second challenge imposed by this intermittent nature: how to handle the moments where there is not enough energy to execute a given application, and what are the possible leverages that can be utilized in order to not compromise the application's QoS? Here QoS can be reduced if the task finishes after the expected finish time in case of batch, or if the resources allocated to services are reduced. Also, in case of performance degradation, i.e. service receiving less resources or batch finishing after the expected time, what tasks should be degraded. Finally, another point is if all the users should get financial compensations as done today by the traditional cloud data centers disregarding their flexibility (time between task submission

and finish time)?

1.4 Research Goals

In this thesis we limit our scope to single data centers only with renewable energy. We focus mainly on the IT part where we decide where to place each task among the several computing resources in the data center, choose the amount of resources delivered, as well as the time where the tasks start executing. The energy management is considered as an input of the system according to the DataZERO project. Considering the challenges previously highlighted, the research question that we aim to answer is: "How to schedule a mix of batch and services in a cloud data center with a power envelope as constraint?".

1.5 Approach to Manage the Problem

We utilized an incremental approach for the highlighted problems. We start by studying the task scheduling of simplified batch tasks (constant resource consumption) with a power envelope as constraint in Chapter 4. The evaluated tasks originate from traces of Google datacenters and to schedule them we focus on the extension of classical literature algorithms, also utilizing leverages such as DVFS and switch nodes on/off when unused. To evaluate the approach we consider the number of tasks that are violated (exceed the expected finish time) as a QoS metric.

Later, in Chapter 5 we include a detailed workload model with a mix of batch and service tasks also based on real cloud data centers. We also considered resources variations during the execution time and algorithms that explore these variations. In this same approach we include a financial profit metric, detailed in Section 3.6 that considers the impact when resources cannot be delivered to the users in case of services and violations in the expected finish time in case of batch tasks. Several algorithms are proposed to handle the scheduling of these tasks, including variations that utilize cross-correlation to find a better placement. Finally, we explore an online variation of the previous approach in Chapter 6 and how the system behaves when the behavior of the tasks is not equal to the expected one (tasks having a longer duration or consuming more resources for instance).

1.6 Summary of Contributions

In this thesis we proposed 5 contributions, where the primary contributions are the efficient scheduling heuristics and meta heuristics for cloud data centers powered exclusively by renewable energy sources discussed in Chapter 4, the detailed mixed workload approach presented in Chapter 5 and an online adaptation is presented in Chapter 6. The secondary contributions is the extensive evaluation process that each of those approaches had to undergo to be validated and their specificities. We highlight that each one of the approaches takes in consideration the DataZERO scenario detailed in Section 2.5. We organize these contributions as follows:

1 Single Phase Batch Scheduling [21, 22] (Chapter 4): Set of heuristic and meta heuristic approaches to schedule single phases batch tasks under a power envelope maximizing the performance;

2 Multi Phases Mixed Workload Scheduling (Chapter 5): Set of heuristic approaches to schedule multi phases batch and services under a power envelope maximizing the profit;

3 Phases Based Applications with Degradation [24] (Chapter 5): An extension of phases based applications [38] including the degradation factor and what is the impact of it in the proposed scheduling algorithms;

4 Cloud Based Profit with Flexibility [24] (Section 3.6): An extension of traditional compensation mechanism in case of violation in cloud based data centers to fairly compensate users that are willing to give more time flexibility to execute a given application (which goes in line with better utilization of renewable energies);

5 Multi Phases Mixed Workload Online Scheduling with Uncertainty (Chapter 6): An adaptation of the offline multi phases approach in order to consider scenarios where tasks arrive at any time and have a different resources consumption and execution time than the one expected.

1.7 Publications and Communication

Submitted Journals:

- Gustavo Rostirolla, Leo Grange, Minh-Thuyen Thi, Patricia Stolf, Jean-Marc Pierson, Georges da Costa, Gwilherm Baudic, Marwa Haddad,

Ayham Kassab, Jean-Marc Nicod, Laurent Philippe, Veronika Rehn-Sonigo, Robin Roche, Berk Celik, Stephane Caux, Jerome Lecuivre. Sizing and Management of Energy Sources for Green Datacenters with Renewable Energy. Renewable & Sustainable Energy Reviews.

- Minh-Thuyen Thi, Jean-Marc Pierson, Georges Da Costa, Patricia Stolf,Jean-Marc Nicod,Gustavo Rostirolla, Marwa Haddad. Negotiation Game for Joint IT and Energy Management in Green Datacenters. Future Generation Computer Systems.

Accepted Journals:

- Jean-Marc Pierson, Gwilherm Baudic, Stéphane Caux, Berk Celik, Georges Da Costa, Leo Grange, Marwa Haddad, Jerome Lecuivre, Jean-Marc Nicod, Laurent Philippe, Veronika Rehn-Sonigo, Robin Roche, Gustavo Rostirolla, Amal Sayah, Patricia Stolf, Minh-Thuyen Thi and Cristophe Varnier. "DATAZERO: Datacenter With Zero Emission and Robust Management Using Renewable Energy," in IEEE Access, vol. 7, pp. 103209-103230, 2019.

Accepted Peer Reviewed Conferences:

- Stephane Caux, Paul Renaud-Goud, Gustavo Rostirolla, Patricia Stolf. IT Optimization for Datacenters Under Renewable Power Constraint. Euro-Par 2018: Parallel Processing. Euro-Par 2018.

- Stephane Caux, Gustavo Rostirolla, Patricia Stolf. Smart Datacenter Electrical Load Model for Renewable Sources Management. International Conference on Renewable Energies and Power Quality (ICREPQ 2018), Salamanca, Spain. Vol. 16, European Association for the Development of Renewable Energies, Environment and Power Quality, p. 127-132, April 2018.

- Berk Celik, Gustavo Rostirolla, Stephane Caux, Paul Renaud-Goud, Patricia Stolf. Analysis of demand response for datacenter energy management using GA and time-of-use prices. IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe) 2019.

- Stephane Caux, Paul Renaud-Goud, Gustavo Rostirolla, Patricia Stolf. Phase-Based Tasks Scheduling in Data Centers Powered Exclusively by Renewable Energy. Symposium on Computer Architecture and High Performance Computing (SBAC-PAD) 2019.

Others Disseminations:

- Talk: Optimisation IT sous contrainte de puissance provenant de sources renouvelables, GreenDays@Toulouse, July 2018
- Poster: Challenges and Solutions for Data Center Powered by on Site Renewable Energy Only. Gustavo Rostirolla, Leo Grange, Patricia Stolf, Stephane Caux, Georges da Costa, Jean-Marc Pierson. European Science Open Forum (ESOF 2018), Toulouse 9-14 July 2018.
- Talk: Tasks Scheduling in Cloud Datacenters Under Renewable Power Constraint, Journées Cloud 2018, Troyes, September 2018
- Talk: Phase-based tasks scheduling in data centers powered exclusively by renewable energy. Workshop on Renewable Energy-Powered Datacenters. October 14-16, 2019, in Besancon, France.

1.8 Dissertation Outline

The remaining portion of the dissertation is organized as follows:

Chapter 2 - Related Work and Context: In this chapter we focus on presenting the fundamentals that will help the reader to understand the remainder of the work. We start by presenting concepts such as cloud computing, what are tasks, the main components when talking about energy consumption in cloud data centers and how is renewable energy inserted in this context. Then, we present the DataZERO project context where this work is inserted, how tasks can be placed and what are the problems involved. After we present a comprehensive list of works that approach the energy consumption optimization in cloud data centers powered by renewable energies and conclude by discussing the gaps in studied approaches.

Chapter 3 - Model: In this chapter we focus mainly in the model abstractions that will be utilized in the remainder of this work. We first present the IT infrastructure model, followed by the power consumption model and the distinctions that will be made between the two different approaches (single phases and multi phases based). We then introduce the tasks model for each approach and their particularities, followed by the data sources for the workload generators and how these generators can be utilized for future approaches. Finally, we present the power production data sources for wind turbines and solar panels considered in this work and how all the aforementioned features are combined in the chosen simulator.

Chapter 4 - Single Phase Batch Scheduling: In this chapter we propose to optimize the IT scheduling of batch tasks to execute tasks within a given power envelope of only renewable energy as a constraint. We utilize

greedy heuristics as well as meta-heuristics to perform the task placement, with aim at minimizing the due date violations. The results presented are related to the publications in [22] and [21].

Chapter 5 - Multi Phases Mixed Workload Scheduling: In this Chapter we improve the application model from the previous chapter and added a cost model for the task placement. We utilize greedy heuristics, as in the previous chapter, and adapt them to handle the new task and cost models. We also introduce the usage of cross-correlation in the algorithms and the reduction of the delivered resources (degradation). The results presented are related to the publication in [24] and [23].

Chapter 6 - Towards a Multi Phases Mixed Workload Online Scheduling: In this Chapter we consider the same application and cost model from the previous one. We introduce online events such as task arrival, and resources consumption changes that could occur during the application execution. These variations are handled by reactive actions, utilizing an adaptation of some of the algorithms previously presented.

Chapter 7 - Conclusions and Perspectives: Finally, in this chapter we make the overall conclusions and summarize the contributions of this work, along with the discussion of future works.

Chapter 2

Related Work and Context

In this chapter we focus on presenting the fundamentals that will help the reader to understand the remainder of the work. We start by presenting concepts such as cloud computing, what are tasks, the main components when talking about energy consumption in cloud data centers and how is renewable energy inserted in this context. Then, we present the DataZERO project context where this work is inserted, how tasks can be placed and what are the problems involved. After we present a comprehensive list of works that approach the energy consumption optimization in cloud data centers powered by renewable energies and conclude by discussing the gaps in studied approaches.

2.1 Cloud Computing

The term cloud computing was coined by Compaq in 1996 [32]. The model appeared commercially in early 2000s through which customers could access computing services over the Internet and pay for what they use (Pay-as-you-go). Its popularization started in late 2000s led by enterprises such as Amazon and Microsoft. This new paradox changed the way companies managed their computing infrastructures. In this section, we discuss in more details the main characteristics of Cloud Computing and what are the forms of deployment.

2.1.1 Definition of Cloud Computing

By the definition of [60], cloud computing is a model for allowing ubiquitous, convenient and on-demand access to a shared set of configurable computing resources that can be quickly allocated and released with minimal management effort. With the introduction of cloud computing, tasks that were previously executed locally are now placed on servers of unknown location over the Internet.

The two key actors in Cloud Computing are: (i) Cloud computing service providers and (ii) customers (here also called clients/users). A Cloud computing service provider (or cloud provider) is the entity, person or organization responsible for making a service available to the cloud customers. The cloud provider is also responsible for managing the necessary infrastructure to deliver its services to the cloud service customers according to the established contract.

In the year of 2018 for instance, we can cite as the main cloud providers [42]: Amazon EC2 [6] with 47.8% of the market share, Microsoft Azure [92] with 15.5%, Alibaba Cloud [4] with 7.7%, Google Cloud [50] with 4.0% and IBM Cloud [63] with 1.8%.

A cloud service customer is the person or organization that maintains a business contract and utilizes the service of cloud providers. In this case, cloud computing services are served on-demand and accessed through the Internet. This means that customers can request computing resources from the cloud providers at any time.

Cloud provider and customer are linked by a contract. This contract, as defined by the National Institute of Standards and Technology (NIST), is called Service Level Agreement (SLA) [9]. It establishes prices, time to recover from operational failures and Quality of Services (QoS) [111] that should be respected. Here we follow the NIST definition of QoS [111] which is the measurable end-to-end performance properties of a network service, as to satisfy specific customer application requirements. The properties can include throughput, transit delay, error rates, security, packet loss, resources violation, execution time violation and so on. We detail after on each of our approaches what was the specific QoS on each case.

2.1.2 Cloud Computing Deployment Type

Regarding the deployment, cloud infrastructures can be maintained in the following ways: (i) Private Cloud; (ii) Public Cloud; (iii) Community Cloud; and (iv) Hybrid Cloud [104]. Below we detail these deployment models, according to the definitions of [90] and [104]:

1. Private Cloud: The cloud infrastructure is provisioned for the sole use of a single organization. It may be owned, managed and operated by the organization, a third party, or some combination of both, and may or may not be physically located at the organization. As example we can cite Amazon VPC, Rackspace Private Cloud and VMware Private Cloud;

2. Public Cloud: The cloud infrastructure is provisioned to be utilized by the general public and this infrastructure is shared among multiple users.

It can be owned, managed, and operated by a company, university, or government organization. It is located in the cloud provider. As example we have Amazon E2C, Google Cloud and Microsoft Azure;

3. Community Cloud: The cloud infrastructure is provisioned for exclusive use by a specific community that has common concerns. It may be owned, managed and operated by one or more community organizations or a third party, and may or may not physically exist at the site. The most common examples are governmental clouds for regulated healthcare such as e-SUS;

4. Hybrid Cloud: The cloud infrastructure is a composition of two or more cloud infrastructures (private, community or public) that remain single entities but are joined by standardized or proprietary technology.

2.1.3 Service Models Provided by Cloud Computing

In a cloud the offered computing resources can range from the physical resources (hardware like CPU and memory), as well as software resources. These different options are called service templates, and offered by a cloud provider according to the type of resources that are available [104]. Each of the available service models is detailed below, according to the definitions of [90] and [104]:

1. SaaS - Software as a Service: The capacity provided to the user is limited to applications running on a cloud infrastructure. Applications are accessible from multiple client devices and the consumer does not manage or control the infrastructure where the software is hosted, this is the responsibility of the service provider.

2. PaaS - Platform as a Service: The capacity provided to the user in this case is located one level below SaaS where applications can be deployed to a given cloud infrastructure where programming languages, libraries, services, and support tools are available. The consumer does not manage or control cloud infrastructures such as network, servers, operating systems or storage, but has control over deployed applications and configurations.

3. IaaS - Infrastructure as a Service: In this model, the user is offered fundamental computing resources such as processing (CPU), memory, storage and networking allowing the user to deploy arbitrary software. Through an hypervisor the virtualization of the computational resources is made available to the user in form of virtual machines (VMs). Resource

virtualization is the key feature of this model allowing the user to run their own operating system, which are running on top of the infrastructure offered.

2.1.4 How Applications are Deployed in Cloud Environment

Since in this work we will focus on IaaS, a better explanation of what are VMs, and how cloud providers manage them, is necessary. VMs, illustrated in Figure 2.1 are hosted in physical machines with the help of a hypervisor. There can be several VMs running at the same time in the same physical machine, however, one VM is not aware of the existence of the others. Several VMs can be running in the same machine without the user knowledge. VMs have access to virtualized resources which can be CPU, Memory, Disk and so on, where these resources have the same functionality as the physical hardware, and the hypervisor is the responsible for managing them. The user applications are then deployed on the selected VMs.

Each cloud provider dispose of a catalog with multiple VMs configurations and a given price for it. This aspect will be further discussed when we present Section 3.6 where we discuss the cost model. The key aspect is that the customer is the one that should identify the most adapted VM types for his application, which in many cases might lead to over-provisioned resources. To benefit from this aspect, cloud providers use something called overcommitment of resources, which means that a given resource might be shared among several users in values bigger than the ones available (i.e. 4GB of memory might be shared among 5 VMs each requesting 1 GB). Overcommitment is possible and profitable for cloud providers due to the fact that the majority of cloud users does not exceeds 80% the utilization of resources reserved [109].

2.2 Type of Tasks in Cloud Data Centers

Tasks in cloud data centers are considered here as an application of a given type that the user submitted to be executed. We consider that each task is isolated in a virtual machine and can be directly deployed on the requested hardware. We classify our applications in two different types:

- Batch jobs [34, 105]: This type of task does not run interactively. It is rather submitted to a batch scheduler that can thus delay it in time, assign it to different hosts or control the voltage and frequency of its processor to limit the power it consumes. A batch job is characterized

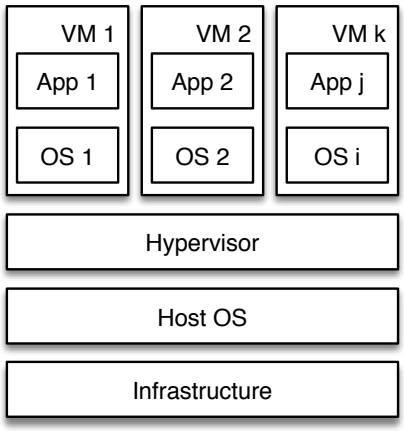


Figure 2.1: Virtualization layers of computing resources in cloud data centers.

by its submission date and may have a due date or a deadline indicating when it must be finished. A due date can be violated but then generates a quality of service penalty. A deadline can not be violated, if the job has not finished before it is canceled.

- Services [114]: Is an application that handles requests issued by interactive clients. A service is characterized by a service rate, number of handled requests per second and consequent consumed resources to process these requests. Power consumption limitation can be achieved by decreasing the service rate. A service may only run in a virtual machine. In that case, limiting the resources allocated to the virtual machine may decrease the service energy consumption and migrating virtual machines to group them may be a way to power down some servers.

The usage of both task models aim to represent the different type of tasks found in datacenters as demonstrated by Jia et al. [66]. These task models will be utilized later to classify the works from the literature. A more detailed model of the tasks considered in this work is presented in Section 3.2 as well as how each task execution is translated into a given power consumption in Subsection 3.1.1.

2.3 Energy Consumption in Cloud Data Centers

With the rapid growth in data centers energy consumption [69, 72, 78] the reduction of energy usage became a major concern. Several techniques are pre-

sented to save energy [97] in this context, and opportunities and challenges [36] that are still open. In this section we present what are the sources of energy consumption in data centers and some of these research initiatives.

When looking at large-scale distributed systems, several energy-saving techniques can be used, as presented in [97], illustrated in the taxonomy of Figure 2.2. In this Figure the author presents several leverages that can be utilized to improve the energy consumption in data centers. Another important factor that should be highlighted is cooling consumption, which can represent around 30% and 40% of the total energy consumption of a data center. Nevertheless, thermal management has been widely studied in the literature [27, 102, 116] and we will not focus on it in this work. The thermal aspects can be explored lated as independent problems. Another way to improve this factor is to utilize more energy efficient hardware in the data center, as the technology evolves.

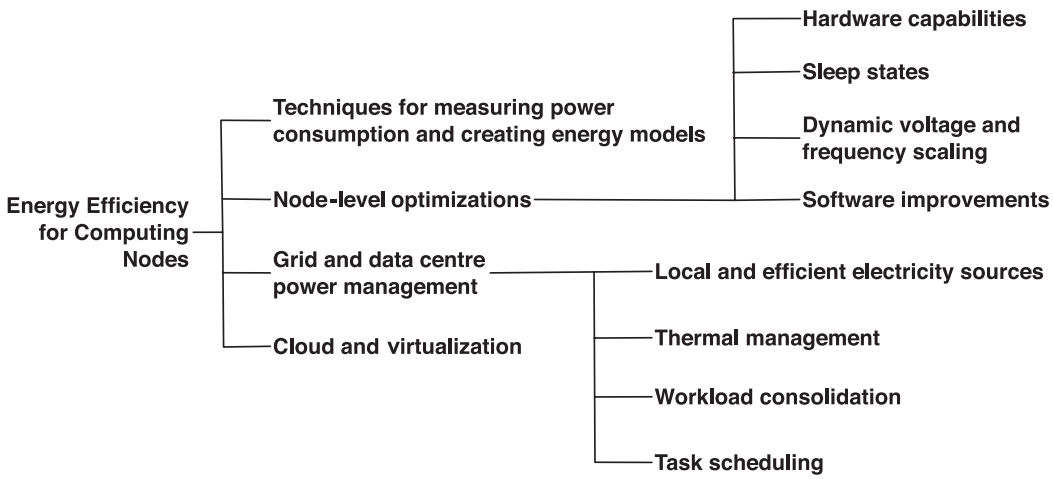


Figure 2.2: Taxonomy of techniques for improving energy efficiency in large-scale distributed systems from [97].

Considering the aforementioned taxonomy, in this work we profit from the utilization of cloud computing and virtualization, previously detailed, where the energy saving comes from resources sharing and overcommitment to reduce the wasted energy on over-provisioned VMs. We also utilize Dynamic Voltage and Frequency Scaling (DVFS), which consists in reducing the frequency of a given processor, consequently reducing its voltage and energy consumption. Another technique explored is the optimization of task scheduling, which will be detailed in Section 2.6. Finally, another technique that is utilized is switch on/off the computing servers, here also called computing nodes.

Furthermore, regarding the energy consumption of a node we can separate

it in two parts named statics and dynamic where the total power consumption is the sum of these two factors. In the static part we have what would be the power consumption of a node that has been switched on and is in a steady state, without any workload. This static power comes from components such as motherboard, fans, idle disks and so on. The other part (dynamic) comes from the actual usage/processing of a given load and increases according to the usage of CPU and memory for instance. In Figure 2.3 in the green block, we present an illustration of how this power consumption works. On the top, in green, we have the power consumption due to the execution of each one of the 3 phases of the load presented in the bottom, this allows the visualization of the two different factors. Furthermore we have some part P(idle) that is consumed without processing any task. The power model utilized is presented in details in Section 3.1.1.

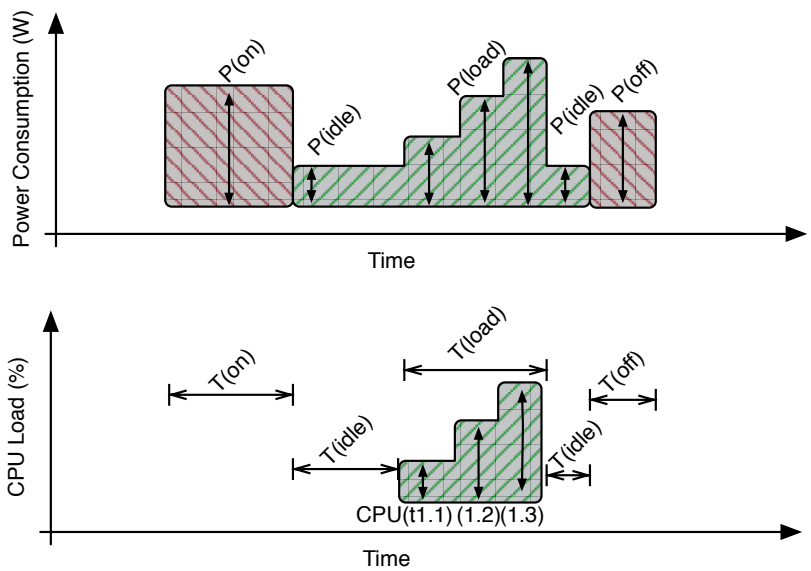


Figure 2.3: Switch on, off, idle and dynamic power consumption illustration.

Other than the energy that is consumed by the task execution and idle, we also have the energy consumed by switching on ($P(on) \times T(on)$) and off a node ($P(off) \times T(off)$). This is represented in Figure 2.3 in the two red blocks, where we have a power peak to switch on and off the node.

2.4 Renewable Energy in Cloud Data Centers

Other than trying to reduce the use of the overall energy in cloud data centers, one can also utilize local renewable resources (on-site generation) and/or renewable energy available on the grid (off-site generation) [106]. The selected approach depends on the availability of these energy sources (solar irradiance and wind speed for instance) and the cost to deploy the intermittent infrastructure. Studies can be found on what are the alternatives that are more adapted for each region [108] and the cost to deploy it [94].

Using green energy in data center also implies to change the data center outside electrical infrastructure. Regarding this aspect, we present in Figure 2.4 an illustration on how renewable energy could be linked in a 2N infrastructure (2 power supply circuits), as utilized in [53]. In this case, more than the connection between the Power Distribution Unit (PDU) to the Uninterruptible power supply (UPS) which is finally connected to the grid and power generators, we have another branch in which the UPS is connected to renewable energy sources (on or off site). We also have an ATS (Automatic Transfer Switch), which means that the systems can hot swap between each other in case of failure.

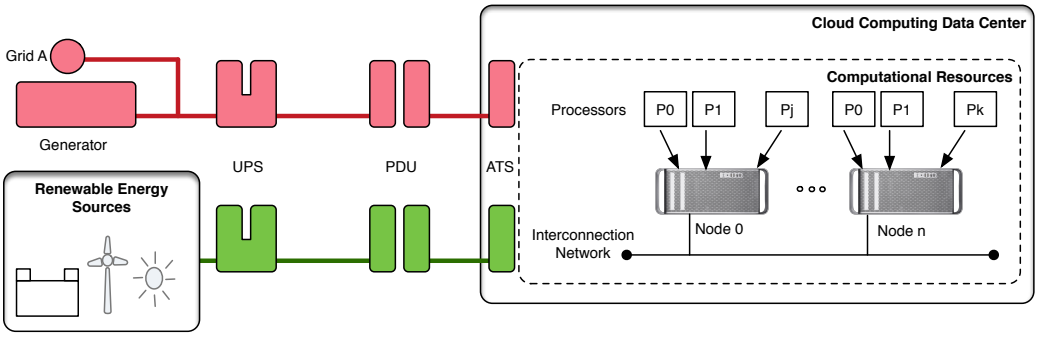


Figure 2.4: Data center electrical infrastructure connected to renewable energy sources.

Nevertheless, most of the IT companies that have large scale distributed systems and advertise them as 100% renewable, such as Google[1] since 2017, generate the energy off-site. This means that several other transmission (electrical grid might not able to support) and distribution costs are involved, as well as losses that occur in the transmission [128]. If the cloud provider desires greater control on energy supply, the most recommended approach is on-site generation. In this case the provider avoids the incurred transmission and distribution costs,

[1] https://sustainability.google/projects/announcement-100/

also reducing the system losses [128]. The main drawback is that this type of installation requires large financial investment, which might not be possible for some small IT companies [94]. Despite of the chosen method, one of the electrical branches is normally connected to a reliable (not intermittent) energy source[118].

In Section 2.5 we present how the infrastructure considered in this work is structured and in Section 3.5 how we abstract it to focus on the IT optimization.

2.5 DataZERO Project Context

The DataZERO project [101] investigates ways to operate a data center with renewable energy sources on site. As these sources are intermittent [1], there is a need to optimize the IT load to the energy availability. Also contrarily, it is necessary to optimize the energy production to the incoming IT load.

From the electrical point of view, redundancy is a main feature for data centers. Most of them rely in the classical 2N [98] and variations such as N+1 or 2N+1, since data centers have to guarantee QoS to the clients in order to not give money/credit back. N+1 systems have 1 additional UPS module to handle an adequate supply of power. This system however is not fully redundant and can still fail, since it runs on a common circuit. The 2N approach however consists in connecting the data center to two distinct power systems, each containing N components (i.e. double the whole electrical infrastructure with no single points of failure). Additional variations such as 2N+1 can also be found, consisting in two distinct power systems (2N) and one additional UPS (+1).

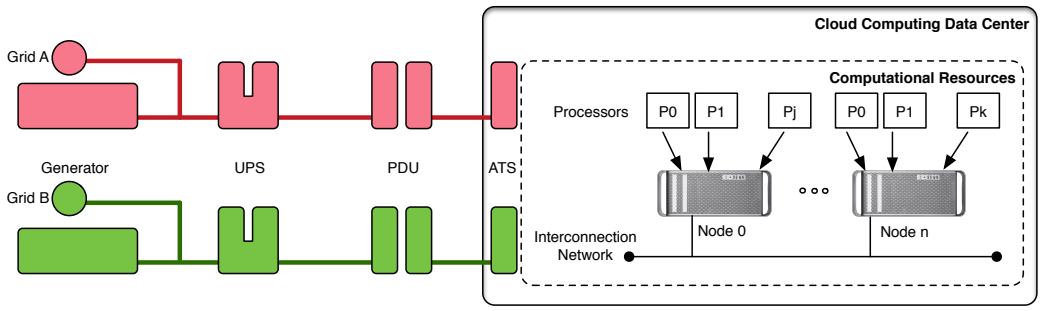


Figure 2.5: Classical data center architecture with 2N

In Figure 2.5 we illustrate a classical 2N infrastructure where we have the machines connected to an ATS, which means that the system can hot swap between each other in case of failure. Next we see the PDU connected to the UPS which is finally connected to the grid and power generators.

Some other projects such as GreenDataNet [53] aim to integrate renewable production in the data center. In this case the authors utilize GDN UPS (Green Data Net - Uninterruptible Power Supply), presented in Figure 2.4, to secure the green supply. The infrastructure consists in an N+1 where one of the branches of the electrical power would be completely composed by green energy. In our case we consider that none of the branches will be connected to the grid. DataZERO proposes a "Breakthrough" data center architecture [101]. The infrastructure is illustrated in Figure 2.6 where instead of a regular UPS we have a green energy sources only connection. This also brings several new challenges in how to handle the intermittent nature of these energy sources.

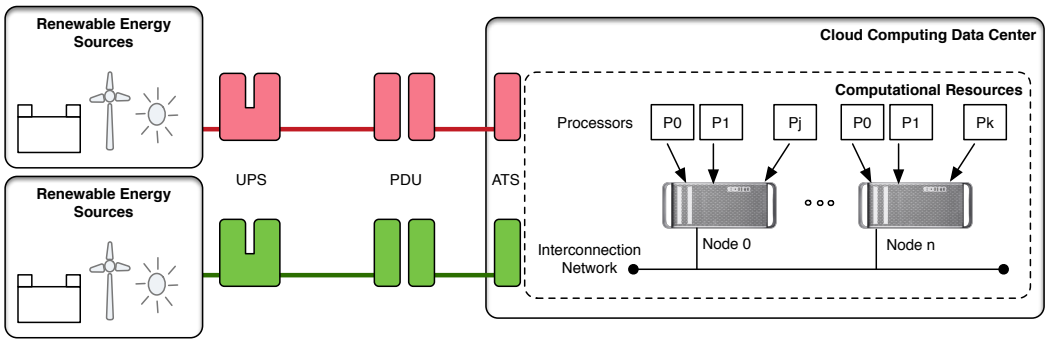


Figure 2.6: Breakthrough architecture with 2N redundancy (Green + Green).

In DataZERO project, the approach considers that IT part and the electrical part are able to optimize and handle their operations individually, negotiating between each other when necessary. In the general framework of DataZERO, the project introduces three decision modules which have to cooperate, namely Information Technology Decision Module (ITDM), Power Decision Module (PDM) and Negotiation Module (NM).

Figure 2.7 gives an overview of the information exchanged between the three decision modules to obtain a trade-off between the consumption needed by the IT part and the power production achievable by the electrical side. The DM are considered as black boxes and the communication is only based on power envelopes. The concept of *power envelopes* is used for representing either the power required for the IT part or the power proposed by the electrical part at any time during a given time horizon. A power envelope (here also called power curve or power constraint) is a set of power values for the different time steps of a given time horizon. In both cases (ITDM and PDM), a profile is associated to a utility value, which can be the profit/cost of utilizing it.

The roles of each module is defined as follow:

- The Power Decision Module is in charge of managing the power sources

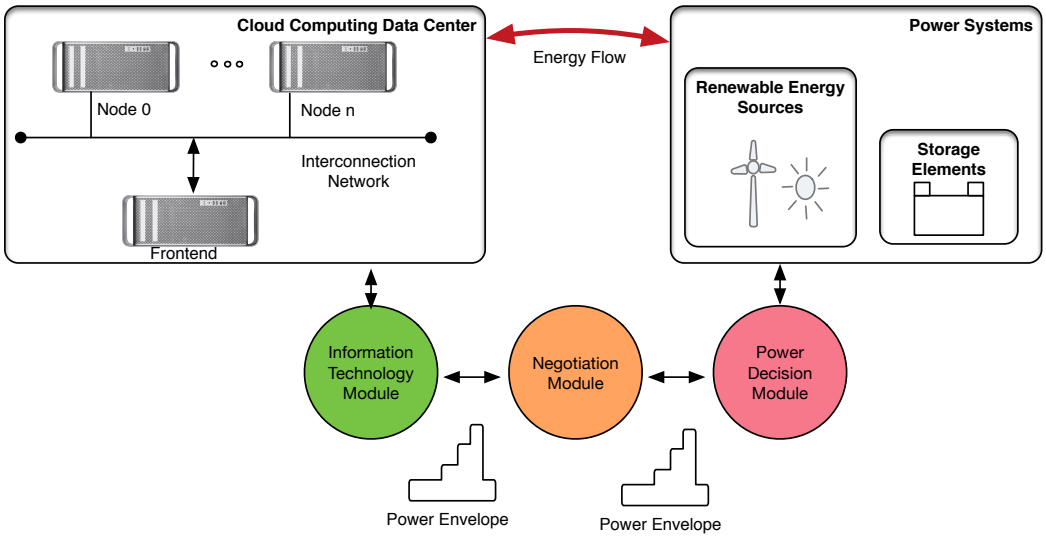


Figure 2.7: DataZERO interaction overview.

and storage. Based on the power system state, it proposes power envelopes to the Negotiation Module. Once a profile is negotiated, the PDM is in charge of deciding which power source or storage to engage.

- The IT Decision Module is in charge of managing the power needs of the IT resources. It uses the power envelopes defined in the application model to compute a global power envelope that is then used as proposition for the Negotiation Module. Also, the ITDM uses the profile negotiated with the Negotiation Module to decide which application to run, and which IT equipment to power on or off, depending on the constraints and resources available. The basis for this problem are explained in the next Section.

- The Negotiation Module is in charge of making the power envelopes matching between the ITDM and the PDM. The choice is based on the matching of the profiles and a utility value which is associated to each profile. This value can be either QoS or the profit/cost obtained to implement it.

This thesis focus specifically in the IT Decision Module inside the DataZERO project. In this sense, we will not go into details of the other modules, and we assume that cases of failure in the power or negotiation will be handled by the corresponding modules.

2.6 Optimization Strategies for Task Scheduling in Cloud Computing - ITDM

Regarding the aforementioned tasks and the DataZERO infrastructure, another problem that rises is how to choose when and on which resources to deploy the tasks. The scheduling of these tasks can be performed either manually or in an automated way. In large scenarios, however, as the number of possible configurations grow it becomes difficult for an application manager to take into account all application constraints manually.

The problem of placing an application in an infrastructure is NP-Complete [2, 68], meaning that there is not a known polynomial time algorithm that can solve it optimally. Calculating the scheduling can be an issue when application and infrastructure become larger. In this thesis we consider that there are hardware constraints (i.e. amount of CPU, number of processing elements available, amount of memory), which should match the task requirements, as presented in Figure 2.8. In the presented case, Task 1 would not be able to start at that moment since it requires 100% of two processing elements, and a new node should be switched on (if there is enough power).

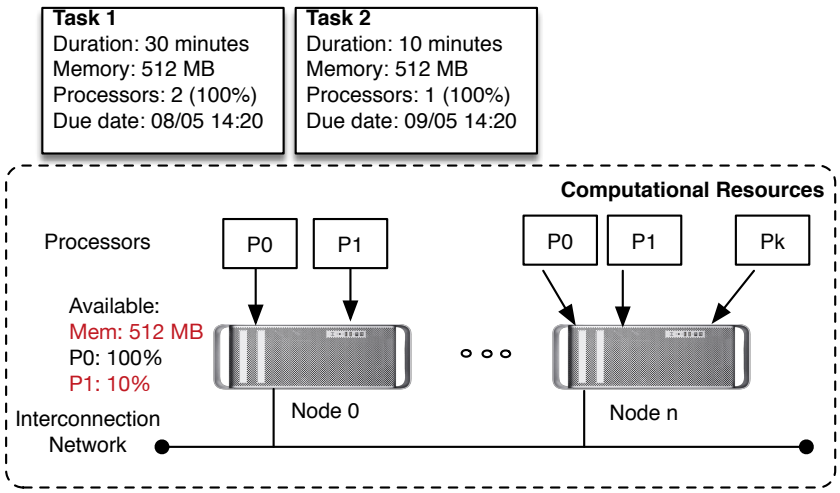


Figure 2.8: Tasks scheduling in node considering resources constraint.

Considering the DataZERO scenario, we also have the power curve (renewable energy produced) as constraint. In Figure 2.9 we display a possible optimization where the tasks are converted into the amount of power they would consume in a given hardware under a given configuration, and placed under the renewable energy power curve. Considering the amount of tasks and nodes available, and depending on what node is chosen, switch on/off, the power

consumption changes, the number of possible infrastructure configurations increase considerably. In those cases, performing placements manually may not be an option and automation emerges as a necessity.

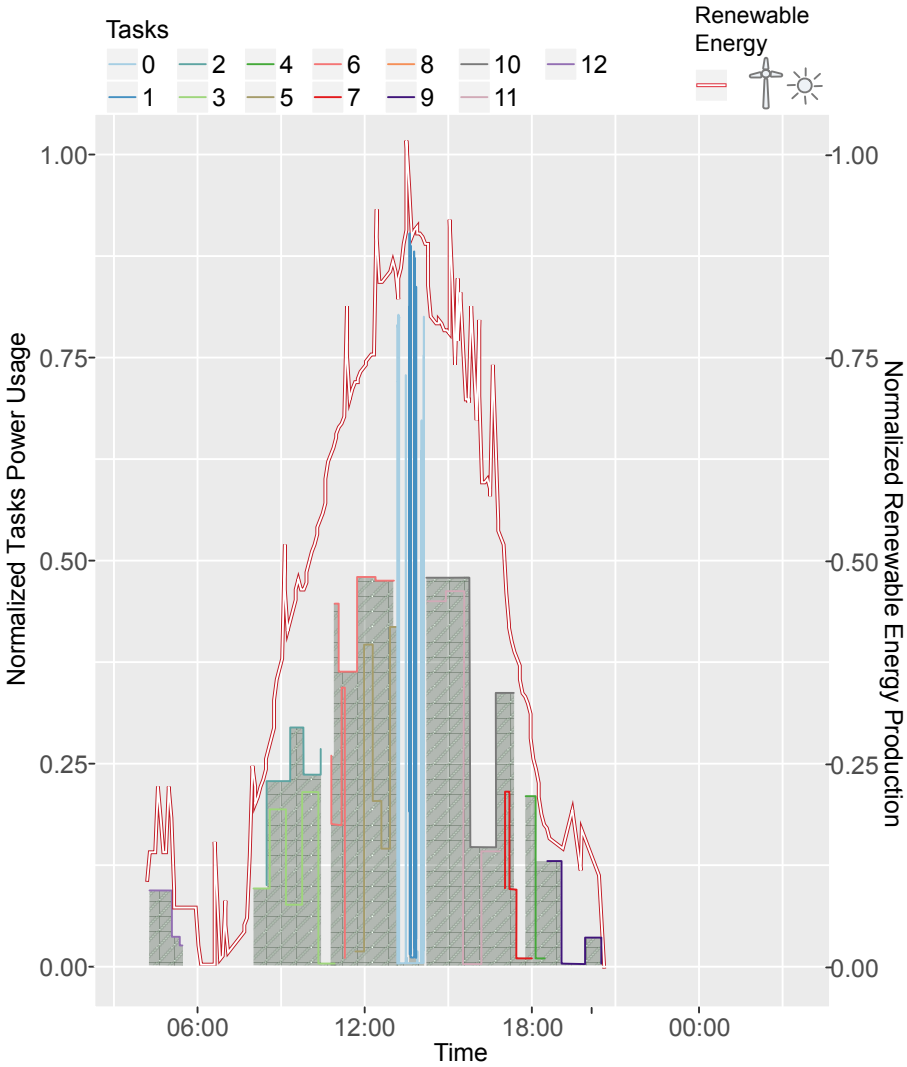


Figure 2.9: Task power consumption and renewable energy production after optimization in a data center powered only by renewable energy.

We can separate the scheduling problems in two types: (i) offline scheduling where the amount of tasks, resources consumption and energy available are known for the evaluated time interval; and (ii) online scheduling, where nothing is known in the beginning of the time interval, so tasks may arrive at any moment, the amount of energy available can change, as well as the resources

consumed by a given task. In this thesis we start by an offline approach and in the end we tackle the online approach utilizing reactive actions.

Regarding the ways to automate the scheduling we have several optimization methods which can be divided in three groups: (i) exact algorithms; (ii) greedy heuristics; and (iii) meta-heuristics.

The exact approaches aim at calculating optimal solutions for a given problem. The most common exact approach is to use linear programming (integer or mixed) [58] through solvers such as CPLEX[62]. The main advantages are the formal modeling of a problem, characterization of a solution and the comparison with approximate methods on small problems. Nevertheless, the disadvantages are the slow resolution on large problems, which makes the solution of realistic IT problems unfeasible. For these reasons, in this work we focus on greedy heuristics and meta-heuristics.

Heuristics are a way to solve a problem, in some cases without guaranteeing the optimal solution. It is commonly used to calculate approximate solutions to problems that do not have polynomial time algorithms to them and widely utilized in scheduling approaches [3, 8, 16, 49, 67, 83, 84, 86]. This method is usually capable of computing solutions faster than exact or meta-heuristic approaches.

Meta-heuristics are also utilized to approximate solutions to optimization problems without known algorithm that provides optimal solution in polynomial time. The main characteristics is that they allow to produce a large number of adapted solutions. It is also possible to approach a local minimum starting from an existing solution, and have a low cost of implementation. There are many meta-heuristics in the state of the art [18] such as Genetic Algorithms (GA), Ant Colony, Particle Swarm and Simulated Annealing. In this thesis, we are specially interested in Genetic Algorithms, which allow us to provide several solutions for the same power envelope, and send to the other modules in DataZERO project.

Genetic Algorithms [115] mimics the natural evolution, where we have as main parameters a population, mutation, cross-over and generations. A population is a set of individuals where each individual has several chromosomes. The chromosomes represent a possible solution for a given problem. To this solution we assign a score (fitness) according to the objective function. Over several generations these individuals will have cross-over (exchange chromosomes) among each other generating new individuals (offspring) that will repeat the process. Sometimes mutations can also occur, generating variations that might not present in none of the individuals. For each generation a subset of the best ranked individuals are kept. For each generation, the new individuals tends to be better than the individuals from the previous generations.

2.7 Literature Review of Task Scheduling in Cloud Data Centers With Renewable Energies

In this section we present research works that tackle the issues related to data center powered by renewable energy, and the problems from the IT point of view. We also highlight that the focus is on works in which the data center is powered not only by the power grid, but have some sort of renewable energy as well. To classify these works we present them in chronological order, separated in three categories by the type of task, which have been previously defined: (i) Batches; (ii) Services and (iii) Mixed (both batch and services).

2.7.1 Batch Task Scheduling

Among the works that focus on batch workload Berral et al. [12], Goudarzi and Pedram [51] focus on scheduling of batch tasks in geographically distributed data centers. Goudarzi and Pedram [51] focus on the load balancing for batch jobs with dependencies in heterogeneous data centers. The authors aim at reducing the cost considering renewable energy sources and energy pricing variations. The proposed approach is an online greedy scheduler where the resources and peak power capacity in each data center is limited. The algorithm obtained an operational cost decrease up to 40%. Berral et al. [12] focus on studying the location and cost of provisioning several green data centers that provide a given level of renewable energy on-site (the rest comes from the grid). The authors then explore the placement of High Performance Computing (HPC) cloud tasks in this infrastructure using GreenNebula (no details on the scheduling policy are provided) with 9 VMs with a CPU-intensive synthetic application.

Klingert et al. [70] studied the operating of a RES-powered data center with workload coming from smart cities. They proposed a both technical and business related solution for managing the share of local RESs. To do that, the authors introduced adaptation strategies together with power management options. The sensor networks in a smart city collect data and store them in a data center. Based on current state of the power grid and the availability of RESs, the data center schedules the IT tasks, virtual machines and servers accordingly.

Several papers from the research of Goiri et al. also consider batch workloads. In Goiri et al. [49], the authors introduced an energy management algorithm for parallel task scheduling, considering photovoltaics and grid power. The scheduler aims to maximize the utilization of the renewable generation as well

as minimizing the cost of electricity consumption of the data center. The grid energy is deployed as the secondary power supply used when there is a lack of solar generation. Hence, the controller schedules the tasks to the low cost hours, in order to reduce the electricity cost. The simulation results are generated for two different systems: SLURM (scheduler for Linux) and MapReduce (scheduler for Hadoop). In Nesmachnow et al. [95], the authors proposed a two-level control method by solving multi-objective optimization problem in a data center. At the upper level, while the algorithm is reducing the deviation between reference and actual power envelopes, it increases the quality of the services at the bottom-level using the received power envelope. The reference consumption profile is predetermined by taking into account the renewable generation (i.e, high at the mid-day due to solar power) near the desired temperature value. Using the Pareto-front optimal solution, the developed control algorithm finds the best control strategy that maximizes the green-energy consumption.

Some other studies from Lei et al. also considered batch workload. In Lei et al. [79], the authors developed a task scheduling strategy for a green data center powered by both electrical grid and renewable energy. In the data center, the green energies are from the wind and solar electrical sources. The provided energy from the main grid is used by taking into account the renewable energy prediction and the dynamic electricity pricing of the grid. The proposed algorithm shows superiority and effectiveness compared to three other scheduling strategies, which focus on three other objectives: maximizing green consumption, reducing cost and greedy (first pick high generation, then low cost). Lei et al. [80] use Dynamic Voltage Frequency Scaling (DVFS) for handling IT consumption. DVFS is one of the most common energy management techniques on the IT part. It is usually applied when a slack period is detected, then the processor can run at lower frequency or operate on lower voltage, which decreases power consumption. The authors proposed a multi-objective co-evolutionary algorithm for scheduling tasks in data centers partially powered by renewable energy. Each chromosome represents a possible solution, and each gene assigns one task to a processor and sets its voltage. The GA operators shuffle the assigned processor and voltage parts of the genes through the chromosome, while the order of tasks remains fixed according to their index. The GA's objective is to match the workload with the renewable energy supply based on the prediction of renewable energy, with the aim of maximizing the utilization of renewable energy and minimizing the makespan of tasks and the total energy consumption.

Iturriaga and Nesmachnow [64] presented the approach of simultaneously scheduling the IT jobs, the states of servers and the cooling devices. The authors proposed a multi-objective method for scheduling energy consumption

in data centers. The data centers are powered by both traditional grid and renewable energy. The goal is to minimize the energy consumption, the energy consumption deviation from a reference profile, and the number of due date violated tasks. Based on a Pareto-oriented methodology, the authors proposed two multi-objective evolutionary algorithms to solve the problem. A greedy scheduling algorithm and a simulated annealing algorithm are also proposed, in addition to the evolutionary algorithms. Whereas the evolutionary algorithms schedules the server and cooling devices, the greedy heuristic schedules the IT jobs, and the simulated annealing algorithm is used as a post hoc optimization mechanism.

The research from Kassab et al. [67], [68] considers the problem of scheduling HPC independent tasks under power constraints in data centers powered by renewable energy. Several common standard scheduling techniques such as list scheduling algorithm and binary search technique were adapted to take power constraints into account. They were tested on a multi-core machine computational platform that is powered solely by renewable sources. Their approach does not aim to reduce the energy consumption, but rather to optimize typical common scheduling objectives such as minimizing the makespan and the total flow time, while respecting the power constraint. The amount of jobs that can be executed simultaneously at a certain time is limited by how much power is being produced by the renewable sources, which naturally varies through time. Their results show that the list scheduling algorithms that they adapted to this problem give fast and good solutions. The works of the previously presented authors [67, 68] have been developed in the context of DataZERO project so, do not consider connection to the grid.

Courchelle et al. [33] proposed a priority-based scheduling, taking into account the forecast renewable energy and batteries. The authors also introduced a sizing methodology, in which the amount of solar panel and battery are formulated as a function of the expected workload. The scheduling uses genetic algorithm to propose task allocation, using the available renewable resources and storage capacity. The overall decision on the electrical infrastructure depends on the IT scheduling decision under the constraint of available renewable energy. In Grange et al. [52] the authors present an approach for scheduling batch jobs with due date constraints, which takes into account the availability of the renewable energy to reduce the need of brown energy and therefore running cost. The scheduling algorithm proposed is agnostic of the electrical infrastructure. While the other works try to solve their optimization problem in a fully centralized way, we propose instead to have two parts communicating with each other. Each part (electrical and IT) does domain-specific optimizations without knowledge of the model and characteristics of the other part.

The scheduler, contrary to other approaches, never deals directly with the model of the electrical infrastructure, but instead uses partial and abstracted information.

2.7.2 Services Task Scheduling

Among the works that focus on service workload, Sharma et al. [112] is the only work that does not consider connection to the grid. The author focus mainly on deactivating server when there is no energy, which can be useful for some web applications (even though it neglects the users experience), but not realistic on the vast majority of applications running in cloud platforms. In Ghamkhari and Mohsenian-Rad [43] considered rate allocation in green data centers, which has local renewable power generation. The authors considered a trade-off between maximizing data centers' profit and minimizing their energy expenditure. The profit is defined to be the difference between revenue and cost. The decision variable of the proposed optimization model is the service rate at which the service requests are handled by the servers. The model considers the factors of local renewable energy availability and the stochastic nature of workload in data centers. In Paul et al. [100], the workloads are also services, but Paul et al. considered them as deferrable deadline-oriented workloads. The problem is viewed from the perspective of demand response (change in the consumption to better match the demand from the supply), which is a spatial/geographic load balancing problem. The authors proposed a straightforward strategy for shifting in time the deadline of IT jobs, in order to match the renewable energy availability.

Also considering the workload of services, the research in Wang et al. [125] uses the approach of virtual machine migration for managing a sustainable data center powered by renewable energy. The authors propose a green-aware power management strategy, considering the energy consumption of both IT functional devices and cooling devices. The objective is to use green energy sufficiently and maintain the discharged power at an acceptable level. An optimization problem of virtual machine migration is formulated and solved by combining heuristic and statistical searching. The proposed strategy is valuated by dynamically migrating VMs across the physical machines. This strategy shows the advantages over other approaches because it utilizes more efficiently green energy.

Since both renewable energy availability and the workload are expected to differ from one data center to another, migrating some VMs/containers/tasks from a data center with low local renewable energy production and high workload demand towards another data center with extra renewable energy to spare, reduces the need for brown energy. The initial works that explored the

usage of Geographical Load Balance (GLB) and renewable energy are Liu et al. [85] and Lin et al. [84]. Liu et al. [85] investigate the feasibility of powering cloud data centers using renewable energy. The study focus on geographical load balancing, and the optimal mix of renewable energy using a concept called "follow the renewables" in which the workload is migrated among data centers to improve the renewable energy usage. Lin et al. [84] present an extension of the previous work with online algorithms to exploit the geographically distributed data centers and their renewable energy availability from solar panels and wind turbines.

In [55], Gu et al. deal with the problem of IT job management by dynamically distributing the computational requests to the RES-powered data centers. The decision variables determine the number of requests to be assigned to each data center for each time slot. The authors model the problem as a constraint optimization problem. The objective is to minimize carbon emissions under a fixed electricity budget. The constraints are (1) the processing time, (2) the electricity budget in each time slot, (3) the intermittent energy sources, and (4) the maximal number of servers in each data center. Another research from Gu et al. [56] considers data centers powered by wind, solar and brown energy. The authors also rely on storage devices to buy energy at low prices (time-varying and location-varying electricity prices) or store renewable energy surplus. The work focuses on two optimization problems: the first, to respect the QoS demand and minimize the energy cost; and the second to minimize the total carbon emissions given a budget for energy. The problems are formulated as mixed integer linear programming and the results show that the usage of storage devices can reduce more than 25% for both energy cost and carbon emissions. Paul et al. [99] focus on online algorithms to minimize cost, exploiting the variations in the grid prices and the renewable energy utilization, also in a scenario with several data centers. The authors also explore the prices volatility, leveraging from contracts in the forward electricity market anticipating the amount of energy that should be bought. The validation consists of simulations with real data traces with two variations of algorithms to evaluate also the server switching.

More recent works focus also on maximizing the renewable energy usage, as in Toosi et al. [119] and Atefeh et al. [7]. Toosi et al. [119] consider GLB to reduce energy cost and to maximize renewable energy utilization where the main difference is focused on practical considerations and the evaluation in a real environment (Grid5000) using traces of English Wikipedia traffic, associated with renewable energy and electricity prices traces. The authors consider a global and a local load balancing to distribute the load among the different data centers, the policies tested are weighted round robin and another proposed

by Le et al. [77]. Results showed that the policy is able to reduce the cost by 22% and 8% and brown energy by 17% and 7% in comparison between round robin and Le et al. [77] while maintaining the average response time. Atefeh et al. [7] consider VM migration between data centers to maximize renewable energy (solar/wind) consumption, in this case for data centers located within the same region with a simplified model (high level view of the data center, not as machines, processors...). The authors present three different approaches: (i) the optimal offline cost for using the grid; (ii) the competitive ratio (price from buying energy from the grid) for an online deterministic algorithm, without any future knowledge of renewable energy; and (iii) a future-aware online algorithm with a look-ahead window and limited knowledge of a determined window-size of solar and wind energy.

Other works such as Zhang et al. [129] and Deng et al. [37] focus on different aspects, or metrics. Zhang et al. [129], approach the problem trying to minimize the brown energy usage but also focusing on the network, more specifically in Elastic Optical Networks (EONs). The authors use virtual machine migration according to the availability of renewable energy. They propose two heuristics for renewable energy-aware VM migration that take into account the influence of the network capacity, saving up to 31% the brown energy usage. Deng et al. [37] studied on workload dispatching at geographically distributed cloud data centers considering multiple electrical resources (power grid, energy storage system, solar and wind powers). The presented work focuses on minimizing the eco-aware energy cost, different from the normal cost, by assigning an eco-factor parameter to each type of power supply. Eco-factor refers to a level that indicates the green degree of the utilized energy from every supply. The parameter is determined according to the environmental impact of each supply during the energy production (grid power) and equipment manufacturing (green resources). The presented eco-aware algorithm is compared with four scenarios: price first with/without RES, service quality first with/without RES. The proposed method showed better cost reduction of 20% cost reduction.

In more recent works such as Li et al. [83] and Aujla et al. [8], the authors use the same concepts but introduce the usage of containers and edge computing. Li et al. [83] propose to leverage renewable energy production of edge nodes. The authors proposed an analytic model for deciding whether to offload computation from the objects to the edge or to the Cloud according to the renewable energy availability and QoS requirements. The works rely heavily on an on-site renewable energy production and storage at the edge nodes. Aujla et al. [8] propose a renewable energy-aware job classification and scheduling scheme using Container as-a-Service (CoaaS) for data Geo-distributed data centers. In the proposed scheme, the workloads are transferred to the DC which has

sufficient amount of renewable energy available at the moment or in some cases it draws energy from the grid if green not sufficient. The authors perform this task using a greedy algorithm to select the hosts and containers to schedule and consolidate, where the aim is to minimize the energy usage from the grid while maintaining the QoS. The proposed scheme is evaluated using Google workload traces, and indicates an energy saving between 10.55% and 28% in comparison to the existing schemes of its category.

2.7.3 Mixed Workload Scheduling

Here we present the works that consider the mixed workload where we have both batch and services. Aksanli et al. [3] propose an adaptive data center job scheduler which also utilizes short term prediction but in this case of solar and wind energy production. The aim of the scheduler is to reduce the number of canceled or violated jobs, and improve the efficiency of the green energy usage. The authors assume that each server has one web services request queue (load based on Rubis), and one or more slots to execute batch jobs. Services start execution whenever there are available computing resources and the system provides enough brown energy to maintain these services.

The research from Goiri et al. [48], Goiri et al. [46] focuses on building a research platform to study green data centers. The authors introduced Parasol, a prototype of a green data center with solar energy, batteries, and net metering. Additionally, the authors introduced GreenSwitch, a dynamic scheduler for workload and energy sources. The main contributions of this research are (1) the analysis of the main trade-offs in data centers powered by solar and/or wind energy, (2) the design of the Parasol platform and of the GreenSwitch system. The considered trade-offs are *grid-centric approach and self-generation approach*, *space and cost of solar energy*, and *space and cost of wind energy*. The GreenSwitch in [46] tries to minimize the cost of grid energy, with regard to the workload characteristics and the battery lifetime. The real experiments with Parasol and GreenSwitch show that an intelligent control of IT workload and energy source can help to reduce operation cost significantly. However, the renewable energy source in Goiri et al. [48] and Goiri et al. [46] only considers solar panel. Moreover, the IT scheduling in those papers is limited; specifically, the authors use scheduling algorithm to select which energy source to use (renewable, battery, and/or grid), and choose the storage medium (battery or grid) at each time period.

In Liu et al. [86], the authors also considered both service and batch workload. The authors provide intensive analysis about the potentials of demand response in data centers, and propose that prediction-based pricing is an appealing design for demand response management. The workload in Cioara et al. [30]

is not clearly stated to be batch and service, but instead, "real-time" and "delay-tolerant". The authors considered a straight-forward approach that shifts energy demand of the data center in time, from time intervals with low RES due to weather conditions, to time intervals with high predicted RES. The objective is to maximize the usage of locally produced renewable energy. The shifting is realized based on the flexibility mechanism for components such as electrical cooling system, IT workload, and diesel generators. The in-lab simulation results show that the proposed approach increases the usage of renewable energy by 12%, thanks to the shifting of the power demand to the time intervals with high renewable energy supply.

In Beldiceanu et al. [11]. the authors considered simultaneously (i) IT jobs' spatial scheduling (on the appropriate servers), (ii) IT jobs temporary scheduling (over time, with jobs that can be planed in the future), VM suspend/resume switching, and VM migration. A proposed heuristic algorithm schedules jobs spatially, temporally, and a VM controlling mechanism manages the VM switching and migration. The authors introduced EpoCloud, a prototype that optimizes the energy consumption of mono-site data centers, which are powered by the regular grid and RES. The objective is to find the best trade-off between energy cost and QoS degradation.

In Li et al. [82], the authors introduced two approaches to improve the utilization of renewable energy in a small/medium-sized data center. The first approach is an opportunistic scheduling that run more jobs when renewable energy is available. In the second approach the renewable energy surplus are stored and used later when renewable energy is scarce. The objective is to maximize the utilization of on-site renewable energy. The experiments with real-world job workloads and solar energy traces show that the proposed approaches can reduce the demand for energy storage devices' capacity. However, the proposed management of power sources is simplified. This management mainly focuses on the control of energy storage devices, considering their characteristics, e.g., battery charging rate limit, battery depth-of-discharge.

2.7.4 Discussion of Literature and Classification

In Table 2.1 we present all the evaluated works considering 5 criteria. First we present the objective of the approach, followed by the electrical infrastructure that the authors utilize, the workload type (batch, service or mixed). We then present the evaluation method, whether it is a simulation or real infrastructure, and the optimization method.

Table 2.1: Summary of characteristics for existing renewable data center scheduling works.

Approach	Objectives	Electrical infrastructure	Workload type	Evaluation method	Method type
Aksanli et al. [3]	Max. job completion time & number of task run	Photovoltaic, wind turbine & Grid	Batch and service	Simulation	Greedy heuristics
Sharma et al. [112]	Min. performance degradation	Photovoltaic & wind turbine	Service	Testbed	Greedy heuristics
Liu et al. [85]	Minimize Cost of System (Energy & Violations)	Photovoltaic, Wind Turbine & Grid	Service	Simulation	Greedy Heuristics
Lin et al. [84]	Minimize Cost of System (Energy & Violations)	Photovoltaic, Wind Turbine & Grid	Service	Simulation	Greedy Heuristics
Goudarzi and Pedram [51]	Minimize Operation Cost (Energy, Violations & Migration)	Renewable Energy (Unspecified)& Grid	Batch	Simulation	Greedy Heuristics
Ghamkhari and Mohsenian-Rad [43]	Max. profit (revenue - cost)	Wind turbine & Grid	Service	Simulation	Optimization
Liu et al. [86]	Min. user cost (the frequency of violations of voltage constraints)	Photovoltaics & grid	Batch and service	Simulation	Greedy Heuristics
Berral et al. [12]	Minimize Building Cost & Minimize Brown Energy Usage	Photovoltaic, Wind Turbine, Battery & Grid	HPC Batch	Simulation	Not Specified

Approach	Objectives	Electrical infrastructure	Workload type	Evaluation method	Method type
Wang et al. [125]	Max. the utilization of green energy (max. profit, min. energy & operational cost)	Photovoltatic & grid	Service	Simulation	Heuristic and genetic algorithm
Klingert et al. [70]	Max renewable energy	Renewable (unspecified) & Grid	Batch	Simulation	Not Specified (position paper)
Goiri et al. [49]	Max. green energy consumption	Photovoltaics & Grid	Batch	Testbed	Greedy Heuristics
Nesmachnow et al. [95]	Multi obj.: min. deviation to renewable power, min deviation to temperature, min. deadline violation	Unspecified	Batch	Simulation	Multi-objective evolutionary algorithm
Lei et al. [79]	Max. utilization of renewable energy & min. the total cost of energy	Photovoltaic, Grid with dynamic cost	Batch	Simulation	Greedy Heuristic
Dupont et al. [39]	Max. utilization of renewable energy	Solar Panels & Grid	Batch and service	Testbed	Not Specified
Zhang et al. [129]	Minimize Brown Energy Usage	Photovoltaic, Wind Turbine & Grid	Service	Simulation	Greedy Heuristics
Gu et al. [55]	Minimize Carbon Emissions Under a Fixed Electricity Budget	Photovoltaic, Wind Turbine & Grid	Service	Simulation	Mixed Integer Linear Programming

Approach	Objectives	Electrical infrastructure	Workload type	Evaluation method	Method type
Lei et al. [80]	Multi obj.: max. renewable energy, min. total energy, min makespan, max. QoS	Photovotaic, Grid	Batch	Simulation	Multi-objective evolutionary algorithm
Iturriaga and Nesmachnow [64]	Multi obj.: min. deviation to renewable power, min. energy cost, min. due date violation	Renewable (unspecified) & Grid	Batch	Simulation	Multi-objective evolutionary algorithm
Gu et al. [56]	Minimize Carbon Emissions & Minimize Energy Cost	Photovoltaic, Wind Turbine, Battery & Grid	Service	Simulation	Mixed Integer Linear Programming
Paul et al. [99]	Minimize Energy Cost	Renewable Energy (Unspecified) & Grid	Service	Simulation	Convex Optimization Solver
Deng et al. [37]	Minimize Eco-aware Energy Cost	Photovoltaic, Wind Turbine, Battery & Grid	Service	Simulation	Lyapunov Optimization
Toosi et al. [119]	Minimize Cost and Maximize Renewable Energy Utilization	Photovoltaic, Wind Turbine & Grid	Service	Testbed	Greedy Heuristics
Atefeh et al. [7]	Minimize Cost	Photovoltaic, Wind Turbine & Grid	Service	Simulation	Greedy Heuristics

Approach	Objectives	Electrical infrastructure	Workload type	Evaluation method	Method type
Li et al. [83]	Maximize Renewable Energy Usage	Photovoltaic, Battery & Grid	Service	Simulation	Greedy Heuristics
Paul et al. [100]	Max. renewable energy, min. total cost	Renewable (unspecified) & Grid with dynamic price	Batch and service	Simulation	Heuristic (Model Predictive Control)
Kassab et al. [67]	Min makespan & total flow time under power constraint	Renewable Energy Only Power Envelope	HPC Batch	Simulation	Greedy heuristics
Kassab et al. [68]	Min makespan & under power constraint	Renewable Energy Only Power Envelope	HPC Batch	Simulation	Greedy Heuristics & Genetic Algorithm
Courchelle et al. [33]	Min. non-renewable energy	Photovoltaic, grid, batteries	Batch w/o due date	Simulation	Greedy heuristic
Beldiceanu et al. [11]	Trade-off between energy cost and QoS degradation	Wind Turbines, Solar Panels & Grid	Batch and service	Simulation	Heuristic scheduling algorithm
Grange et al. [52]	Multi obj.: Min. non-renewable energy or total energy cost, max. QoS	Photovoltaic, grid (dynamic price)	Batch with individual due dates	Simulation	Greedy heurisitic
Aujla et al. [8]	Minimize Non Renewable Energy Consumption	Photovoltaic, Wind Turbine, Batteries & Grid	Service	Simulation	Greedy Heuristics

The only work outside the DataZERO project in which no connection to the grid is considered is Sharma et al. [112] which considers services (more specifically web services). The main problem is that the focus is only in deactivating servers (blinking) when there is a drop in the energy available. This would lead to a significant degradation in the QoS and would not be supported by most of batch applications where the execution would have to be restarted (assuming there is no checkpoint strategy).

Also, even though batch tasks and services are present in the daily workload of a cloud data center and more detailed models have been proposed and used in various contexts (*e.g.* evolving, moldable and malleable tasks, by Feitelson and Rudolph [41]), the majority of works in the literature for renewable energy still consider a simplified task modeling with rigid resources assignment.

All the remainder works assume connection to the grid, showing that there is a gap in approaches that would handle the intermittent nature of renewable energy sources. In the remained of this thesis, we will present approaches that consider scheduling simple tasks under a renewable energy only constraint. We then proceed to explore scheduling a mix of batch tasks and services in the same type of infrastructure but considering a variable workload (with different resources requirements over time) where the resources assigned can be smaller than requested.

In the best of our knowledge there is still no approaches for data centers only powered by renewable energy that considers both batch tasks and services scheduling with a more detailed application model/behavior, where the resources assigned can be smaller than the one requested. This possibility of delivering less resources than what was requested is fundamental when we consider an environment without connection to the power grid and intermittent power supply coming from renewable energy.

2.8 Conclusion

This chapter focused on presenting the fundamentals that are necessary for understanding the remained of this work, as well as the current state of the literature in data centers powered by renewable energy. Moreover, we presented the key concepts of cloud computing and who are the actors. This concept will be important to understand how the billing model works in the following chapters. We also introduced how renewable energy is integrated in cloud data centers and tasks, which will be detailed and formalized after. Finally we present a comprehensive list of works that approach the energy consumption optimization problem and what are the gaps that will be addressed in this work.

Chapter 3

Modelling, Data and Simulation

In this chapter we focus mainly in the model abstractions that will be utilized in the remainder of this work. We first present the IT infrastructure model, followed by the power consumption model. We then introduce the tasks model for each approach and their particularities. Next we introduce the data sources for the workload generators, which are utilized to generate the sets of tasks for experiments, and the details on how they work. Finally, we present the power production data sources for wind turbines and solar panels considered in this work and how all the aforementioned features are combined in the DCWoRMS data center simulator.

3.1 IT Infrastructure Modeling

In this section we describe the IT resources model and the power consumption. All the variables regarding the IT resources are summarized in Table 3.1. Our aim is to schedule tasks in the IT infrastructure, taking into account the power and resources available. To do so, we start by defining the infrastructure model.

We consider a platform of W hosts $\{\mathcal{H}_h\}_{h\in\{1,\ldots,W\}}$, such that each host $\mathcal{H}_h$ is composed of C_c Processing Elements (PE) equipped with DVFS and contains M_h memory. Here we generalize C_c on purpose, not defining as cores or as processors, in order to utilize the same IT model for both approaches. Moreover, in Chapter 4 we consider C_c as processors and in Chapter 5 C_c refers to each core.

Here we consider that each $PE_{h,c}$ can receive more than one task, as long as the PE usage (%) $c_{h,c}$ does not exceed 100%. In this case, we consider $c_{h,c}$ as the sum of all the load running in the core in a given instant. In the same way, the amount of memory M_h used is given by the sum of the memory (MB) of all tasks running on $\mathcal{H}_h$ in a given instant. This allows us to over-commit also

the memory usage allowing to use a lower number of nodes and consequently reducing the total energy consumption. We also consider DVFS as leverage where a given frequency $freq_{h,c}$ present in the set $\mathcal{F}_{h,c}$ can be chosen for each $PE_{h,c}$. Moreover, we do not consider the impact of time sharing among the VMs, where extra delays could be introduced by the switching between VMs. We assume that one VM is not interrupted by others.

Table 3.1: Variables notation of IT infrastructure.

Variable	Description
W	Number of hosts
$\mathcal{H}_h$	Host h
C_h	Number of Processing Elements on host h
$PE_{h,c}$	Processing Element c of host h
M_h	Memory available in host h
s_h	State of host h (on or off)
$run_{h,c}$	Boolean indicating if PE c in host h is in use
$freq_{h,c}$	Frequency of PE c in host h
$\mathcal{F}_{h,c}$	Set of available frequencies for c in host h
$c_{h,c}$	CPU percentage used of PE c host h
P_h	Total power of host h
$P_h^{(\text{idle})}$	Idle power of host h
$P_h^{(\text{dyn})}$	Dynamic power of host h
$P_h^{(\text{on})}$	Power overhead to switch host h on
$P_h^{(\text{off})}$	Power overhead to switch host h off
$t_h^{(\text{on})}$	Time overhead to switch host h on
$t_h^{(\text{off})}$	Time overhead to switch host h off

3.1.1 Power Model

The power consumed by $\mathcal{H}_h$ is computed based on Mudge [93] presented in Equation 3.1. In particular we consider $c_{h,c}$ which is the percentage of a given PE that is being used, since in some cases the tasks do not use the entire PE, which does have a proportional impact in the power consumption [88].

$$P_h = \begin{cases} P_h^{(\text{idle})} + \sum_{c=1}^{C_h} run_{h,c} \cdot P_h^{(\text{dyn})} \cdot \left(freq_{h,c}\right)^3 \cdot c_{h,c} & \text{if } \mathrm{s_h} = on \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where s_h determines whether $\mathcal{H}_h$ is *on* or *off*, $P_h^{(\text{idle})}$ is the idle power, $run_{h,c}$

is a boolean describing whether there is a task running in the PE, $P_h^{(\text{dyn})}$ is a host-dependent coefficient, $freq_{h,c}$ is the clock frequency of the $PE_{h,c}$ on host h, and $c_{h,c}$ is the percentage of this PE that is being used. Every PE has a set of available frequencies $\mathcal{F}_{h,c} = \{F_{h,c}^{(1)}, \ldots, F_{h,c}^{(FM_{h,c})}\}$, in such a way that at any instant, $freq_{h,c} \in \mathcal{F}_{h,c}$. Finally, note that under any clock frequency, a power overhead of $P_h^{(\text{on})}$ (resp. $P_h^{(\text{off})}$) is paid during $t_h^{(\text{on})}$ (resp. $t_h^{(\text{off})}$) when $\mathcal{H}_h$ is turned *on* (resp. *off*).

3.2 Task Model

3.2.1 Single Phase Batch Tasks

For the single phase task model we consider that each task utilized an entire $PE_{h,c}$ and the resources consumed are constant over the entire execution. The system receives a set of n tasks $\{\mathcal{T}_j\}_{j\in\{1,\ldots,n\}}$, characterized by the following information: et_j represents the execution time of task $\mathcal{T}_j$ running at a reference frequency $F_{1,1}^{(1)}$. In this case, the reference frequency is defined according to the base hardware where the task was profiled. In another words, the task has the execution time et_j in a given hardware and it can change according to the hardware where it is executed. mem_j is the requested memory, rt_j represents the release time of the task (the moment when $\mathcal{T}_j$ can start to be executed), and d_j represents the due date of this task (the moment when $\mathcal{T}_j$ must be finished). We also highlight that a task is not deleted if it has to be executed after the due date, and that all resources requested must be satisfied.

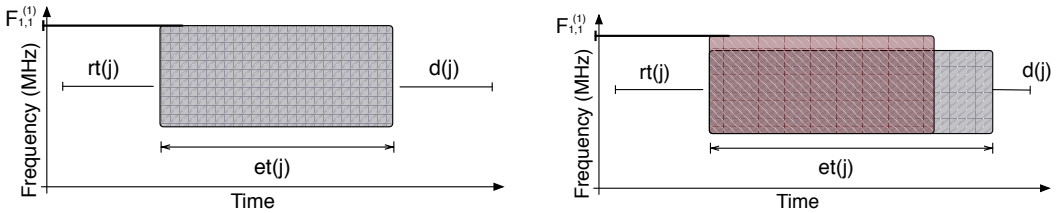
This task model is based on Da Costa et al. [34] which evaluates the traces of Google clusters [105], and as the authors describe the tasks modeled are similar to the one running on HPC infrastructure shared by a large number of users i.e. business intelligence (MapReduce), search, and scientific computing applications. In Table 3.2 we present a list of the considered variables for this model.

Table 3.2: Variables notation for single phase batch tasks.

Variable	**Description**
$\mathcal{T}_j$	Task j
et_j	Execution time of task j
mem_j	Requested memory of task j
rt_j	Release time of the task j
d_j	Due date of the task j

The execution time of $\mathcal{T}_j$ changes according to the $freq_{h,c}$, which allow us to

explore DVFS to change the frequency, and heterogeneous infrastructures. For validation purpose we assume that the execution time varies in a linear way according to the frequency when considering homogeneous nodes, as in Wang et al. [124]. An illustration of a single phase batch task is presented in Figure 3.1. In Figure 3.1 (a) we show the task executing with the maximum frequency available in the node, and in (b) we show the task executing under a lower frequency and the inversely proportional impact that it would have on its execution time. For single phase batch tasks we consider a QoS metric violation when a task $\mathcal{T}_j$ is executed after the d_j.



(a) Single phases batch task executing with highest frequency.

(b) Single phases batch task executing with a lower frequency.

Figure 3.1: Illustration of a single phase batch task.

3.2.2 Multiple Phases Tasks

The previous batch model is used as a starting point for our experiments, but to the best of our knowledge this task modeling represents mostly computation intensive tasks where the CPU would be fully utilized. To better represent the variations that can occur over time in the CPU, memory and network consumption we included this concept of phases presented by Diouri et al. [38], where one single task is subdivided in several time intervals (here called phases) separating where no significant changes in any resource usage occur. Profiting from this phases based model we present two different types of tasks, here called multi phase batch and services. The usage of both task models aim to represent the different type of tasks found in datacenters as demonstrated by Jia et al. [66].

We also introduce here the concept of task phase degradation, where a phase $\mathcal{TP}_{j,t,\varphi}$ of a given task $\mathcal{T}_{j,t}$, where φ is the index of each phase, can receive less resources than the requested, or even no resource at all in the case of services. Here we also consider jobs defined as $\mathcal{J}_j$ that belongs to a user $\mathcal{U}_j$ and have a task of type $type_j$, where type can be *Service* or *Batch*. One $\mathcal{J}_j$ can have several $\mathcal{T}_{j,t}$ inside with different resources request. For this model of batch task we also consider a QoS violation only when $\mathcal{T}_{j,t}$ executes after $d_{j,t}$.

A detailed definition of both type of tasks, as well as the impact of phases degradation on each is presented in the following subsection. We also show in Table 3.3 a summary of all variables to facilitate the comprehension.

Table 3.3: Variables notation of phase based tasks.

Variable	Description
N	Number of jobs
$\mathcal{J}_j$	Job j
$\mathcal{U}_j$	User of $\mathcal{J}_j$
$type_j$	Type of $\mathcal{J}_j$
M_j	Number of tasks in job j
$\mathcal{T}_{j,t}$	Task t of job j
$ett_{j,t}^{(ref)}$	Execution time of task $\mathcal{T}_{j,t}$ in base hardware
$nc_{j,t}^{(ref)}$	Reference number of cores in base hardware of $\mathcal{T}_{j,t}$
$f_{j,t}^{(ref)}$	Reference frequency of cores in base hardware of $\mathcal{T}_{j,t}$
$rt_{j,t}$	Release time of $\mathcal{T}_{j,t}$
$d_{j,t}$	Due date of $\mathcal{T}_{j,t}$
$mem_{j,t}^{(ref)}$	Reference memory in base hardware of $\mathcal{T}_{j,t}$
$L_{j,t}$	Number of phases in $\mathcal{T}_{j,t}$
$f_{j,t}$	Frequency of PE running $\mathcal{T}_{j,t}$
$\mathcal{TP}_{j,t,\varphi}$	Phase φ of $\mathcal{T}_{j,t}$
$etp_{j,t,\varphi}^{(ref)}$	Execution time of $\mathcal{TP}_{j,t,\varphi}$ in base hardware
$ucpu_{j,t,\varphi}^{(ref)}$	CPU percentage usage of $\mathcal{TP}_{j,t,\varphi}$ in base hardware
$mem_{j,t,\varphi}$	Memory usage of $\mathcal{TP}_{j,t,\varphi}$
$uupl_{j,t,\varphi}$	Upload usage of $\mathcal{TP}_{j,t,\varphi}$
$udown_{j,t,\varphi}$	Download usage of $\mathcal{TP}_{j,t,\varphi}$
$cratio_r^h$	Conversion ratio between reference and target hardware
$puc_{j,t,\varphi,h,c}$	Boolean indicating if $\mathcal{TP}_{j,t,\varphi}$ is in $PE_{h,c}$
$ft_{j,t}$	Finish time of $\mathcal{T}_{j,t}$
$nc_{j,t}$	Processing elements received to execute $\mathcal{T}_{j,t}$
$ucpu_{j,t,\varphi}$	CPU percentage usage of $\mathcal{TP}_{j,t,\varphi}$ in current hardware

Multi Phase Batch Tasks

When the IT system receives a set of tasks $\{\mathcal{T}_{j,t}\}_{t\in\{1,\ldots,M_j\}}$, and $type_j = batch$, they are characterized by the following information: $ett_{j,t}^{(ref)}$ represents the execution time of task $\mathcal{T}_{j,t}$ running on $nc_{j,t}^{(ref)}$ cores of reference hardware at frequency $f_{j,t}^{(ref)}$. The resources consumed at this given piece of hardware are

given later in the phases description. Such information of a given task can be obtained by profiling.

The user also inputs $mem_{j,t}^{(ref)}$ which is the requested memory and $nc_{j,t}$ which is the requested number of cores for this task. Here we consider that a job $\mathcal{J}_j$ can request a different amount of resources for each task, giving the user more flexibility. $rt_{j,t}$ represents the release time of the task (the moment when $\mathcal{T}_{j,t}$ can start to be executed), and $d_{j,t}$ represents the due date of this task (the moment when $\mathcal{T}_{j,t}$ should have finished).

Each task is composed by $L_{j,t}$ phases, where each phase $\mathcal{TP}_{j,t,k}$ is an interval with duration $etp_{j,t,k}^{(ref)}$ consuming a constant amount of resources. In this case we consider $ucpu_{j,t,k}^{(ref)}$ as CPU percentage, $mem_{j,t,k}$ as memory consumed (MB), $uupl_{j,t,k}$ as upload bandwidth consumed (kbps) and $udown_{j,t,k}$ as download bandwidth consumed (kbps), where all values are previously obtained on the given base hardware through profiling. In this way, a phases based task can be illustrated as in Figure 3.2. The advantages of modeling a batch task in a phases manner are to represent applications behavior in a more realistic manner, and also the possibility to slow down each phase individually. Examples of service applications degradation can be seen daily, for instance in video streaming, when we have a reduction of the streaming quality [65], or web requests to a given site that could take longer, or not be answered. For batch we can utilize as example big data, and data analysis applications [127] or bank batch processing which could be slowed down and would only be considered degraded if executed after the due date specified.

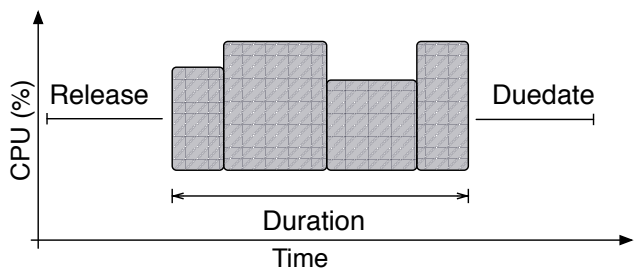


Figure 3.2: Illustration of a phases based batch task.

Multi Phase Services

Despite of the improvements in representing a more realistic workload behavior utilizing batch phases, according to Shen et al. [114] this type of workload has enabled studies in applications such as MapReduce and scientific computing,

but not in business-critical workloads. Business-critical workloads are composed by web services, mail services and application services in general.

In this sense, integrating this diverse service workload is key in evaluating a cloud data center. Contrary to regular batch tasks the patterns of resource usage fluctuates significantly over time, the peak CPU resource usage is 10–100 times higher than the mean [114]. Also for these services, CPU and memory resource usage can be predicted in short-term [61, 114, 121], and network input/output (I/O) shows seasonal patterns. To keep the same modeling, one service is also represented as a job, containing a single task. We know the resources allocated/consumed, but we have no knowledge about the end of the service execution (i.e., it can run for a few hours or for days or never ends). The same phases based modeling for both batches and cloud services allows us to handle the tasks on the same way, prioritizing or not services.

Based on the previous study of [114], IT system receives a set of tasks $\{\mathcal{T}_{j,t}\}_{t\in\{1,\ldots,M_j\}}$, and $type_j = service$, contrary to *batch* , they do not contain the execution time of the tasks nor the due date. The information given are $nc_{j,t}^{(ref)}$ and $f_{j,t}^{(ref)}$, and the resources consumed at this given hardware are also described following the same $\mathcal{TP}_{j,t,k}$ previously presented. These tasks describe long running services, such as web-services, where the cloud provider and the user do not know what will be the amount of resources consumed, when and for how long. In this sense the user only informs $mem_{j,t}^{(ref)}$ which is the requested memory and $nc_{j,t}$ which is the requested number of cores as the description of the machine where the application will be executed. In Figure 3.3 we present a graphical representation of a multi phase service. For phases based services we consider a QoS violation when a task is executed with less resources than requested.

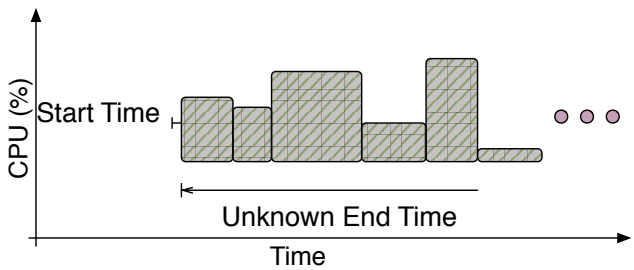


Figure 3.3: Illustration of a phases based cloud service.

3.2.3 Phases Resource Change Impact

This section details how each task is impacted from receiving a different amount of resources than requested. The same applies for the conversion between the

load profile reference resources (ref) and the assigned hardware, as well as when a given phase task is degraded. Phases degradation correspond to a given $\mathcal{TP}_{j,t,\varphi}$ receiving a different amount of resources. The phases degradation can be achieved in several ways, utilizing *cgroups* or *cpulimit* for instance, and have a different impact according to the type of task that is being degraded. We will call a phase degradation for batch tasks as well when a given phase is slowed down, but the task itself is only considered degraded with impact in QoS if the due date is exceeded.

Since the tasks (batch and service) are profiled on a specific hardware we need to consider the changes in execution time according to the resources that are assigned to execute the same task in the data center. For batch tasks we consider memory as a rigid resource, meaning that a task cannot be executed if less memory than the reference memory is available. For services we consider memory as a flexible resource, meaning that a $\mathcal{TP}_{j,t,\varphi}$ can run with less than the reference memory, though resulting in a QoS degradation. This reduction is resources for services could result in a degradation of a video streaming quality, or user requests that would not be answered. For the amount of CPU we allow both services and batches to receive less than the requested.

In summary, according to the definitions presented by Feitelson and Rudolph [41], our workload is composed by moldable batch jobs and rigid services, regarding the number of processors. By definition moldable jobs can execute on multiple resource sizes (number of processors or processor percentage), but once this size is chosen, it cannot be changed during the execution of the job. In addition to that we also consider the variations in the amount of resources (CPU, Memory and Network) during the execution time and the degradation of these resources in some cases. A detailed description is presented below.

Batch Tasks

Batch tasks change their execution time according to the amount of cores and their frequency, *i.e.* according to the computing power of the machines. These changes are based on the base hardware specified for $\mathcal{T}_{j,t}$ ($nc_{j,t}^{(ref)}$ and $f_{j,t}^{(ref)}$) and the target hardware. This hardware change will have a direct impact on the execution time of each phase $\mathcal{TP}_{j,t,\varphi}$ (*i.e.*, if a phase is executed in a slower hardware than the one it was profiled on, it will take longer to finish).

The execution time conversion is expressed in Equation 3.2, where $f_{j,t}^{(ref)}$ is the frequency, $nc_{j,t}^{(ref)}$ the number of cores or processing elements and $cratio_r^h$ is a conversion ratio between the reference hardware r and target hardware h for

application performance. The ratio $cratio_r^h$ is based on the ECU[1] and ACU[2] Amazon (EC2) and Azure computing units respectively, where benchmarks can establish a performance equivalence between two different computing hardware. This ratio is introduced to allow us to convert the execution time of a task among heterogeneous hardware.

$$etp_{j,t,\varphi} = \frac{f_{j,t}^{(ref)} \cdot nc_{j,t}^{(ref)} \cdot ucpu_{j,t,\varphi}^{(ref)}}{f_{j,t} \cdot nc_{j,t} \cdot ucpu_{j,t,\varphi}} \cdot cratio_r^h \cdot etp_{j,t,\varphi}^{(ref)} \tag{3.2}$$

The value of $etp_{j,t,\varphi}$ estimates how long the phase φ will take to be executed in the new hardware. Note that $ucpu_{j,t,\varphi}$ represents the percentage of the requested cores that is actually given for the task to be executed.

Services

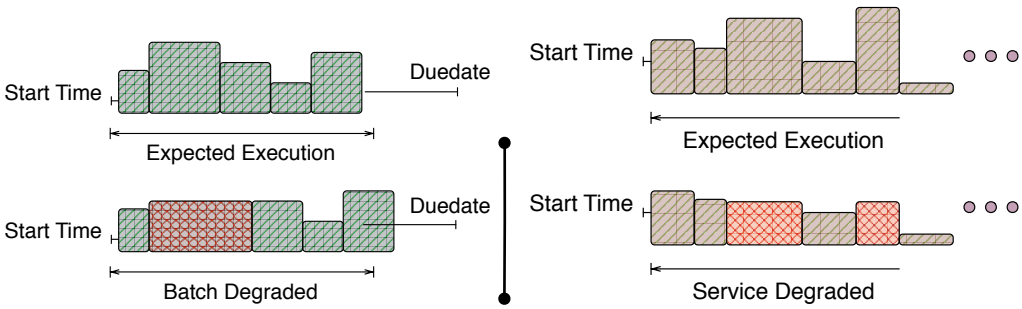
Service tasks do not suffer from time increase if not enough resources are given in relation to the base hardware. When a phase of a service is degraded (receiving less resources than requested), we consider that it will result in users whose requests are not answered or delayed. This will impact only the quality metrics and not the execution time. We also consider that services are multi-core and can receive a different amount of CPU percentage and memory, but not a different number of cores than the one requested.

In Figure 3.4, we present an example of tasks phase degraded, being (a) batch and (b) service. This degradation occurs during the execution time where the phases represented in red receive less resources than requested. The degradation will impact the QoS metric in (b), and in (a) only in the execution time, since the task finishes before the due date even with degradation.

3.3 Workload Generation

In this section, we describe how the workloads used in the experiments were generated and what are the characteristics of each type of task. To perform the evaluation that will be shown in this thesis we considered three different types of tasks generator: batch tasks with single phase, batch tasks with multiple phases and services which are managed through multiple phase workload. Below we detail the three different workload generators.

[1] https://aws.amazon.com/ec2/faqs
[2] https://docs.microsoft.com/en-us/azure/virtual-machines/windows/acu



(a) Batch Degradation. (b) Service Degradation.

Figure 3.4: Graphical representation of a phase degradation in batch (a) and service (b) with and without time increase.

3.3.1 Single Phase Batch Tasks

As an intent to assess the performance of our proposal in a more practical context, we use batch tasks based on a Google workload generator from [34]. One of the authors contributions is an actual implementation of the workload generator with possibilities to adapt to several use cases. The data utilized come from one of the data centers of Google, and was obtained in a dataset provided by [105] representing the servers usage in the year of 2011 over a period of 29 days. The monitored data center is composed by more than 12,000 servers with heterogeneous characteristics. One of the advantages of this generator is that it fits the workload into a distribution allowing us to change the duration of the evaluated period without loosing its characteristics.

The batch tasks are described in the format of an XML file supported by the cloud simulator. In Appendix A.1, we present the XML output of two single phase batch tasks with start and end periods, the execution time of each task, the number of CPUs (PEs) and memory consumed. In the base generator presented by [34], the number of CPUs and amount of memory are always fixed. In the example of Appendix A.1 each task consumes 1 PE and 1024 MB of memory the first one with duration of 2 minutes and 17 seconds, release time 05:59:17 and due date at 06:06:34 of the same day.

3.3.2 Multi Phase Batch Tasks

The previous batch model is used as a starting point, but to the best of our knowledge this task modeling represents mostly HPC tasks where the CPU would be fully utilized. To better represent the variations that can occur over time in the CPU and memory consumption, we included in the previous

generator the concept of phases [38], where one single batch task can be subdivided in several time intervals with different resources consumption.

To do so, we utilized the same Google base workload generator, and included a normal distribution (the parameters can be specified by the user in the generation time) for the CPU percentage consumed. In this way, we artificially introduce variations in the resources. The advantages of modeling a batch task in a phases manner are to represent applications behavior in a more realistic manner, and also the possibility to degrade each phase individually.

In Appendix A.2, we present an XML output of a phases based batch task. We highlight that in this case we also present the reference number of cores in the field "RefCores" and frequency in the field "RefFreq". These values are the ones mentioned in Subsection 3.2.3. The phases XML also includes the total percentage of these PEs utilized in the field "PM_CPU_Usage", as well as memory, upload and download usage.

3.3.3 Multi Phase Services

The service tasks are based on traces from [114], where the authors study traces of business-critical workloads, which represent applications that have a financial impact for the business in case of unavailability or resources degradation. These applications are generally supporting business decisions and are contracted under strict Service Level Agreement (SLA) requirements.

The traces come from a random selection of Virtual Machines (VMs) from the Bitbrains data center between August and September of 2013, in order to guarantee absolute anonymity, since the service provider hosts many services from banking, credit card operators and insurance companies. The monitoring tools record VM performance metrics, sampled every 5 minutes: the number of cores provisioned, the provisioned CPU capacity, the CPU usage (average usage of CPU over the sampling interval), the provisioned memory capacity, the actual memory usage (the amount of memory that is actively used) and the network I/O throughput. In our experiments, we do not consider the disk I/O.

We implemented a services workload generator based on the same mechanism as [38] to detect phases change over the previously mentioned traces. In Appendix A.3, we present an XML output representing a cloud service, compatible with the utilized simulator. This output is generated after running one of the traces presented through our service generator script. We also highlight that in the same way as the phases based batch, we also present the reference number of cores and frequency which are utilized later for the hardware conversion. The main change in this case is that services do not contain start and end dates, so the field "periodStart", "periodEnd" and "executionDuration" are not present. This means that services start running at the beginning of the

evaluated period (which can be without any resource consumption, representing resources that are reserved by a user but without applications being executed) and can end at any time.

3.4 Data center Simulator

Perform cloud data center experiments in real infrastructures can be cost and time consuming. In addition, the experiments can lack reproducibility and suffer effects from several external factors such as location of the reserved cloud instances, latency of the network, and interference from co-scheduled applications. Due to these factors, simulation is a valuable tool and utilized by several authors to evaluate new propositions. Among the available simulators, we can highlight the commonly used in the HPC and Cloud Computing communities: SimGrid [20], CloudSim [19] and DCWoRMS [75].

One possibility, utilized in several works, is the development of simulators specifically for a given work [28, 33, 82]. The main problems are the continuity of these specific simulators and limited functionalities. Another approach, such as the one utilized by Grange et al. [52] is to propose extensions for the commonly used simulators, in this case DCWoRMS, where the authors include the ability to simulate electrical infrastructure with renewable energy sources and storage devices. We chose to utilize DCWoRMS due to these adaptations, as well as the possibility to integrate phases based applications. To integrate an offline power envelope as a power constraint limiting the power consumption of the datacenter, we developed a plugin for DCWoRMS to read data from a CSV file in the format (Relative Time (seconds), Power (Watts)). This plugin allows the input of both power output values as well as forecasted values. Furthermore, this plugin is fully integrated in the simulator and can be utilized along with the extensions of [52] where an offline power forecast can be integrated with the previously available battery and solar panel models.

3.5 Power Production Data Source

In Section 2.5 we detailed the power infrastructure utilized in DataZERO. For the IT side, the whole breakthrough infrastructure can be abstracted as a power envelope. To use as an input for a simulated breakthrough infrastructure we collected data from 6 days solar power production in Toulouse, collected on site at LAPLACE's Laboratory[3]. We considered 3 days of between January

[3] http://www.laplace.univ-tlse.fr

and March of 2016 and 3 days between August and October of 2016 for both productions. The wind data comes from meteociel[4] from the same city.

The samples interval is one hour, and in order to obtain smaller time interval each sample is replicated 20 times (5 seconds interval) to match the solar data time interval. We also included a 10% probability of the occurrence of wind gust. The results obtained, in combination with the solar power data is the one utilized as input in the simulator (i.e. power envelope). The details on the total power production and the graphical representation of the profiles are provided on each section according to the experiments and the infrastructure utilized.

3.6 Cost Metric for Cloud Computing

The costs of building/maintaining a data center has been widely studied. Some methods and softwares have been proposed to define it [81] and [73]. In this section we focus on how to charge the users depending on what task they want to execute and on what machine.

To better understand how cloud providers charge users, we start by studying how the cost metric works in the two main cloud providers [42]. In 2018 Amazon EC2 [6] was responsible for 47.8% of the market share and Microsoft Azure [92] for 15.5%.

3.6.1 Type of Computing Services

Here we focus only in computing/processing Linux machines offered by the providers. In this category we have two possibilities:

- On-Demand (EC2) or Pay as you go (Azure): In this type of instance the user pay for compute capacity by per hour or per minute depending on which instances selected. No longer-term commitments or upfront payments are needed. One can increase or decrease the compute capacity depending on the demands of the application.

- Reserved Instances (EC2) or Reserved Virtual Machine Instances (Azure): This type of instances are the same as previous but are reserved for a period of 1 to 3 years. The main advantage is the price and instances can be assigned to a specific Availability Zone (geographic location). Also a user can select to launch/consolidate instances when desired and it is

[4] http://www.meteociel.fr/temps-reel/obs_villes.php?code2=7630&jour2=13&mois2=11&annee2=2016&envoyer=OK

mainly utilized for applications that have steady and long term state or predictable usage.

- Amazon (exclusively) also offers two other types of instances: (i) EC2 Spot instances which allow the user to request spare computing capacity. Basically it consists in a pool of machines that users reserved for a given amount of time and don't want anymore (sold back); and (ii) a Dedicated Host, which in the end results in a cost which can be up to tens of thousands per month. These two types of instances will no be considered in this study due to its high price variability (spot instances) and specific niche of applications (reserved machine).

Physically there is no difference between Reserved Instances and On-Demand instances. The only difference is in billing and availability. For this reason and for being the most popular/utilized one, we will focus on the On-Demand instances.

3.6.2 Compensation and Resources Guarantee

Considering the on-demand cloud instances, there is no performance guarantees in anyone of the providers, just a compensation in case of degradation in the QoS [5, 10, 91]. The client is the one responsible for detecting and reporting the problem. Some of the providers such as Azure establish a time period of 30 days to report, otherwise the claim is not accepted. The compensations provided are not in form of money, but as service credits that are applicable in the next bill.

The mapping of vCPU is generally hidden and there is no guarantees that the user is being given a full core and therefore only looking at vCPU and price relation can be misleading. According to the official documentation, by default OpenStack, which is the commonly employed cloud manager, allows the overcommitment[5] of CPU and RAM on compute nodes. This allows the provider to increase the number of instances running on the host at the cost of reducing the performance of the instances. The default CPU allocation ratio for OpenStack of 16:1 means that the scheduler allocates up to 16 virtual cores per physical core. For example, if a physical node has 12 cores, the scheduler sees 192 available virtual cores. With typical flavor definitions of 4 virtual cores per instance, this ratio would provide 48 instances on a physical node.

[5] https://docs.openstack.org/arch-design/design-compute/design-compute-overcommit.html

3.6.3 How is the performance of the machines presented

The data evaluated contains mainly the number of vCPU, Memory, Storage (if already included) and the price of the instance. For Amazon and Azure two customized metrics, ECU and ACU respective, are introduced.

ECU[6] (EC2 Compute Unit): The metric represents the amount of CPU that is allocated to a particular instance expressed in terms of EC2 Compute Units. The value is based on several benchmarks (not disclaimed) and tests. One EC2 Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. The goal is to provide a consistent amount of CPU capacity no matter what is the actual underlying hardware.

ACU[7] (Azure Compute Unit): Similar to ECU, based on a small standard A1 and translated to the remainder instances available, but in this case the metric is not explicitly presented, and the user needs to make the conversion based on the type of server manually.

These metrics help specially when considering heterogeneity in the data center. This factor makes hard to estimate the performance. For instance, one vCPU (Virtual CPU) in a given hardware can have half of the performance in another one. In Table B.2 we have a machine A4v2 with 4 vCPUs 8GB of memory at $0.191/hour, and a machine F4 with the same amount of vCPUs and memory at $0.219/hour. Logically one would chose the A4 where the user gets the same resources for a smaller price. Nevertheless when converting to the ACU performance metric the F4 provides more than double of the performance.

3.6.4 How to Calculate Profit

To define how much the user should pay for task execution in a given machine usage, before we start applying the compensations in case of violation. Pricing informations where obtained in the links below on "US Central" regions and are also presented in the Appendix B. The objective here is to determine which resource impacts on pricing in real cloud platforms so we can follow a similar cost model in order to maximize the return on the scheduled tasks.

Storage is generally considered as a separated service but some instances already come with some storage available. Nevertheless, as we can observe by the fitting below it doesn't play a significant role in pricing, compared to memory and CPU capacity and is generally sold separately, so it will not be presented here, nor it is considered in our workload model. The evaluation presented here is based on the relation between price, memory, and processing

[6] https://aws.amazon.com/ec2/faqs/#What_is_an_EC2_Compute_Unit_and-why_did_you_introduce_it

[7] https://docs.microsoft.com/en-us/azure/virtual-machines/windows/acu

capability. We present both evaluation based on ECU and ACU and also on vCPU only.

3.6.5 Price Calculation Based on Amazon EC2

To evaluate Amazon prices we consider Amazon pricing[8] presented in Table B.1 which contains mainly the number of vCPU, ECU, Memory, Storage (if already included) and the price of the instance. We start by analyzing in Figure 3.5 the number of vCPUs in the X axis, the memory (GB) in the Y axis and the price represented by the circle radius. This chart shows that when comparing all the instances together, the factor with the highest weight for pricing is the memory allocated and not the number of vCPUs. This can be observed by the growth of circle radius in the vertical and not so much in the horizontal. The same can be observed when considering ECU metric in Figure 3.6.

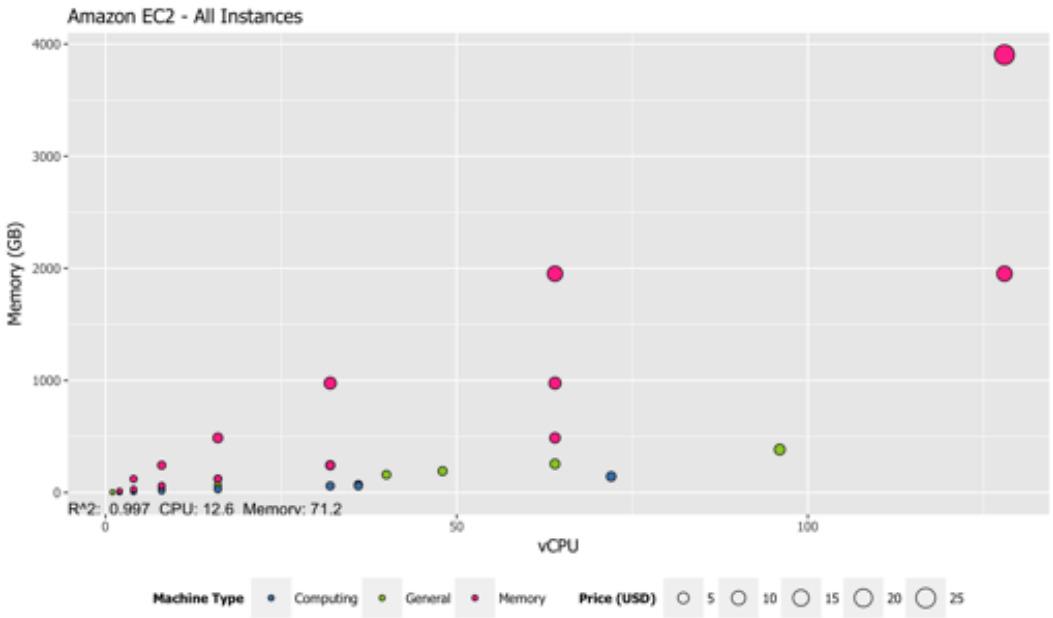


Figure 3.5: Price variation according to traces of vCPU and Memory for all instances of Amazon EC2.

To confirm it, we performed a relative variable importance analysis using a multiple linear regression, and evaluating the t-statistic of each variable which can be compared across all variables. The values for the R^2 obtained in this case is 0.997 for vCPU and 0.998 for ECU. We utilize then the regression coefficients

[8] https://aws.amazon.com/ec2/pricing/

to calculate the base price (without violation) for each task, according to the hardware allocated (vCPUs/ECU and amount of memory) and the time.

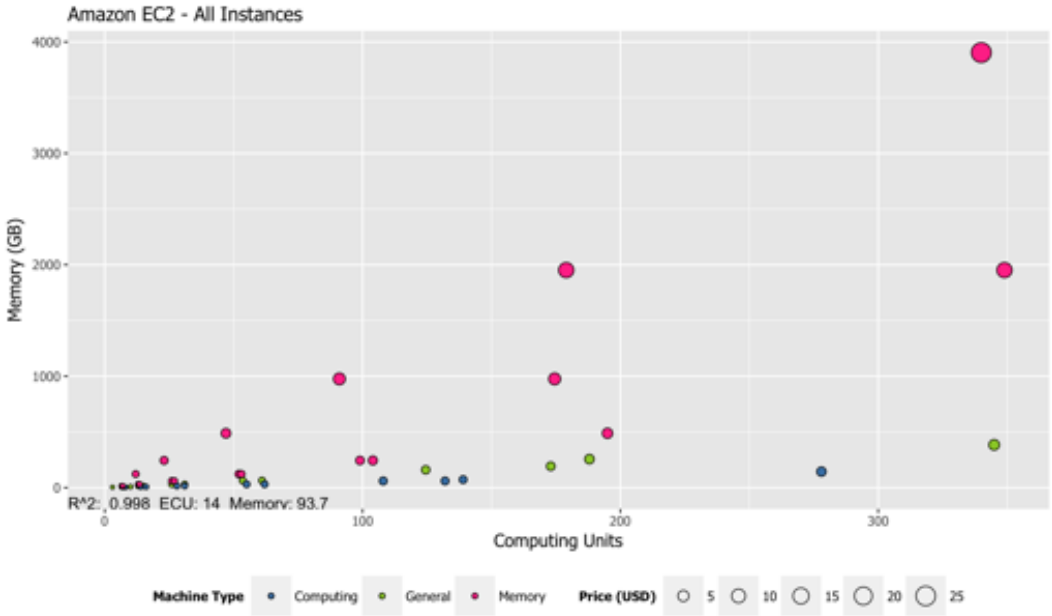


Figure 3.6: Price variation according to traces of ECU and Memory for all instances of Amazon EC2.

3.6.6 Price Calculation Based on Microsoft Azure

To evaluate Azure prices we consider Azure pricing[9] presented in Table B.2 which contains mainly the number of vCPU, ACU, Memory, Storage (if already included) and the price of the instance. We start by analyzing in Figure 3.7 and Figure 3.8 where the charts can be interpreted the same way as the previous ones from Amazon. Again, the factor with the highest weight for pricing is the memory allocated and not the number of vCPUs. This can be observed by the growth of circle radius in the vertical and not so much in the horizontal. The same can be observed when considering ECU metric.

The fitting obtained for both providers allow us to estimate the cost of running each task presented in previous models. In our case we consider that the user is billed for the machine allocated in a minute basis. For services, we consider that the user reserved the amount of requested resources and will pay even if it is not used (i.e., reserved a machine with 10GB of memory but

[9] https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/

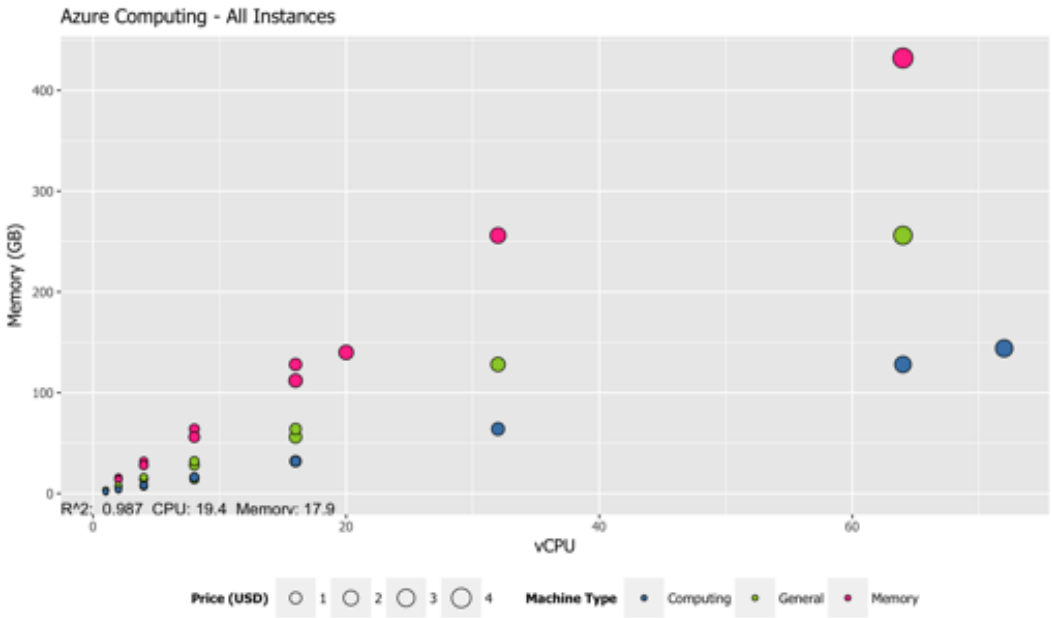


Figure 3.7: Price variation according to traces of vCPU and Memory for all instances of Azure.

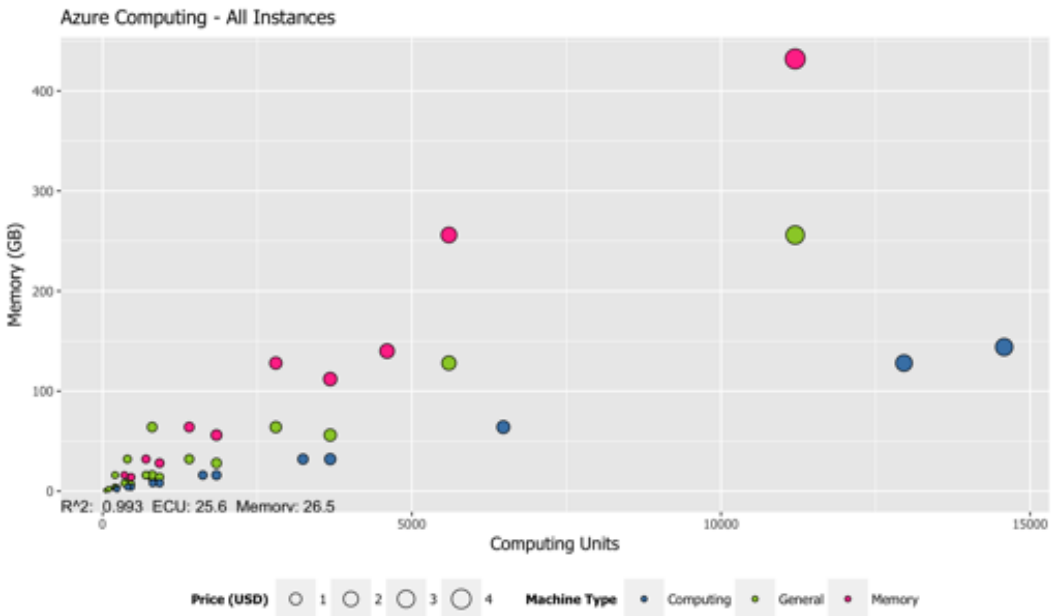


Figure 3.8: Price variation according to traces of vCPU and Memory for all instances of Azure.

is using only 2GB). On the other hand, for batch tasks the user pays only for the amount of resources that are utilized during the execution, also in a minute basis. These models can be utilized to calculate the profit that could be obtained to run a task on any given machine using as input the amount of memory and CPU utilized. In the corresponding part where we approach profit in Chapter 5 we present the degradation and loss in profit that can occur on each task and that will result in a compensation for the user.

3.6.7 Network Price

The cost for network traffic within the data centers is not charged. The traffic from server to the Internet follows the Table 3.4. In this case both providers have similar prices, and we present only Amazon EC2 prices.

Table 3.4: Data movement price in Amazon EC2.

Data Movement Price Range	**Cost per GB**
<1 GB	0.00
0.001-10 TB	0.09
10-50 TB	0.085
50-150 TB	0.07
150> TB	0.05

3.7 Conclusion

This chapter focused on presenting the infrastructure, power and task models that are utilized in the remainder of this thesis. The aim is to provide the basis to understand the following chapters. Moreover, we presented also what is the impact on the resources changes in the tasks performance, the concept of base hardware where the applications are profiled, and what are the data sources considered. We also present an extension of a workload generator from the literature to include phases batch tasks as well as a service workload generator based on the same task model. Finally we introduced the simulator utilized to conduct the experiments and the changes that were made to input the different power envelopes to test our scheduling proposals in the next chapters.

Chapter 4

Single Phases Scheduling Approach

In this chapter we propose to optimize the IT scheduling of batch tasks to execute tasks within a given power envelope of only renewable energy as a constraint. We utilize greedy heuristics as well as meta-heuristics to perform the task placement, with aim at minimizing the due date violations. The results here presented are related to the publications in Caux et al. [22] and Caux et al. [21].

4.1 Introduction

Since energy efficiency in data centers is directly related to the resource consumption of a computing node [97], performance optimization and an efficient load scheduling is essential for energy saving. Today, we observe the use of cloud computing as the basis of data centers, either in a public or private fashion. The task management is first optimized by Virtual Machine (VM) management [15], where a task should be placed considering an energy consumption model to describe the task's consumption, depending on the resource description (processor and memory power characteristics) and task's demand (resources usage) while respecting the Quality of Service, which in this case we consider as the number of due date violations.

To address the IT load scheduling while considering the renewable energy available we start our approach by proposing a module to schedule single phase batch tasks as presented in Section 3.2, which are characterized by their release time, due date and resource demand, in a cloud data center while respecting a power envelope. This envelope represents an estimation which would be provided by a power decision module and is the expected power production

based on weather forecasts, states of charge of storage elements and other power production characteristics. This chapter aims at maximizing the Quality of Service with a constraint on electrical power. There are several possible power envelopes which could be generated using only renewable energy sources and the different moments when storage elements can be engaged. This interaction between data center electrical consumption and electrical power sources part is fundamental to profit as much as possible from the renewable energy sources. In this chapter we detail and evaluate the approach with a comparison between classical greedy algorithms and meta-heuristics constrained by power envelopes.

The remainder of this section will present the problem statement in Section 4.2, followed by the proposed resolution in Section 4.3. The evaluation methodology and results for homogeneous data centers is presented in Section 4.4 as well as the results obtained for heterogeneous data centers in Section 4.5. Finally, we compare the approaches in both infrastructures in Section 4.6, followed by the conclusions of the chapter in Section 4.7.

4.2 Problem Statement

The IT scheduling problem, in our case, consists in allocating tasks on the IT resources under constraints depending on the computing platform current state and on energy availability. Several levels of decision are concerned as IT resource management (server switch on/off, process migration, voltage and frequency scaling, etc). On the other power production side, we have the power systems where several power profiles could be provided, depending on the moment when the renewable energy is produced and the batteries are engaged for instance. In our case, according to the DataZERO project context we abstract it as a power envelope.

According to the general model presented in the previous chapter here we define the C_c granularity to a Processor. The aim is to find when and at which frequency to run every task, *i.e.* to find assignment functions σ_{PE}, σ_{host} and σ_{freq} expressing that $\mathcal{T}_j$ runs on processor $\sigma_{\mathrm{PE}}(j)$ of host $\sigma_{\mathrm{host}}(j)$ at frequency $F^{(\sigma_{\mathrm{freq}}(j))}_{\sigma_{\mathrm{host}}(j),\sigma_{\mathrm{PE}}(j)}$, and a starting point function st expressing that $\mathcal{T}_j$ starts at time $st(j)$. We denote by $ft(j)$ the finish time of $\mathcal{T}_j$, hence, for all j:

$$ft(j) = st(j) + \frac{F^{(\sigma_{\mathrm{freq}}(j))}_{\sigma_{\mathrm{host}}(j),\sigma_{\mathrm{PE}}(j)}}{F^{(1)}_{1,1}} \cdot et_j. \tag{4.1}$$

The problem can then be formulated as follows: minimize $\sum_j \max(0, ft(j) - d_j)$, where $ft(j)$ is the finish time of task j and d_j the due date, subject to memory and power constraints.

4.3 Proposed Approach

Finding a mapping of the tasks onto the processors such that no due date constraint is violated is an NP-complete problem [2, 68], while DVFS is not enabled and memory is not taken into account, even with two processors. In this way, we focus on approximation methods. More specifically, we explore Greedy Heuristics and Genetic Algorithms, introduced in Section 2.6 as proposals to solve the allocation problem. Greedy Heuristics can provide locally optimal decisions, and in general have a short execution time. On the other hand, the combinations of choices locally optimal do not always lead to a global optimum. The second approach (Genetic Algorithm), can provide a large number of adapted solutions and also makes possible to approach a local minimum starting from an existing solution. Nevertheless, the problem of GA methods can be the execution time on large scale problems. In this work we utilize a time window approach, meaning that the scheduling is performed for a specific time interval. More specifically, an off-line resource allocation problem is considered with a fixed set of tasks that have constant resource needs.

The difference from regular scheduling algorithms is that in this case we need considering the power envelope as a constraint. To do so, the implemented algorithms use a power check function which is responsible for evaluating if a task can be scheduled in a given processing element on the desired time interval. It returns how much power would be consumed to schedule the task using a specific processor and frequency. Hereafter, two different approaches that provide scheduling possibilities are presented.

For GH, we considered three versions of the Best Fit, where we use different sort task functions. The algorithms try to fit the tasks in the node that presents the smallest power consumption, respecting the power envelope and resource constraints. We also present three versions of the First Fit algorithm which schedules a task at the first available node which can finish the task before the due date. The difference among the three versions of each algorithm is the way that the tasks are sorted: (i) Due date, closest task first; (ii) Arrival time, first task that arrives is the first to be scheduled; and (iii) Task size, longest one first. Even though the changes occur only in the task ordering, the impact on the results can be significant.

All considered GH algorithms must respect the power envelope, meaning that if there is not enough power in a given time step to power a machine, this task will be delayed until the next time step in which a possible solution is found (increasing the start time). The tasks are not canceled if executing after the due date. A pseudocode of the Best Fit Due Date used is presented in Algorithm 1 where we start sorting the tasks by due date, then creating a Map of which processors are in use and at what time interval. We schedule all the

tasks in the queue verifying the power envelope constraint and then memory and processors usage.

Algorithm 1: Best fit constrained by a power envelope pseudocode.

```
   input  : Set of tasks in queue, set of resources available, power envelope for the
            window
   output : Tasks with time and processor assigned
 1 begin
 2     sortTasks(queue);
 3     Map<Processor,TimeIntervalArray> timeIntervals;
 4     while queue.hasTasks() do
 5         t = queue.getTask();
 6         processor = null;
 7         startStep = t.release;
 8         foreach Processor currentP in resourcesAvailable do
 9             if processor == null and
               haveEnoughPowerToSchedule(t,currentP,powerEnvelope) and
               verifyConstraints(t,currentP,timeIntervals,startStep) then
10                 processor = currentP;
11             else if processor != null and
               haveEnoughPowerToSchedule(t,currentP,powerEnvelope) and
               verifyConstraints(t,currentP,timeIntervals,startStep) and
               energyIncrease(currentP) < energyIncrease(processor) then
12                 processor = currentP;
13             end
14         end
15         if processor != null then
16             schedule(t,processor,startStep);
17             updateTimeIntervals(processor,timeIntervals,startStep,t);
18             updatePowerEnvelope(processor,timeIntervals,startStep,t);
19             queue.remove(t);
20         else
21             startStep += stepSize;
22         end
23     end
24 end
```

Regarding the GA, each chromosome represents a scheduling possibility for the given power profile. In Figure 4.1 we show how each individual is initially represented, as well as an example of cross-over operation (Algorithm 2) where each gene represents a task and the value is the node where it will be executed. For the crossover operation we consider two points crossover since it allows the change of a higher number of genes in a single operation, and the selection consists in tournament selection, which allows the best fitted genes to survive. After that, the processor, frequency and time are assigned using a greedy

algorithm, illustrated in Figure 4.2 since we are working with time as well and include it in the representation would generate several invalid solutions.

To improve the execution time of both GAs (the verification of the power available occurs for each step in the power envelope) we also utilize a power envelope with two different granularities. The first one provides a rough scheduling based on an aggregation of the initially provided envelope, reducing in this case the number of *steps*. After obtaining an initial placement, a fine grained power envelope (smaller steps) is used to absorb power peaks and respect the given power envelope.

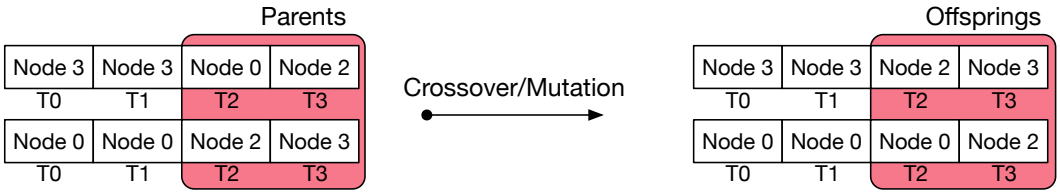


Figure 4.1: Genetic algorithm chromosome representation and crossover example.

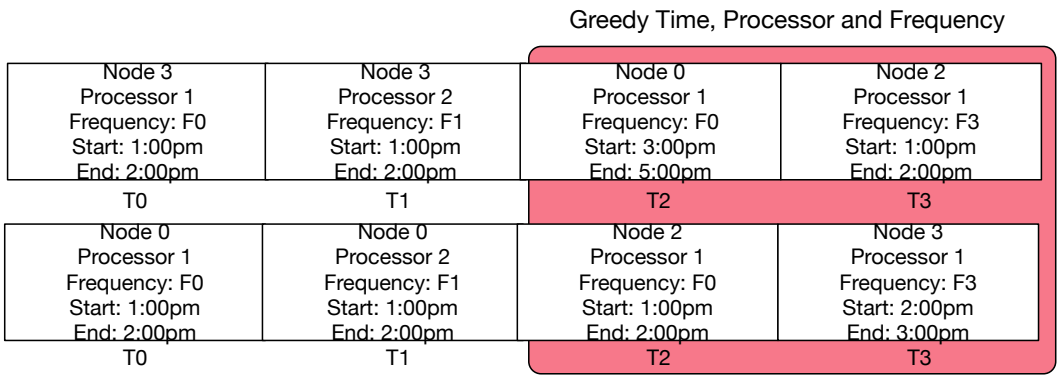


Figure 4.2: Genetic algorithm greedy assignment of processor and frequency.

Utilizing this modeling for the GA we propose two fitness variations. The first one where the fitness function consists only in reducing the number of due date violations, and the second one uses a weight based approach, also trying to minimize the power consumption in a Mixed Objective (hereafter called MPGA - MultiPhase Genetic Algorithm and MPGA-MO - MultiPhase Genetic Algorithm Mixed Objective, respectively). Equation 4.2 is used to normalize all metrics considered for each chromosome $\mathcal{C}_k$, described below, where $M^{(\max)}$ is the maximum value for a given metric, $M^{(\min)}$ is the minimum, and M_k is the value of the kth chromosome. The normalized values are then inputs

in Equation 4.3 where DD_k is the normalized due date violations and E_k is the normalized energy consumption. The metrics should be weighted using α, depending on the importance of the objective (for MPGA the only metric considered is the number of due date violations, *i.e.* α is equal to 1).

$$M_k^{(\text{norm})} = \frac{M^{(\max)} - M_k}{M^{(\max)} - M^{(\min)}} \tag{4.2}$$

$$\mathit{fitness}_k = \alpha \times DD_k + (1 - \alpha) \times E_k \tag{4.3}$$

A pseudocode of the GA used is presented in Algorithm 2 where it can be seen the generation of the simplified envelope in line 2 (assigned to individuals in line 4), the first execution from line 6 to 11, and the execution with the detailed power envelope and the respective stopping criteria from line 12 to 19. The stopping criteria for the MPGA, since it only considers the number of due date violations, is when it has at least one chromosome that has no violation, or the maximum number of generations is reached. For the second algorithm (MPGA-MO) the stopping criteria is only the number of generations, since the minimum energy to schedule the tasks in advance cannot be defined easily.

When a set of individuals of a generation is computed, the greedy algorithm in Algorithm 3 is utilized for the time schedule and DVFS adjustment is done with Algorithm 4 (*scheduleAndCheckConstraints* called in lines 8 and 16). In a simplified manner, how the tasks would be allocated in a processor is presented in Figure 4.3 where we illustrate a node with two processors. In (a) we present the scheduling after the greedy algorithm that defines the time and processor inside a node is executed. The aim of this greedy algorithm is to align the execution of the processors of the same node to be able to switch it off. First we populate an associative array with all the tasks and the time intervals where they can be scheduled. After, we get the first unscheduled task and compare if there is another task which the time to be schedule intercepts this time interval. The algorithm evaluates then, what is the earliest start time in which the tasks can be allocated and not violated.

The pseudocode of the algorithm that defines the time and processor inside a node where the tasks will be executed is presented in Algorithm 3. The aim is to align the execution of the processors of the same node to be able to switch it off. In line 4 we populate an associative array with all the tasks and the time intervals where they can be scheduled. After, in line 7 we get the first unscheduled task and compare if there is another task which the time to be scheduled intercepts this time interval (line 10). The algorithm evaluates then, between lines 10 and 14 what is the earliest start time in which the tasks can be allocated and not violated. Finally, between lines 15 and 31 the algorithm finds a free processor inside the node and schedule the tasks in parallel (as

Algorithm 2: Multiphase genetic algorithm pseudocode.

```
input  : Set of tasks in queue, set of resources available, power envelope for the
         window, selection method, population size, number of generations first phase,
         number of generations second phase, number of simplified steps, mutation
         probability, crossover probability
output : Tasks scheduled, actions to be performed in nodes, QoS metrics, power
         consumption estimation
1  begin
2      simplifiedPowerEnvelope = generateSimplifiedEnvelope(powerEnvelope,nSteps);
       /* First Phase - Simplified Power Envelope                                  */
3      foreach Individual i in population do
4          i.setPowerEnvelope(simplifiedPowerEnvelope.copy);
5      end
6      generateInitialPopulation();
7      for (g=0; g < generationsFirstPhase; g++) do
8          scheduleAndCheckConstraints(individuals);
9          calculateFitness(individuals);
10         selectionMethod.select(individuals);
11     end
       /* Second Phase - Detailed Power Envelope                                   */
12     foreach Individual i in population do
13         i.setPowerEnvelope(powerEnvelope.copy);
14     end
15     while StopCriteriaNotReached do
16         scheduleAndCheckConstraints(individuals);
17         calculateFitness(individuals);
18         selectionMethod.select(individuals);
19     end
20 end
```

illustrated by $\mathcal{T}_1$ and $\mathcal{T}_3$ in Figure 4.3 (b)). We also highlight that the algorithm always verifies the power envelope and resources constraints.

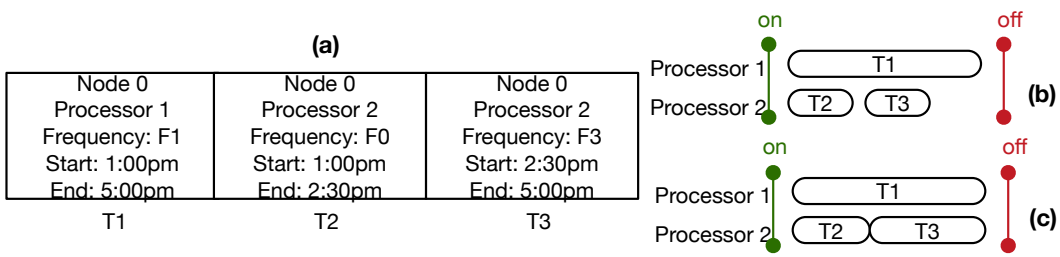


Figure 4.3: Tasks allocation inside a node with two processing elements using greedy scheduling inside GA (a), and DVFS adjustment where (b) is before DVFS and (c) after DVFS adjustment.

Algorithm 3: Time scheduling algorithm, called for each node with the corresponding tasks.

```
   input  : Set of tasks in queue, set of resources available, power envelope for the
            window
   output : Tasks with time and processor assigned
 1 begin
 2     sortByArrivalTime(queue);
 3     foreach Task t in queue do
 4         tasksWindow.add(t,new TimeInterval(t.arrival,t.duedate));
 5     end
 6     while queue.hasTasks() do
 7         t = queue.getTask();
 8         scheduled = false;
 9         processor = null;
10         startStep = getTimeIntercept(t,tasksWindow);
11         end = endTimeEstimation(t,startStep);
12         if !(startStep > -1 and startStep > t.arrival and end < t.duedate) then
13             startStep=t.arrival;
14         end
15         while !scheduled do
16             foreach Processor currentP in resourcesAvailable do
17                 if haveEnoughPowerToSchedule(t,currentP,powerEnvelope) and
18                 verifyConstraints(t,currentP,timeIntervals,startStep) then
19                     schedule(t,currentP,startStep);
20                     updateTimeIntervals(currentP,timeIntervals,startStep,t);
21                     adjustTimeIntersect(tasksWindow,t.start,t.end);
22                     powerEnvelope.subtractPower(t,currentP,startStep);
23                     queue.remove(t);
24                     scheduled=true;
25                     break;
26                 end
27             end
28             if !scheduled then
29                 startStep++;
30             end
31         end
32     end
33     adjustDVFS(tasksInNode,powerEnvelope);
34 end
```

A pseudocode of the DVFS adjustment is presented in Algorithm 4. From line 3 to 14 the objective is to search if there are two tasks that intercept each other (considering the execution time), but are on different processors, to adjust the frequency of the one that finishes earlier (toAdjust). The algorithm then gets the task that will be executed next, in line 16, in the same processor,

to not overlap the execution times. In line 17 the desired end time for the task to be adjusted is calculated, either finishing as close to its due date as possible, before the next task that will be executed in the processor, or as close as possible to the end of the last task in the processors of the same node (to not impact in the idle power consumption). The frequency that matches as close as possible this time is then chosen at line 18. The power consumption of the desired changes are validated between line 19 and 23, and the power consumed in the current envelope is then updated.

To illustrate, in Figure 4.3 (c) we show a per processors DVFS where it reduces the frequency of Processor 2 in this case, to reduce the power consumption, and consequently increasing the execution time of tasks $\mathcal{T}_2$, $\mathcal{T}_3$. The frequency in this case is only reduced if the due date is not violated. This DVFS control does not impact the idle power consumption of a node, allowing an easy consolidation of nodes where more energy saving can be obtained. In this sense, at the end of the task placement and DVFS adjustment we also calculate when each node can be turned off in order to reduce the power consumption without impacting the system performance.

Algorithm 4: DVFS adjustment algorithm to align tasks on different processors at the same node.

```
   input  : set of tasks scheduled in the same node, power envelope for the window
   output : Tasks with DVFS adjusted
 1 begin
 2     sortByStartTime(taskQueue);
 3     adjustedTasksList = new List();
 4     foreach Task t1 in tasksScheduled do
 5         foreach Task t2 in tasksScheduled do
 6             if t1.processor != t2.processor and tasksIntercept(t1,t2) then
 7                 if t1.end > t2.end then
 8                     lastEnding = t1;
 9                     toAdjust = t2;
10                 else
11                     lastEnding = t2;
12                     toAdjust = t1;
13                 end
14             end
15             if !adjustedTasksList.contains(toAdjust) then
16                 nextTasks = getNextTaskStarting(toAdjust,toAdjust.processor);
17                 desiredEnd =
                   getClosestTime(nextTask.start,toAdjust.dueDate,lastEnding.finish);
18                 newFreq = getFreqBasedOnTime(toAdjust.start, desiredEnd);
19                 if haveEnoughPowerToSched-
                   ule(toAdjust,toAdjust.end,desiredEnd,powerEnvelope) and
                   verifyConstraints(toAdjust,toAdjust.proc,toAdjust.end,desiredEnd)
                   then
20                     toAdjust.freq = newFreq;
21                     alreadyAdjusted.add(toAdjust);
22                     updatePower(powerEnvelope);
23                 end
24             end
25         end
26     end
27 end
```

In a real infrastructure all the operations performed by the IT scheduler would occur through the cloud data center middleware. This means that we not only deal with the scheduling but also with switching on and off the machines and adjusting the frequency of the processor. We can consider our approach as an entire module that could be inside or outside the cloud platform, while the communication is established through messages exchange. These messages are interpreted by the middleware and can either concern the tasks or the IT infrastructure. Tasks can be scheduled in a given host or removed and these actions can generally be performed through the cloud middleware API previously presented. For the hosts operation we can either rely on shutdown and Wake-on-Lan like commands or ePDUs [1] that can be controlled remotely which allows both to control and monitor the power consumption of each machine connected.

4.4 Homogeneous Data Center Evaluation

To validate this proposed model we simulated an IT and power production infrastructure (through a power envelope). The module which controls on a simulated IT infrastructure is provided by DCWoRMS [76]. We have implemented the scheduling algorithms previously mentioned inside DCWoRMS as a scheduling plugin.

The IT infrastructure inside the simulator is based on Villebonnet et al. [123], more specifically we are using 30 nodes and the power consumption values of Paravance cluster from Grid5000 [2]. The set of frequencies available and their corresponding power consumption is presented in Table 4.1. For Paravance we considered $P^{(on)} = 112.91\,W$ over $t^{(on)} = 189\,s$ and for $P^{(off)} = 65.7\,W$ over $t^{(off)} = 10\,s$. Regarding the power model presented in Section 3.1.1, in this case we assume that each batch task utilizes the full PE.

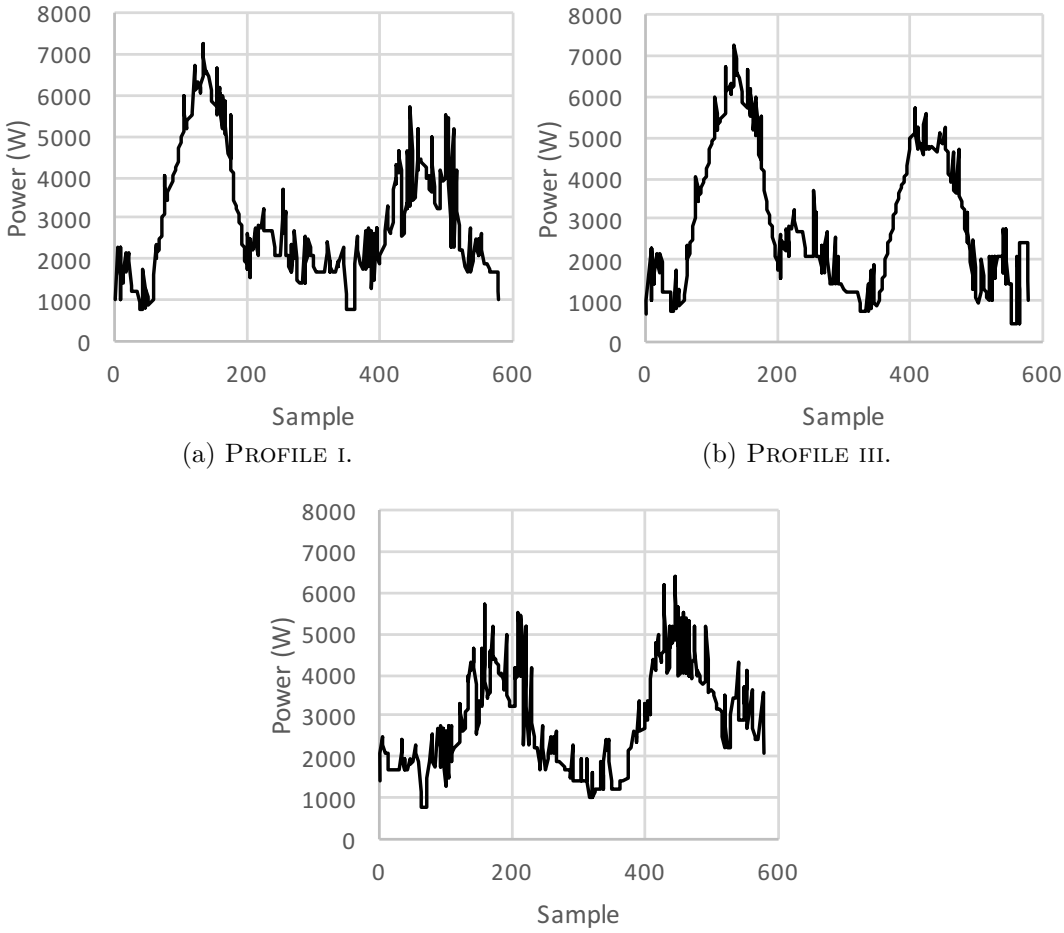
Table 4.1: Power values for each frequency of the considered nodes.

Paravance Freq. (GHz)	2.4	2.16	1.92	1.68	1.44	1.2
Power (W)	200.5	165.10	136.76	114.69	98.10	86.22

Regarding the GA, we bound the number of generations to 100 (resp. 400) with the simplified power envelope (resp. with the original power envelope) and the population size to 100 individuals. The probabilities for crossover and mutation are 0.9 and 0.3 respectively. The results presented are the average

[1] https://powerquality.eaton.com/Products-services/Power-Distribution/ePDU/
[2] https://www.grid5000.fr/

of 10 executions for each GA run. For the MPGA-MO we consider $\alpha = 0.9$ where the main objective is minimize the number of due date violations. These values were obtained empirically through experimentation and comparison of results/improvement obtained. For single phases batch workload generator, presented in 3.3.1, we utilized a window of 2 days (i.e. all the tasks have to be executed inside this interval). With this time window we generate 3 different workloads with 234, 569 and 1029 tasks.



(a) Profile i.

(b) Profile iii.

(c) Profile ii.

Figure 4.4: Graphical representation of the three power profiles.

Each workload is scheduled with 3 different power profiles as observed in Figure 4.4. Profile i with peak production of 7249W and average of 2879W, Profile ii peak production of 7249W and average of 2893W and Profile iii with peak production of 6387W and average of 2756W. Even though the values are similar, the moment in which the power is delivered is

different, as observed in Figure 4.4. These different power profiles will allow us to evaluate the impact of the moment when the power is delivered in the QoS of a cloud-based data center when executing different workloads. The results concerning these experiments as well as a graphical representation of the power profiles are presented below.

4.4.1 Results Evaluation

The objective of this section is to present the results concerning the impact of each power profile in the QoS of a cloud based data center when executing batch tasks. In Figure 4.5 we present the number of due date violations and the power consumption in Figure 4.6 for all the workloads proposed.

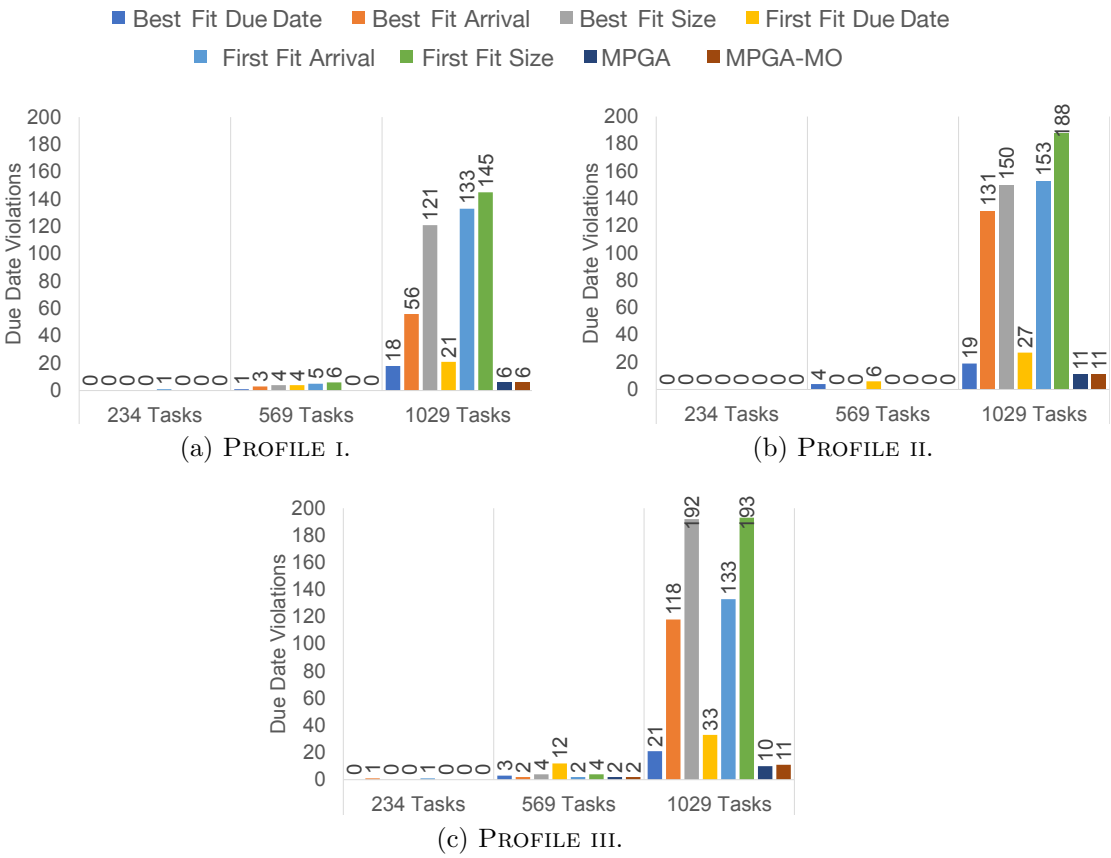
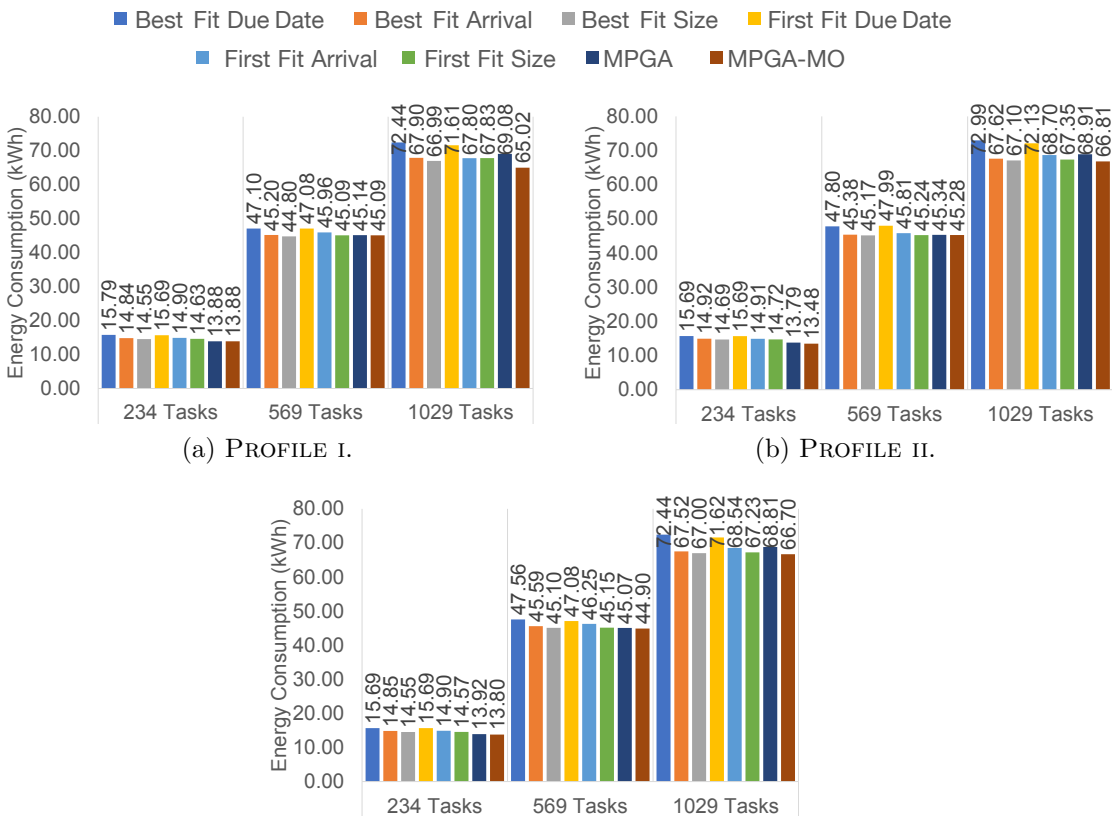


Figure 4.5: Due date violations of all power profiles and workload variations in homogeneous infrastructure.

With Profile i, 234 and 569 tasks MPGA-MO has an energy consumption up to 12.10% smaller when compared with Best Fit Due Date, maintaining 0



(a) Profile i.

(b) Profile ii.

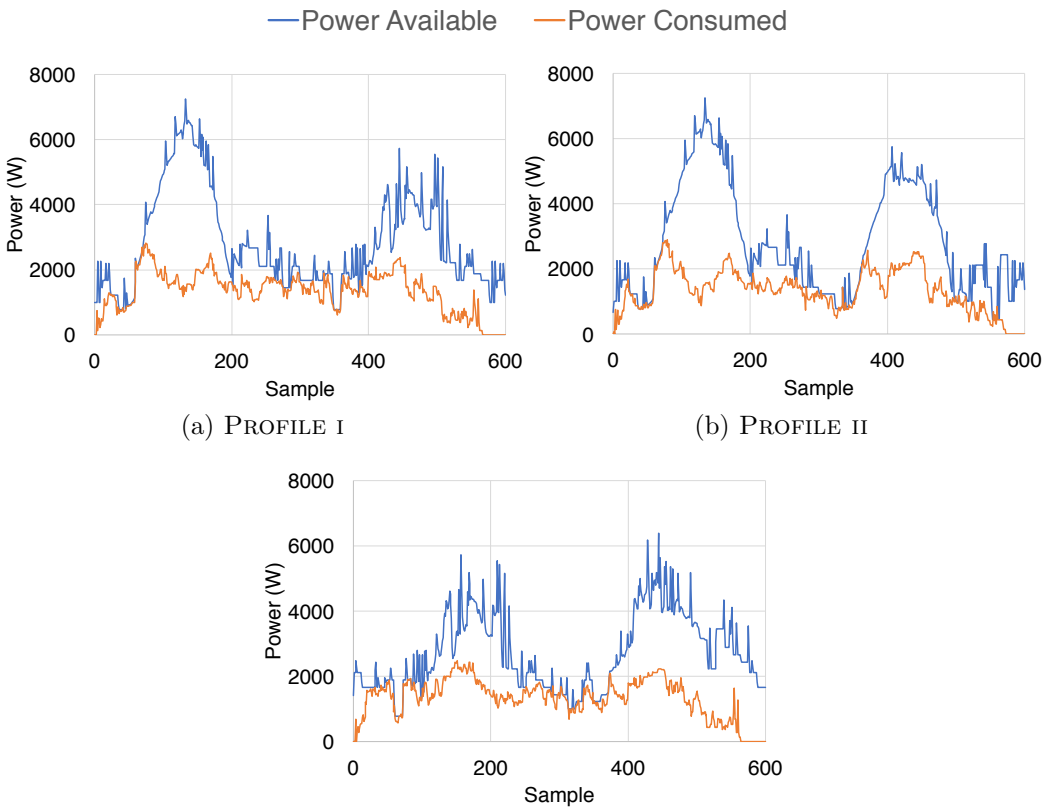
(c) Profile iii.

Figure 4.6: Energy consumption of all power profiles and workload variations in homogeneous infrastructure.

due date violations in Profile i. With 1029 tasks in this same power profile the GA energy consumption can be 10.24% smaller than Best Fit Due Date, with a reduction from 18 to 6 due date violations. With Profile ii, 234 and 569 tasks GA has an energy consumption between 14.12% and 12.14% (MPGA-MO and MPGA respectively) smaller when compared with Best Fit variations, and 0 due date violations for both. With 1029 tasks the GA energy consumption reduction can be 8.47% for MPGA-MO and 5.59% for MPGA compared to the other variations, with a reduction from 19, compared to Best Fit Due Date, or 188 for First Fit Size, to 11 violations for both GAs.

In Figure 4.11 we present the profile of the energy produced and consumed for each one of the three power profiles. These results were obtained when using the genetic algorithm scheduling and the biggest workload (1029 tasks). We can observe that task scheduling, when the power is available is more important than how much power is available. MPGA-MO in Profile iii for instance has

less energy consumed, and less due date violations than Profile i and Profile ii. This occurs because some tasks do not have enough flexibility in terms of time to wait the moment when the power will be available. This highlights the importance of the generation of multiple power envelopes when considering renewable energy sources and storage elements engagement. We could not only save energy but also provide a better QoS. This excess of power available could either be sold or stored. In the DataZERO context, the negotiation loop would be in charge of choosing the most suitable power envelope.



(a) Profile i

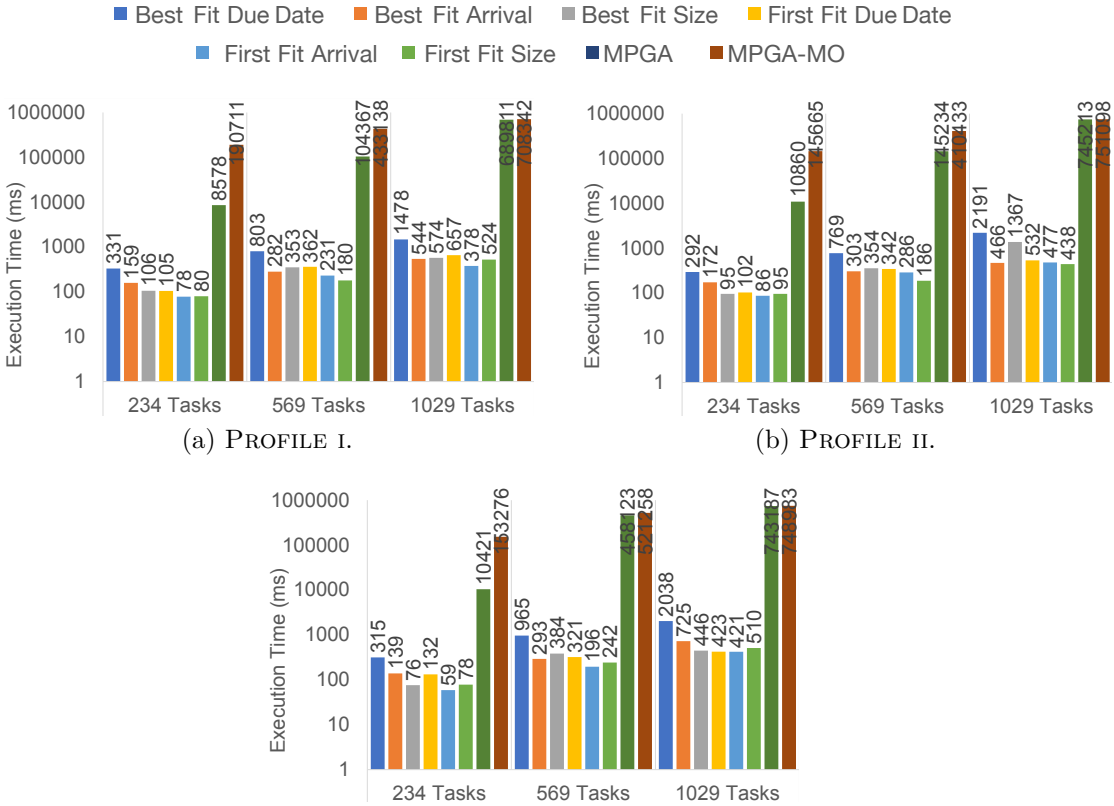
(b) Profile ii

(c) Profile iii

Figure 4.7: Energy available and consumed in the power profiles using MPGA-MO based scheduling plan.

In Figure 4.8 we present the average execution time of all the algorithms. Despite of the smaller number of due date violations and lower energy consumption, as expected, the Genetic Algorithm variations can have an execution time exponentially higher than the greedy ones. Nevertheless, if the scheduling requested is not a reactive action, this execution time is not prohibitive (approx. 12 minutes in the worst case for two days scheduling). We also highlight that it

is possible to improve even more the execution time by improving the stopping criteria, but this might have an impact of the quality of the schedule.



(a) Profile i.

(b) Profile ii.

(c) Profile iii.

Figure 4.8: Execution time of the different algorithms with different number of tasks with all profile variations and homogeneous infrastructure.

4.5 Heterogeneous Data Center Evaluation

The new IT infrastructure inside the simulator is also based on Villebonnet et al. [123], but more specifically we are using 30 hosts (same amount but 15 of each kind) and the power consumption values of Paravance and Taurus clusters from Grid5000[3]. In Table 4.2 we present the distinct frequency values and their respective power consumption. These values are based on the power model presented, using for all hosts $P^{(\mathrm{dyn})} = 4.725\,W \cdot s^3$ (see Equation 3.1) and $P^{(\mathrm{idle})} = 69.9\,W$ for Paravance and $P^{(\mathrm{dyn})} = 5.255\,W \cdot s^3$ and $P^{(\mathrm{idle})} = 95.8\,W$

[3]https://www.grid5000.fr/

for Taurus. For Paravance we considered $P^{(on)} = 112.91\,W$ over $t^{(on)} = 189\,s$ and for $P^{(off)} = 65.7\,W$ over $t^{(off)} = 10\,s$. For Taurus we considered $P^{(on)} = 125.78\,W$ over $t^{(on)} = 164\,s$ and for $P^{(off)} = 106.63\,W$ over $t^{(off)} = 11\,s$. The same number of executions and GA parameters are kept from the previous experiment.

Table 4.2: Power values for each frequency of the considered nodes.

Paravance	**Freq. (GHz)**	2.4	2.16	1.92	1.68	1.44	1.2
	Power (W)	200.5	165.10	136.76	114.69	98.10	86.22
Taurus	**Freq. (GHz)**	2.3	2.07	1.84	1.61	1.38	1.15
	Power (W)	223.7	189.03	161.28	139.67	123.43	111.79

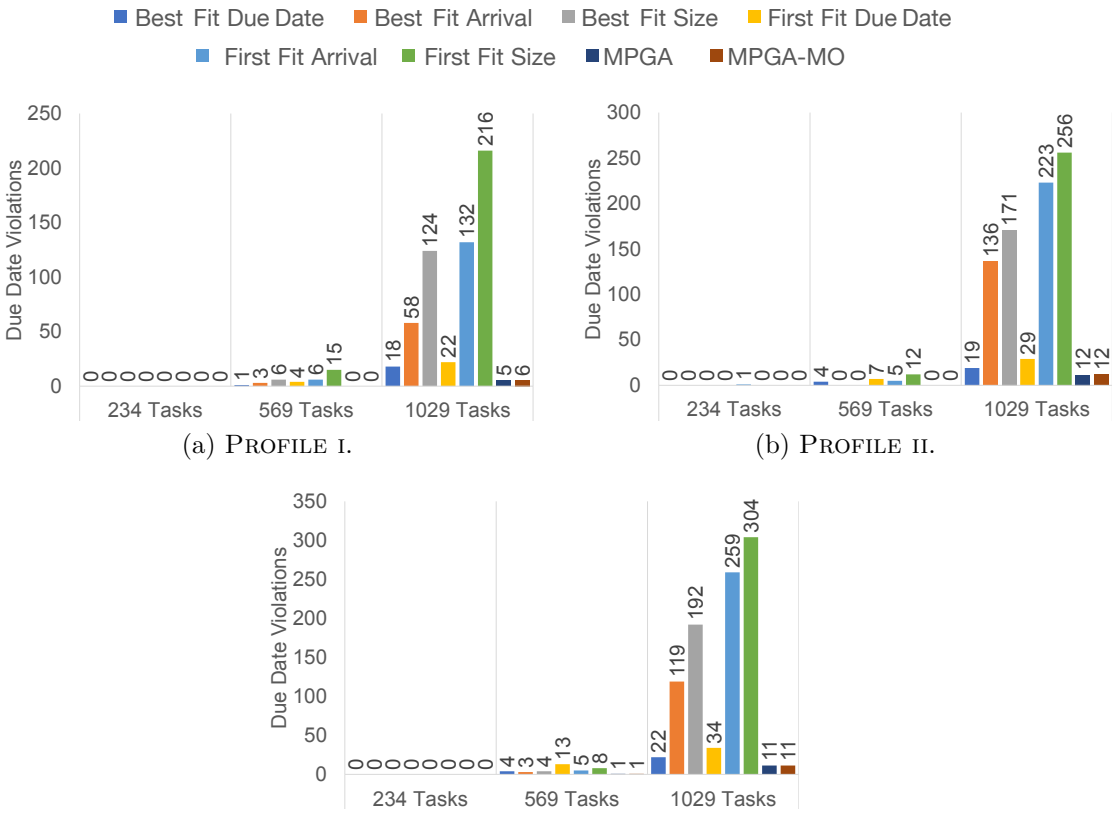
4.5.1 Results Evaluation

In Figure 4.9 we present the number of due date violations and in Figure 4.10 the energy consumption for all the proposed workloads and algorithms, considering the three different power profiles.

Considering the three power profiles with only 234 tasks, almost all algorithms, even with the most contrained power envelope, can reduce the number of violations to 0 and keep the energy consumption around 15kWh. The exceptions in this case are the first fit algorithms which have a higher energy consumption (around 18kWh) and one violation with Profile ii. As the number of tasks increases an expected degradation of performance of both first fit and best fit algorithms is observed when compared to the GA. When considering 1029 tasks we have in Profile i 18 due date violations for the best fit algorithm against 5 and 6 of the two genetic algorithms variations, which also obtained a reduction of 6.3% in the energy consumption. In Profile ii we observed the same behavior, reducing from 19 to 12 due date violations with a reduction of 4.9% in the energy consumption. The same goes for Profile iii which reduced from 22 to 11 due date violations with an economy of 5% in energy. The values for the total time violated of the tasks may seem high but we need to consider that the scheduling is constrained by a power envelope, and in this case the tasks need to be delayed for the next moment with enough power available (if we consider only solar energy for instance, this may take a whole day).

In Figure 4.11 we present the power produced and consumed for Profile i. These results were obtained when using the Best Fit Due Date (a) and MPGA-MO (b) scheduling planners with Profile i and 1029 tasks. We can observe that in some points (such as in the first 100 samples) the power consumption

can be similar for both algorithms due to the high number of tasks that needs to be scheduled and so reaching the maximum power available. This justifies why we have different number of due date violations with the same workload under different power profiles: at some points we have too many tasks to be scheduled, and they lack flexibility (time between release and due date) to wait the next moment where enough power will be available (samples 100-200). We could not only save energy but also provide a better QoS; this behavior can be observed by comparing the results obtained with Profile i against the two others, which have a higher number of violations and in case of Profile ii also a higher energy consumption.



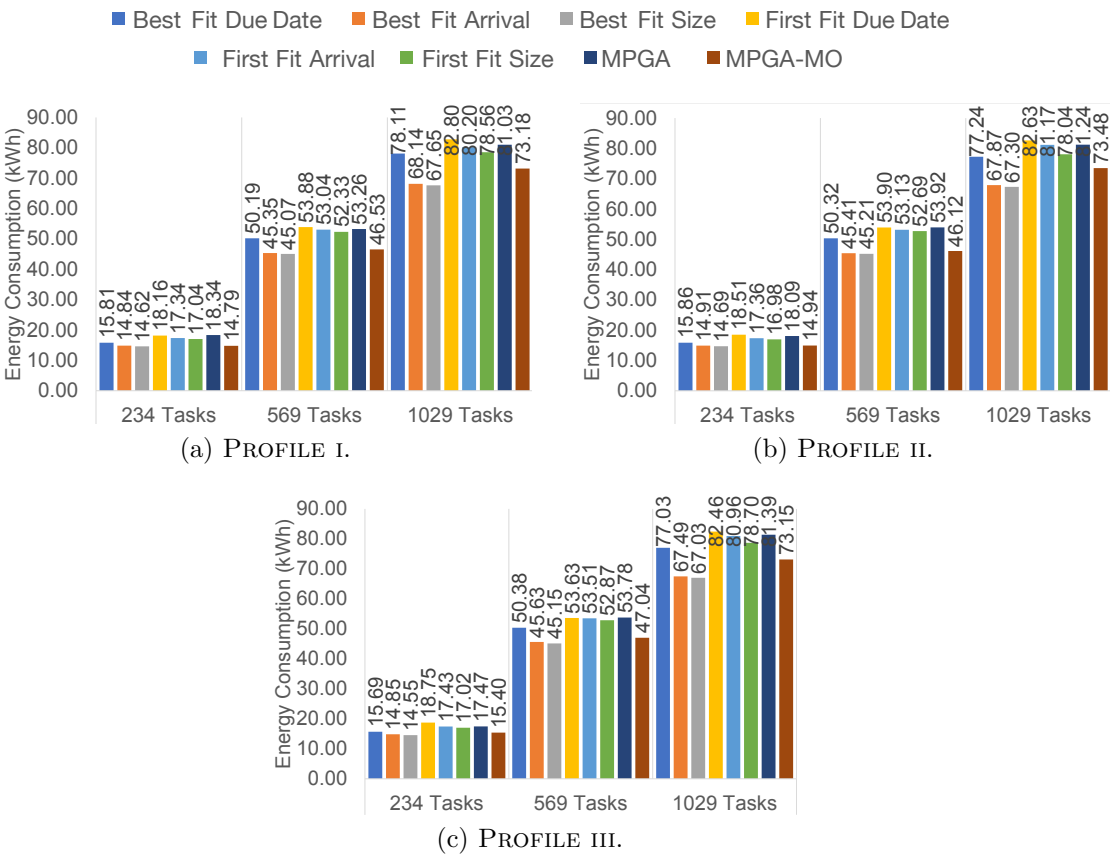
(a) Profile i.

(b) Profile ii.

(c) Profile iii.

Figure 4.9: Due date violations of all power profiles and workload variations in heterogeneous infrastructure.

The results become even more significant if we consider the long term impact that it could provide. For Profile i, displayed in Figure 4.11, in a period of 2 days we could save 164.98 kWh using the MPGA-MO, instead of 155.35 kWh and 160.04 kWh for first fit and best fit due date respectively. This energy



(a) Profile i.

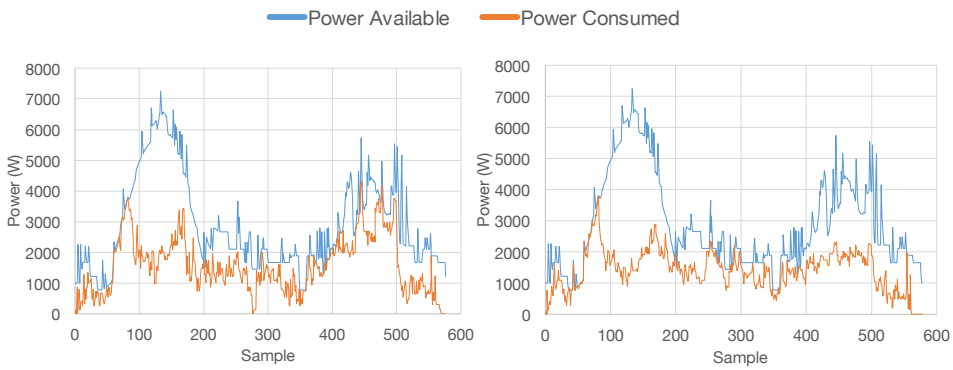
(b) Profile ii.

(c) Profile iii.

Figure 4.10: Energy consumption of all power profiles and workload variations in heterogeneous infrastructure.

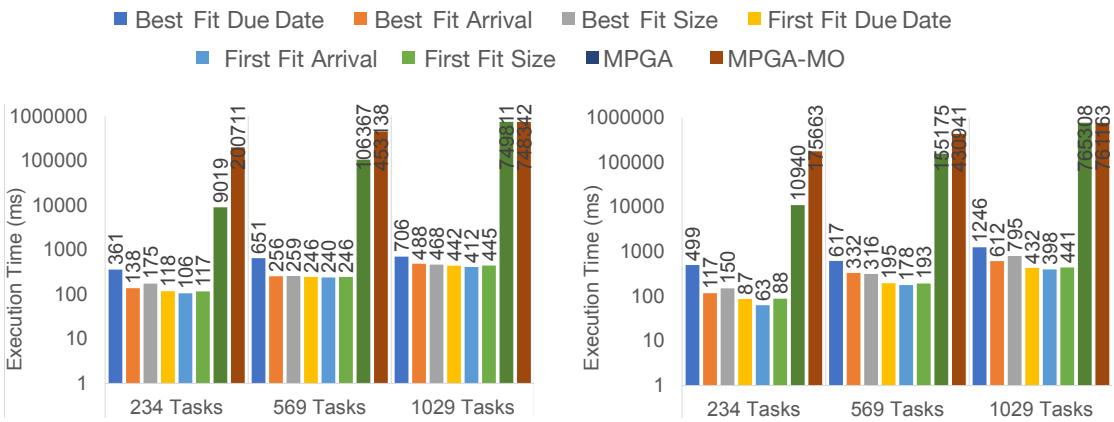
could be stored and used in the generation of the next scheduling windows improving the results, or sold to the grid power provider.

In Figure 4.12 the average execution time of all the algorithms (with minimum and maximum values in the bars) is presented. Despite of the smaller number of due date violations and lower energy consumption, as expected, the Genetic Algorithm can have an execution time exponentially higher than the greedy ones. Nevertheless, if the scheduling requested is not a reactive action, this execution time is not prohibitive (around 12 minutes in the worst case for two days scheduling). We also highlight that it is possible to improve even more the execution time by improving the stopping criteria, but this will have an impact of the quality of the schedule.

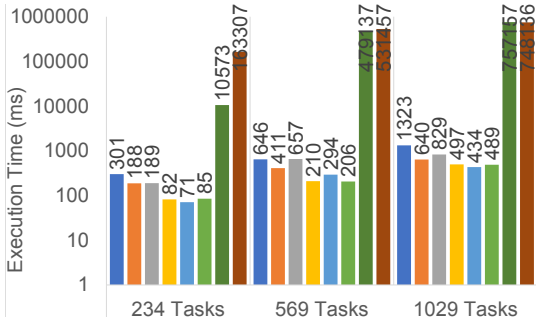


(a) Best Fit Due Date (b) MPGA-MO

Figure 4.11: Power available and consumed in the power PROFILE I considering two different algorithms and 1029 tasks.



(a) PROFILE I. (b) PROFILE II.



(c) PROFILE III.

Figure 4.12: Execution time of the different algorithms with different number of tasks with all profile variations and heterogeneous infrastructure.

4.6 Comparison Homogeneous vs Heterogeneous Performance

Despite of not being the main focus of this section, one can also compare the experiments with an homogeneous infrastructure and heterogeneous one. Regarding the execution time of the MPGA and MPGA-MO in both cases tends to be very close, with an increase of 5% on average for the heterogeneous experiments and almost no variation on due date violations. On the other hand, when comparing the performance of some of the greedy algorithms we can see an increase in the number of due date violations in the heterogeneous platform, mostly when scheduling the biggest workload (1029 tasks).

The increase in the number of violations for the greedy heuristics can be up to 48% in First Fit variations, where First Fit arrival goes from 133 violations in the homogeneous platform to 259 in the heterogeneous one. This increase occurs in the best fit variations, but in a less accentuated manner with an average of 3.02% more due date violations for the heterogeneous platform. This increase is mainly attributed to the fact that the nodes from Taurus consume approximately 11% more than the Paravance ones. In a traditional scenario this would not impact the results, but since we are considering a scenario where the power is constrained by a power envelope, the nodes power efficiency also has a role in the number of violations. One of the aspects also evaluated by DataZERO project, not contemplated in this thesis, is exactly the evaluation of sizing and node efficiency for the cloud data center.

4.7 Conclusion

This chapter focused on presenting and evaluating the scheduling optimization of single phase batch tasks. The aim is to schedule tasks in a cloud data center while respecting the possible power envelopes.

We presented different algorithms, some classical from literature and two genetic algorithm variations that try to minimize due date violations while respecting power and resource constraints. The proposed genetic algorithm approaches (MPGA and MPGA-MO) were able to reduce the number of due date violations in comparison to the other classical algorithms. MPGA-MO was able to keep a lower number of due date violations while also reducing the energy consumption. We have also presented an evaluation of the impact the power envelopes can have in the task scheduling, and concluded that more power does not necessarily means better QoS for the IT part, but it is more important to know when this power is delivered. Finally, we did a comparison of the results obtained in homogeneous data center versus heterogeneous one

and how the algorithms performed in both cases.

This chapter works as basis for the next one, where we will include also services and phases based tasks in the scheduling approach. Furthermore we will explore the changes in the amount of resources delivered to a task, and what are the benefits when considering a power envelope constrained data center.

Chapter 5

Multi Phases Scheduling Approach

In this Chapter we utilize a more precise application model from the previous chapter and add a cost metric for the task placement. We utilize greedy heuristics, as in the previous chapter, and adapt them to handle the new task and cost. We also introduce the usage of cross-correlation in the algorithms and the reduction of the delivered resources (degradation). The results here presented are related to the publication in [24] and [23].

5.1 Introduction

Even though batch tasks and services are present in the daily workload of a cloud data center and involved models have been proposed and used in various contexts (*e.g.* evolving, moldable and malleable tasks, by Feitelson and Rudolph [41]), the majority of works in the literature for renewable energy still consider a simplified task modeling with rigid resources assignment. Some works [3, 11] include services, but yet assume connection to the grid, so do not consider a power constraint over a time-window.

Considering the literature background we focus on maximizing the profit by improving QoS of a data center powered only by renewable energy sources. To handle the several moments where there is not enough power to provide the requested resources for a given workload, the literature [21, 67, 112] considers only the removal of tasks, or power off enough, or all, nodes when there is a reduction in the power production. Also, these authors disregard the reduction of CPU and memory resources of given tasks in a specific period and its impact in QoS, execution time and reduction of the power consumption. This possibility of delivering less resources than what was requested is still fundamental to take

into account the variations in the renewable energy sources [40, 74].

In our proposal, we consider a realistic workload composed by batch and services, where the resources consumption varies over time. We also consider a task model that can receive less resources than requested, and the impact on QoS that it has for each task type, detailed in Section 5.2. Finally, we include the concept of cross-correlation to evaluate where to place a task under a power curve, and what is the best node to place tasks together. The contributions of this work are the following: (i) we propose new scheduling approaches for phases based batches and services; (ii) we focus on non-exact optimization methods, more specifically we explore Greedy Heuristics as a way to validate our proposal with a time window approach and an offline resource allocation problem with a fixed set of tasks. The difference from the previous approaches is that in this case, we need to handle both batch tasks and services, while considering: (1) the power envelope as a constraint and; (2) the multiple phases and resources change of each task.

The remainder of this Chapter is organized as follows. First we introduce the optimization objectives in Section 5.2, followed by the problem formulation in Section 5.3. After we show the proposed resolution in Section 5.4 with a detailed description on the degradation and algorithms. We then show the methodology in Section 5.5, followed by the results in Section 5.6. Finally we present the steps towards a time optimized approach in Section 5.7 and the conclusions in Section 5.8.

5.2 Optimization Objectives

Since we are working with a problem that not only regards the user, but also the cloud data center as a service provider, we need to utilize a well defined and accepted metric. This metric should not be partial nor to the user side neither to the data center side. The metric that we choose is the profit based on the study that identifies the parameter that impact price of tasks on cloud computing of providers such as Amazon and Azure, presented in details in Section 3.6. Even though the profit might seem like a metric that reflects only the data center side, there are several compensations that are applied if the QoS is not kept.

The optimization takes place on the cloud provider side: the objective is to schedule tasks so that the profit of the cloud provider is maximized. This does imply that it can maximize its profit regardless to its user, since they are bound together with an SLA (Service Level Agreement) which stipulates that the user should be compensated if its expectations are not met. Moreover here we define the C_c granularity to a Core.

In the following, we introduce the QoS degradation compensation that we designed after a study and an adaptation of existing cloud providers presented in Section 3.6. The compensations are computed according to the type of task, which was introduced in Subsection 3.2.2.

QoS Degradation Compensation for Batch Tasks

When a phase is degraded, i.e. received less resource than requested, a batch task will increase its execution time, contrary to services. In this case, as long as there is no due date violation no penalty occurs (i.e. the phase is degraded, the task is not). If a due date violation occurs, we can apply a similar compensation for the job, as seen in SLA for cloud providers, according to the percentage of time violated. The violation time ratio is expressed by Equation 5.1, where a higher violated time implies a proportional compensation. A summary of the variables utilized in the notation is available in Table 3.3.

$$QoSb_j = \begin{cases} \frac{1}{M_j} \sum_{t=1}^{M_j} \frac{ft_{j,t} - d_{j,t}}{ett_{j,t}^{(ref)}} & \text{if} ft_{\text{j,t}} > d_{\text{j,t}} \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

Equation 5.1 gives the average violated ratio between the due date $d_{j,t}$ and the finish time $ft_{j,t}$ (in seconds) in relative to the $ett_{j,t}^{(ref)}$ (execution time in base hardware) of all violated tasks in job $\mathcal{J}_j$. For instance, if a job composed by one task with due date at time $t_0 + 10\,s$ (where t_0 means the simulation start time) finishes at $t_0 + 11\,s$ we would have a 10% violation ratio. This value will then be used according to Table 5.1 to give a compensation to the user in case of a $QoSb_j \geq 5\%$ where the maximum of credit received cannot exceed 100% of the total bill.

To encourage users to give more flexibility (time between release time $rt_{j,t}$ and due date $d_{j,t}$), thus allowing the cloud provider to schedule tasks in a moment where more renewable energy is available, we also rely on a flexibility ratio $Flex_j$, defined in Equation 5.2, such that users with higher flexibility will receive a higher compensation in case of QoS violations, limited to a maximum of 30%. The value of 30% was chosen arbitrarily, being the same as the maximum violation ratio.

$$Flex_j = max\left(\sum_{t=1}^{M_j} \frac{d_{j,t} - rt_{j,t}}{ett_{j,t}^{(ref)}}, 30\right) \tag{5.2}$$

Table 5.1: Violation compensation for batch tasks.

Violation Ratio	**Credit**
5%	5% + $Flex_j$ / 100
10%	10% + $Flex_j$ / 100
20%	20% + $Flex_j$ / 100
30%	30% + $Flex_j$ / 100

QoS Degradation Compensation for Services

Based on SLA violation for services, when an execution phase does not receive the reference amount of resources, a QoS degradation occurs, which is expressed in Equation 5.5. Equation 5.3 represents the total computing power of the base hardware given by ($cpuUsage \cdot frequency$). In the same way, Equation 5.4 represents the total computing power of the hardware utilized to execute a given phase φ, where $freq_{h,c}$ indicates the frequency of PE running $\mathcal{T}_{j,t}$ and $puc_{j,t,\varphi,h,c}$ is a boolean indicating if $\mathcal{TP}_{j,t,\varphi}$ is in $PE_{h,c}$.

We also highlight that here we are considering the total computing power of the base hardware as $\sigma_{\text{comp}}^{(ref)}$ and σ_{comp}, which represents the total computing power given to the phase (according to the cores assigned and its frequency/usage), *i.e.*, the equivalence previously presented $100\% \cdot 2Cores \cdot 1GHz \equiv 50\% \cdot 4Cores \cdot 1GHz$ if $cratio_r^h = 1$.

$$\sigma_{\text{comp}}^{(ref)}(j,t,\varphi) = ucpu_{j,t,\varphi}^{(ref)} \cdot f_{j,t}^{(ref)} \tag{5.3}$$

$$\sigma_{\text{comp}}(j,t,\varphi) = \begin{cases} \sum_{h=1}^{C_h} ucpu_{j,t,\varphi} \cdot freq_{h,c} & \text{if } puc_{\text{j,t,}\varphi\text{,h,c}} \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

$$QoSs_j = \frac{\sum_{t=1}^{M_j} \sum_{\varphi=1}^{L_{j,t}} min\{\frac{\sigma_{\text{comp}}(j,t,\varphi)}{\sigma_{\text{comp}}^{(ref)}(j,t,\varphi)}, 1\} \times etp_{j,t,\varphi}^{(ref)}}{\sum_{t=1}^{M_j} \sum_{\varphi=1}^{L_{j,t}} etp_{j,t,\varphi}^{(ref)}} \tag{5.5}$$

In Equation 5.5, the maximum value is limited to 1 to avoid an overprovisioning of resources in one phase acts as compensation for another. The time $etp_{j,t,\varphi}^{(ref)}$ of each phase is added as a weighted sum, since some phases can be shorter than others and should not have the same weight. The QoS calculation is also done for the amount of memory received and requested. The minimum of both values will then be used according to Table 5.2 to give a compensation to the user in case of a $QoSs_j \leq 99.95\%$.

Resources requested by the user will be priced according to the price of the closest machine with enough resources to execute the job. The price of these machines is proportional to the amount of CPU and RAM available based on

previously studied cloud providers in Section 3.6. The ratio between resources received and requested for services will be considered for the compensation following the same principles as public cloud providers, and presented on Table 5.2.

Table 5.2: Violation compensation for services.

Availability Ratio	**Credit**
99.95% to 99.00%	10%
99.00% to 95.00%	25%
95.00% to 90.00%	35%
<90.00%	50%

For network, the price presented in Subsection 3.6.7 will be used for both batch and services according to $uupl_{j,t,\varphi}$ and $udown_{j,t,\varphi}$.

5.3 Problem Formulation

The aim is to find where and when to run every task, *i.e.* to find assignment functions σ_{core} , σ_{host} and σ_{freq} expressing that phase $\mathcal{TP}_{j,t,\varphi}$ of $\mathcal{T}_{j,t}$ runs on PE (here considered as each core) $\sigma_{\text{core}}(j,t,\varphi)$ of host $\sigma_{\text{host}}(j,t,\varphi)$, and a starting point function st expressing that $\mathcal{T}_{j,t}$ starts at time $st(j,t)$, while respecting the phases sequence order, the IT resources and the power constraints.

The problem can then be formulated as follows: maximize $\sum_j Profit_j$, while fulfilling the power constraints. The profit $Profit_j$ for the cloud provider from a user $\mathcal{U}_j$ is obtained as follows: we compute the price that $\mathcal{U}_j$ should pay to run its job $\mathcal{J}_j$ on a reference hardware, provided in the task description, and priced according to Section 3.6. In case of degradation, a fraction of the profit is returned to the user according to actual $QoSs_j$ and $QoSb_j$, and the compensation tables presented previously. Note that maximizing the $Profit_j$ in this case is equivalent to minimize the amount of credit/compensation given to a user $\mathcal{U}_j$. The electricity does not appear in the bill since it comes exclusively from renewable energy sources, considered at cost 0 after the installation.

5.4 Proposed Approach

In this section we present the problem resolution, starting by the slots structure which we utilize for time aggregation, followed by the approaches to define the tasks start time, how to degrade phases tasks, and finally we present resources

assignment algorithms both from the literature, adapted to get along with phases, and new approaches.

We focus on non exact optimization methods, more specifically we explore Greedy Heuristics as a way to validate our proposal with a time window approach and an off-line resource allocation problem with a fixed set of tasks. The difference from the previous approaches is that in this case, we need to handle both batch tasks and services, while considering the power envelope as a constraint and the multiple phases of each task. The utilization of phases increases the resources control complexity, specially considering the quick changes that can occur in the resources utilization in a small time interval. In order to address these quick changes in resources utilization and reduce the computation time necessary to verify the resources constraint we utilize time-slots. In the end of this chapter we will also present the steps towards an alternative without utilizing time-slots.

5.4.1 Slots Aggregation

To reduce the complexity of the resources as constraint verification, each task is mapped in slots when scheduled. This process is illustrated in Figure 5.1, where several changes on the resources utilization occur (different $\mathcal{TP}_{j,t,\varphi}$) and after scheduling we consider the maximum of these changes on a given time interval. We consider each slot as a constant time interval where one or more phases will be executed and the resources consumption is fixed equal to the maximum of those phases. The aim is to aggregate in case we have several phases that are in the millisecond or second scale. The size of the slots can be changed according to the desired accuracy, varying from milliseconds to minutes.

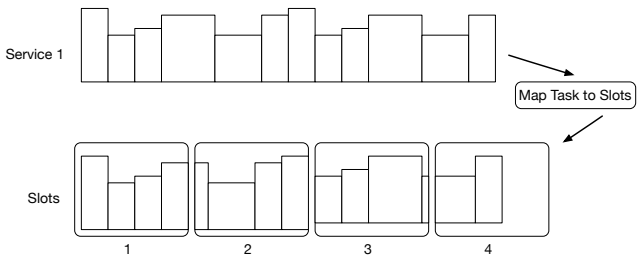


Figure 5.1: Example of a service mapped into slots, where we consider the maximum of resources used on each core for each slot.

In Figure 5.2 we can see the representation of 3 slots and some of the resources that are controlled inside each slot. In this illustration we display a single task $\mathcal{TP}_{1,1,}$ with several phases φ utilizing two cores. The main resources

are the available power, utilized power and the amount of memory and CPU consumed. We are representing one node with 2 cores, where the red line in each core indicates the maximum of CPU percentage consumed on that given slot. Without the aid of time slots, the size of the structure necessary to control the IT constraints would vary according to the number of changes in resources of all tasks.

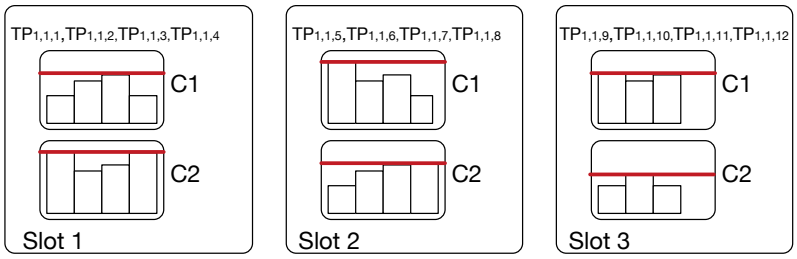


Figure 5.2: Slots representation in an infrastructure with 1 node and 2 cores with different task phases and the maximum of resources considered.

Based on the slots to verify all the constraints and resource allocation we introduce new scheduling approaches for phases based batches and services. We focus on non exact optimization methods, more specifically we explore Greedy Heuristics as a way to validate our proposal with a time window approach and an off-line resource allocation problem with a fixed set of tasks. The difference from the previous approaches is that in this case, we need to handle both batch tasks and services, while considering the power envelope as a constraint and the multiple phases of each task. To do so, the implemented algorithms that utilize the previously presented slots. Hereafter, we present our new approaches as well as algorithms from the literature, adapted to schedule phases based tasks.

We start presenting the start time search approaches, followed by classical scheduling algorithms, as well as our own adaptations. The proposed algorithms utilize the previously presented slots where the power envelope determines the maximum power available for each slots. We consider the bag of tasks, with a mix of batch and services, sorted by profit in order to schedule the tasks that have the biggest possible profit first. The initial profit is calculated as if the tasks had no degradation or violation.

5.4.2 Start Time Search Approaches

To find the start time of each batch task we consider three variations named: (i) First; (ii) MaxE; and (iii) MaxCCPC. Services only require the decision on which core to execute and how much resources are utilized, since the start time cannot be changed.

The “First” variation refers to a search from the release time of a task in a sequential way through the slots until a start slot with enough CPU, Memory and Power is found. In the second variation “MaxE” we aim to find the possible start slots that have enough power to run the task, and we sort the slots in descending energy available order. To do so we calculate the expected dynamic power consumption of a task if the requested resources are delivered, then we search in the power curve what are the start points that could put up this amount of power (not violate the power curve), and sort them from maximum to minimum energy to later select the node and validate CPU and memory constraints.

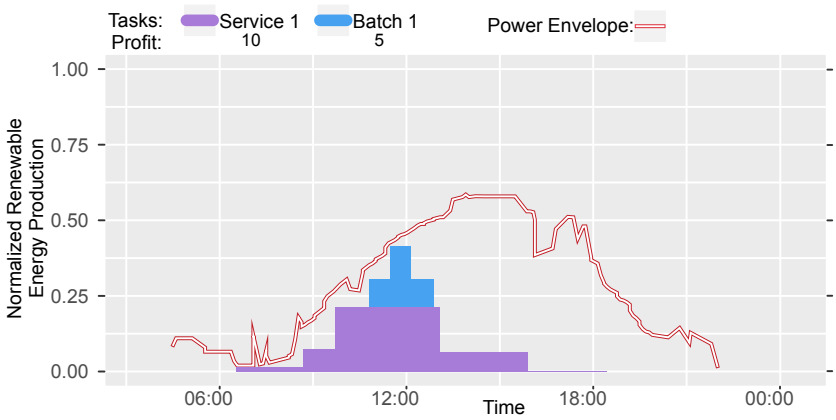


Figure 5.3: Scheduling time selection "First" available place.

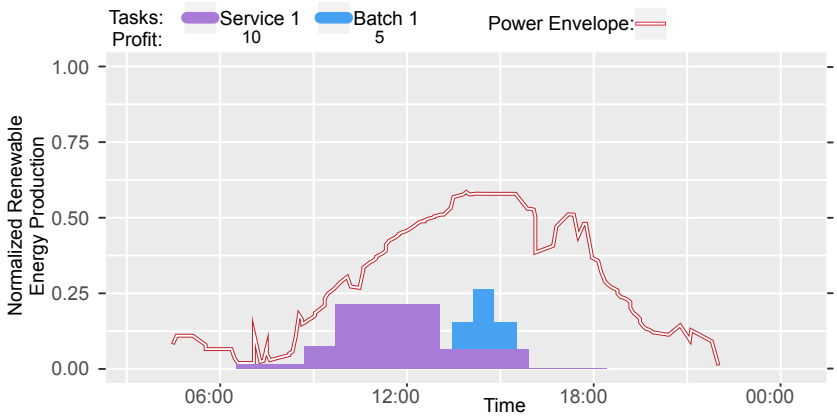


Figure 5.4: Scheduling time selection "MaxE" available place.

In Figure 5.3 and 5.4 we illustrate the differences between the “First” approach and the “MaxE” respectively. In the example we consider that the

release and due date of the batch task would be within the represented time window. The service does not allow time changes, according to the introduced model. In the "First" case we select the first available slot where there would be enough energy to place the task. In the second case ("MaxE") we place the task from the point where the maximum amount of energy is available. In this case, if new tasks arrived, the new maximum would be calculated subtracting the energy that was already spent with the placement of previous tasks.

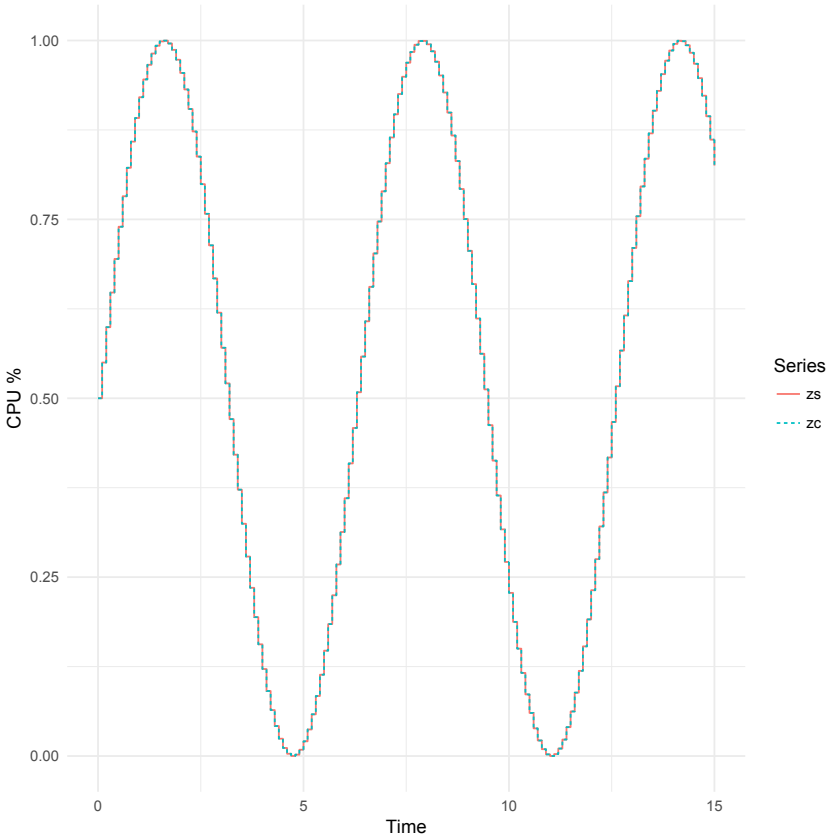
In the new approach proposed "MaxCCPC", we utilize normalized cross-correlation [17, 122], traditionally utilized in signal processing field, which indicates the degree of similarity between two time series in different times, considering its displacement (lag), where a positive value indicates a higher similarity between the two time series, and a negative value the opposite. In this approach we will utilize the cross-correlation concept in two parts: (i) to find the highest cross-correlation value between the power consumption profile of a task and the power envelope (represented as two time series); and (ii) to find the smallest cross-correlation between the task that is being placed and the current load of a given PE

In Equation 5.6 the cross correlation cc at delay d is defined. In this case mx and my are the means of the corresponding series. Considering that this is computed for all delays $d = 0, 1, 2, ...N - 1$ (where N is the number of intervals in the time series) then it results in a cross correlation series of twice the length as the original series. An example in Figure 5.5 presents in (a) two identical time series where the resources consumption follows a sinusoidal curve. In (b) we show the cross-correlation as the tasks slide in the time domain (lag). We can see the reduction of the cross-correlation as we increase the lag, where the minimum is reached near 30 or -30. This indicates a high correlation but of the inverse of one of the series.

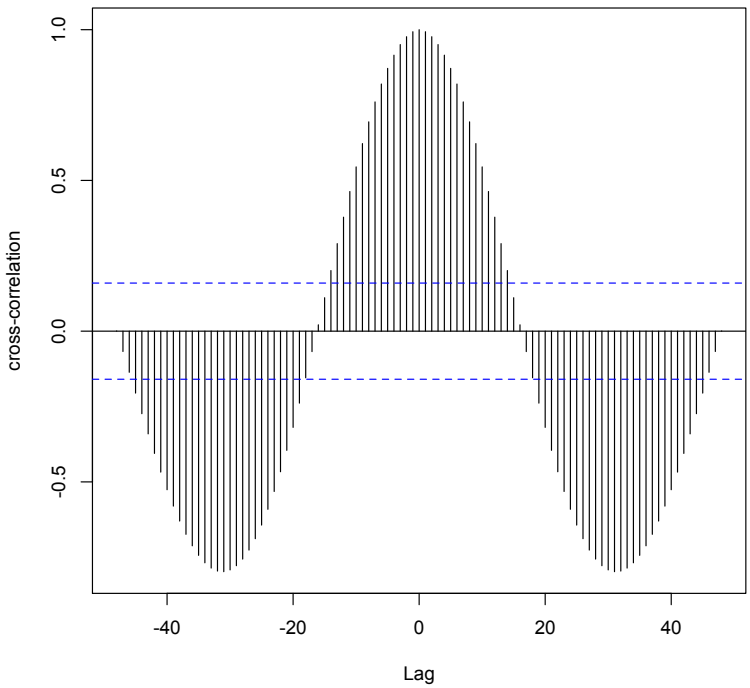
$$cc(d) = \frac{\sum_{i=1}^{N}[(x(i) - mx) \times (y(i - d) - my)]}{\sqrt{\sum_{i=1}^{N}(x(i) - mx)^2)\sqrt{\sum_{i=1}^{N}(y(i - d) - my)^2}} \tag{5.6}$$

For our phases tasks, in Figure 5.6 we can see them represented as two time series (a) and their cross-correlation (b) with different time shifts (lags). The minimum cross-correlation value is obtained with almost no shift between the two time series (in this case with lag 2s), where the matches between the peaks of resource consumption in the two time series occur the least. As we increase this value, we can find points where the majority of these peaks occur on the same time (positive cross-correlation). In this case, we aim to find in decreasing order the cross-correlation between the CPU usage (responsible for the major part of the power consumption) and the power curve, to match as much as possible the CPU usage peaks with the power available.

In Figure 5.7 we present the placement considering the “MaxCCPC” where the task having a low/high/low pattern is placed where there is the same pattern in the power curve. In other words we make the cross-correlation between the power consumption of the task and the power envelope. Also, note that the service is not displaced, since the time shift is not possible for this type of task.

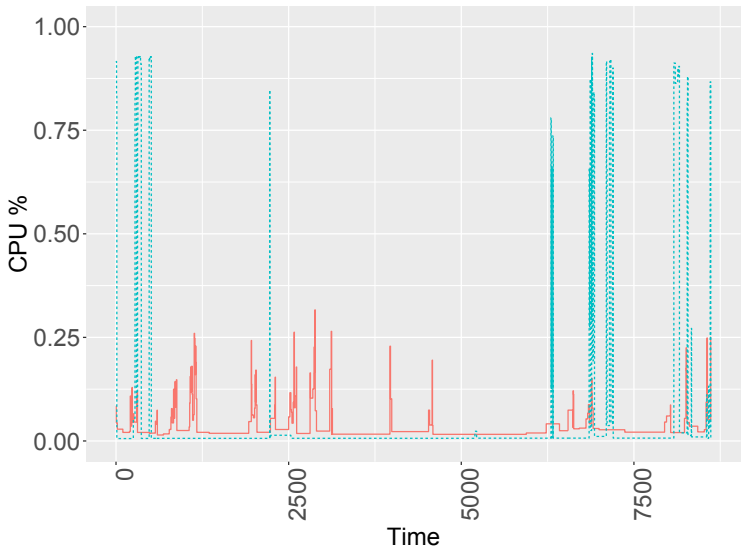


(a) Time Series

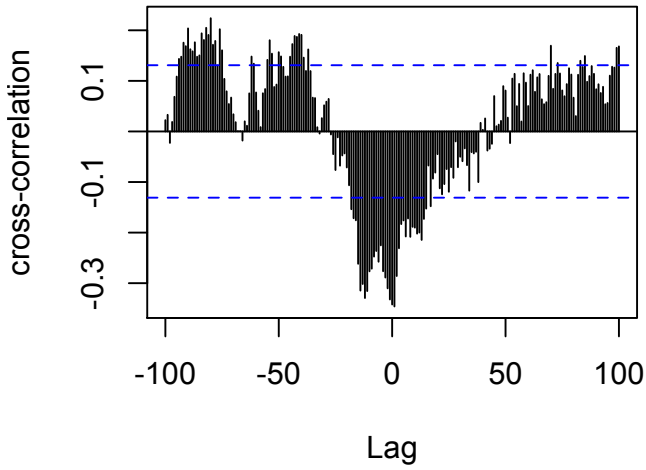


(b) Cross-Correlation

Figure 5.5: Cross-Correlation with lag example of sinusoidal time series.



(a) Time Series



(b) Cross-Correlation

Figure 5.6: Cross-Correlation with lag example of two tasks with the same resource consumption represented as time series.

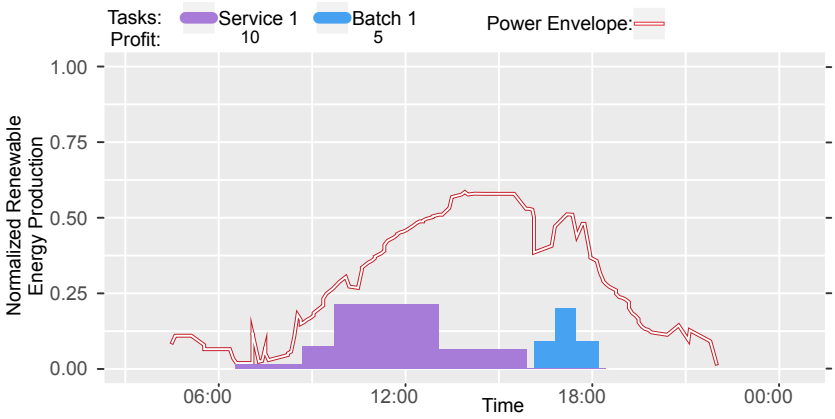


Figure 5.7: Scheduling time selection “MaxCCPC” available place.

5.4.3 Phases Degradation

The concept of phases degradation, which was present previously in subsection 3.2.3 is explored in this problem resolution. Here we explore degradation when some of the constraints would not be satisfied, i.e. when there is not enough power to place the task in a given moment, or the IT resources are insufficient. In regular cloud providers IT performance reduction could be experienced when the overcommited resources of a shared physical machine are utilized at its full for instance.

The function presented in Algorithm 5 is utilized when the resources assignment without degradation is not possible. It allows us to degrade some phases of a task in order to schedule it on a core/node that initially would not be possible. The objective is to degrade as little as possible, in order to reduce the impact in QoS and consequently the profit degradation.

Initially in line 1 the aim is to sort the slots from the release time to the due date by the amount of resources available, in order to cause less degradation as possible. In line 7 we highlight that at this point we need to make a distinction, since batch tasks when degraded cause a change in the execution time. This means that Batch requires the re-validation of all the constraints (CPU, memory), since the violations might not be the same as before. In line 11 we search for the resources available on a given time for the chosen PE (Processing Elements), which will then be utilized in line 12 to degrade the task. Here we set the resources consumption to the maximum possible (according to either power or IT resources available). This process is followed by a validation of the constraints in line 13 which returns valid, or the constraint that was violated.

For Service in line 17 we can directly reduce the resources without recalculate the time according to our task model. If no constraint is violated after the degradation in line 25 we will have as return: the new valid start point and cores, or null if not possible even degrading this task. Again, we highlight that the objective is always to find the point with the minimum degradation/violation reducing the impact in the QoS and by doing so also reducing the impact in the profit loss.

Algorithm 5: Pseudo-code of the algorithm that performs the degradation of task phases.

```
Data: Current Task, Nodes, Slots.
Result: The task degraded and ready to be scheduled or a remove task order.
startSlotPE ← getMaxPowerPE(task,node,slots);
if startSlotPE==null then
    return null;
end
slotStart ← startSlotPE.getStartSlot();
PEChosen ← startSlotPE.getPE();
if task.taskType == Batch then
    while invalidConstraint or allPhasesDegraded do
        for slot = slotStart; slot<slotStart+task.size ; slot++ do
            powerAvailable ← slots.getPowerAvailable(slot);
            PEAvailable ← slots.getPEAvailable(slot,PEChosen);
            degradeToMaximum(task.getPhasesInSlot(slot), PEAvailable);
            invalidConstraint← validateConstraints(task,PEChosen,slotStart,slots);
        end
    end
else
    for slot = slotStart; slot<slotStart+task.size ; slot++ do
        powerAvailable ← slots.getPowerAvailable(slot);
        PEAvailable ← slots.getPEAvailable(slot,PEChosen);
        degradeToMaximum(task.getPhasesInSlot(slot), PEAvailable);
        invalidConstraint← false;
    end
end
if !invalidConstraints then
    return startSlotPE;
end
return null;
```

5.4.4 Resources Assignment Algorithms

Hereafter we present the greedy heuristics utilized to schedule the tasks aiming at maximize the profit. First we will introduce the classical algorithms from the literature that have been adapted to consider the mix of phases based batch and services as well as the profit calculation. Next, we present algorithms that schedule making a distinction between batch and services. We then introduce our approach utilizing cross-correlation to schedule these tasks. Finally, we adapt some of the algorithms from [67] to support phases based tasks, since it is one of the approaches in the literature that considers datacenters powered only by renewable energies.

Classical Approaches Adapted

For the literature approaches we consider a set of algorithms that we call "classical", being First Fit (FF), All Tasks Maximum Energy (AMaxE). None of these approaches makes distinction between batch and services when scheduling the tasks. Then we have Service First Maximum Energy (SFirstMaxE) and Batch First Maximum Energy (BFirstMaxE) where we schedule first either all service tasks or all batch tasks.

We start by detailing the FF algorithm to facilitate the comprehension, since the other mentioned ones follow similar principles. We also highlight that this approach also utilizes degradation and slots. We start by creating the slots structure and then sorting the tasks in lines 1 and 2 by profit without making distinction between service and batch. In line 6 we check if some node is on, or if task requests a new node (the aim is to keep the number of nodes on to the minimum number). We then get in line 10 the PE (Processing Element, in our case cores valid), that do not violate the constraints. This search starts from the minimum start slot until the end of the window. If the interval is not valid the return is empty. The main difference to a classical first fit occurs in line 15 were if we do not have power or resources to add the task we take the decision to degrade or increase start slot. In case of service the only option is to degrade since the start slot cannot be changed.

To illustrate how the placement utilizing the algorithms would be, we present a list of tasks in Figure 5.8 where we have 3 phases based batch tasks and 1 service with phases. We show the duration of each task, and its normalized energy consumption. In this case we consider that the batch tasks can be scheduled at any time during the given time window and the service has to start at 6:30 a.m. In Figure 5.9 we show how the placement utilizing FF of the aforementioned tasks would appear. In this case we display only the power consumption under a given power curve, assuming that the IT resources

Algorithm 6: Phases based service and batch ordered by profit first fit (FF) scheduling algorithm.

```
Data: Phases Tasks, Nodes, Power Profile, Slot Duration.
Result: Tasks execution plan.
slots ← createSlots(powerProfile, durationSlot);
sortByProfit(tasksReceived);
for t in tasksReceived do
    startSlot ← t.minimumStartSlot();
    while !scheduled do
        if needNewNode or nodesOn.isEmpty() and !addedNodeForThisTask then
            addNewNode(nodesOn);
            addedNodeForThisTask ← true;
        end
        validCores ← getValidCores(t,nodesOn,startSlot,slots);
        if validCores.isEmpty() then
            if havePower() and nodesOn < Nodes and !addedNodeForThisTask then
                needNewNode ← true;
            else
                degraded ← degradeOrIncreaseSlot(t,nodesOn,slots);
                if degraded then
                    t.schedule(coresSelected,startSlot,slots);
                    scheduled ← true;
                else
                    if startSlot+1 > windowSize then
                        t.remove();
                        break;
                    end
                    startSlot++;
                end
            end
        else
            coresSelected ← validCores.get(t.getCoresNumber);
            t.schedule(coresSelected,startSlot,slots);
            scheduled ← true;
        end
    end
end
```

constraint would be satisfied. The service 1 is placed first, since it has the highest profit, followed by Batch 2 and 3 which are placed in the first place where there is enough power. Finally, Batch 1 is placed earlier, since it has a lower power consumption.

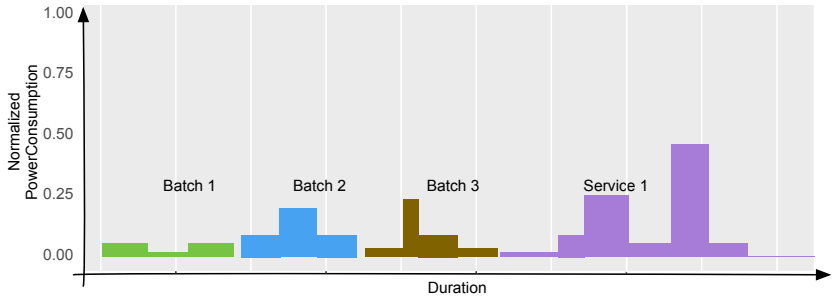


Figure 5.8: List of tasks utilized in the examples.

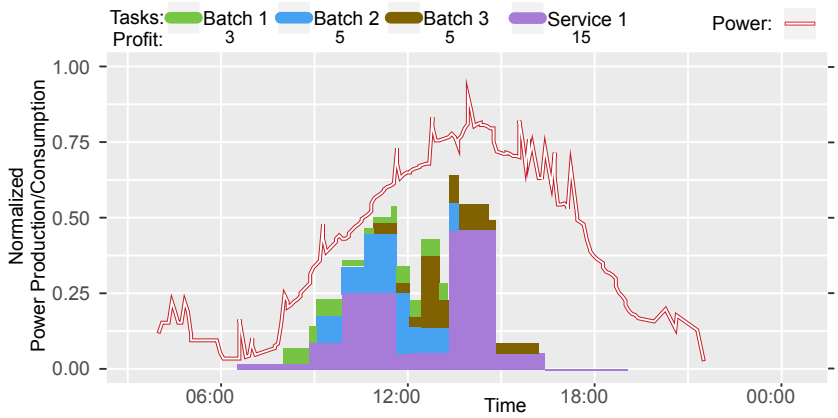


Figure 5.9: Scheduling illustration of a list of tasks utilizing FF algorithm under a power constraint.

The other algorithm that does not make distinction between the type of task is AMaxE, presented in Algorithm 7. In line 5 is where we get the slots from maximum energy available to minimum. This is done according to the expected duration of the task in a homogeneous node, delivering the requested resources. According to this we estimate what are the slots where the task would fit and add them in order from the one with maximum energy available to the minimum. The order is first the slots before violating, then the ones where the task would be violated in sequence. Later in line 8 we verify if some node is on, or if task requests a new node. Line 12 is where all the valid PEs are retrieved, meaning that they do not violate constraints from the start slot

Algorithm 7: Phases based service and batch ordered by profit maximum energy scheduling algorithm.

```
Data: Phases Tasks, Nodes, Power Profile, Slot Duration.
Result: Tasks execution plan.
slots ← createSlots(powerProfile, durationSlot);
sortByProfit(tasksReceived);
for t in tasksReceived do
    startSlot ← t.minimumStartSlot();
    slotsEnergy<Power,startSlot> ←
    getMaxEnergySlots(t.durationSlots,slots)
    while !scheduled do
        startSlot ← slotsEnergy.getNext();
        if needNewNode or nodesOn.isEmpty() and
        !addedNodeForThisTask then
            addNewNode(nodesOn);
            addedNodeForThisTask ← true;
        end
        validCores ← getValidCores(t,nodesOn,startSlot,slots);
        if validCores.isEmpty() then
            if havePower() and nodesOn < Nodes and
            !addedNodeForThisTask then
                needNewNode ← true;
            else
                degraded ← degradeOrIncreaseSlot(t,slots);
                if degraded then
                    t.schedule(coresSelected,startSlot,slots);
                    scheduled ← true;
                else
                    if slotsEnergy.getNext() > windowSize then
                        t.remove();
                        break;
                    end
                    startSlot ← slotsEnergy.getNext();
                end
            end
        else
            coresSelected ← validCores.get(t.getCoresNumber);
            t.schedule(coresSelected,startSlot,slots);
            scheduled ← true;
        end
    end
end
```

until the expected end of task. If we don't have any valid PE the return will be empty. In line 17 is where the degradation can occur following the same principle as the previous algorithm. In case of service the only option is to degrade. Finally in line 22 if a task cannot be executed in this window, even considering degradation, it will be removed.

Again we illustrate a simple placement utilizing the same list of tasks shown in Figure 5.8. In Figure 5.10 we show how the placement utilizing AMaxE, presenting only the power consumption under a given power curve, assuming that the IT resources constraint would be satisfied. Again, service 1 is placed first since it has the highest profit, followed by Batch 2 which is placed approximately in the middle of the time power curve. We then place Batch 3, but in this case the task is placed towards the end of the window, since it is now the place with more energy available. Finally, Batch 1 is placed earlier, since it is the place with the highest amount of energy where the task would fit.

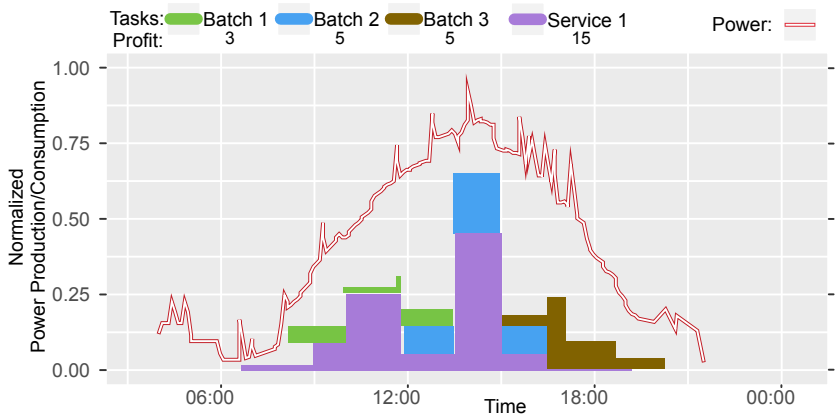


Figure 5.10: Scheduling illustration of a list of tasks utilizing AMaxE algorithm under a power constraint.

Considering the algorithms that make distinction between Batch and Service we have two variations: (i) BFirstMaxE which schedules batch tasks ordered by profit first and utilizing "MaxE" to find the start time; and (ii) SFirstMaxE which schedules services ordered by profit first and utilizing "MaxE" to find the start time. In this case the algorithm would be equals to the one presented in Algorithm 7 with the one exception that in line 2 there would be two separated lists of tasks. To illustrate how that could lead to significantly different results we show in Figure 5.11 the scheduling of the same tasks, but in this case placing the batch tasks first (BFirstMaxE). Using this variation in this specific case would lead to a degradation in Service 1, since it cannot be displaced in time.

We highlight it in red at 15:00 p.m. where there wouldn't be enough power to place the task without degrading it (or changing the batch placement).

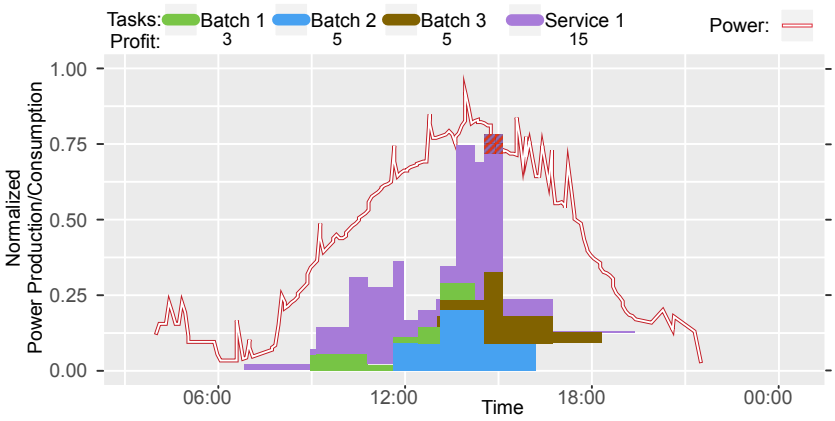


Figure 5.11: Scheduling illustration of a list of tasks utilizing BFirstMaxE algorithm under a power constraint.

Cross-Correlation based approaches

Regarding the cross-correlation approaches, we have three variations, one utilizing each time choice alternative: (i) MinCCMaxE; (ii) MinCCFirst; and (iii) MinCCMaxCCPC. Algorithm 8, called MinCCMaxE, as the name suggests utilizes the previous approach named "MaxE", starting the search for a start point where the maximum energy is available. In this algorithm, we use cross-correlation in line 8 ("getSmallestCrossCorrelationProcessor") to find the PEs where the current load presents the minimum cross-correlation with load of the task that we want to schedule. To do so, we represent the task and the load of each core as a time series. In this case we do not use the lag function, since the start time is already defined. We calculate the cross-correlation with all the PEs of nodes that are currently on at the moment and find the node with enough PEs and memory and smallest cross-correlation of the resources in use.

We present the cross-correlation choice (Figure5.12)utilizing the task list presented in Figure 5.8. In this case we place Batch 3 along with Service 1, and Batch 1 along with Batch 2, since the resources consumptions at those given points have the minimum cross-correlation as presented previously.

If the return is null it means that there is no power in any start point. If the return is empty it means that there is still power but no node available, which may be added in line 27. If we cannot find such set of PEs we apply in line 18 the slow down/change of resources or increase the start slot (Algorithm 5).

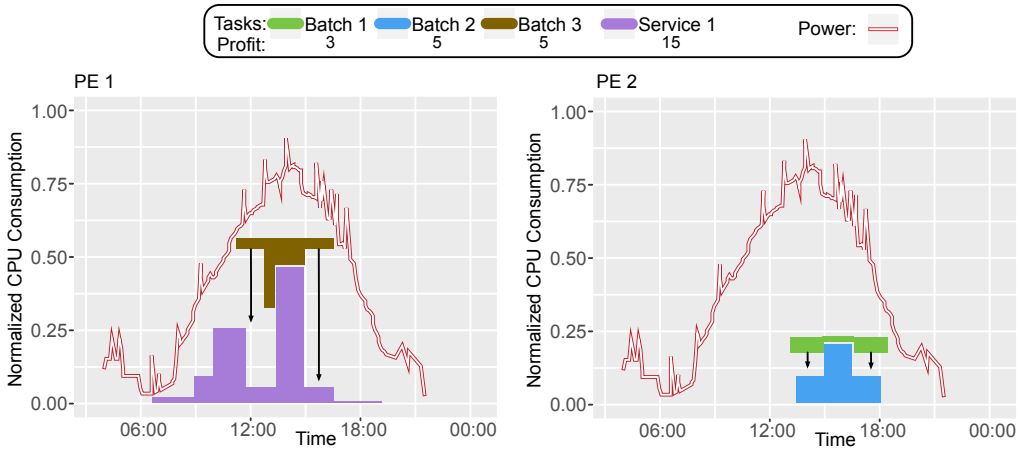


Figure 5.12: Scheduling illustration of a list of tasks utilizing cross-correlation between two PEs.

Services may only be degraded; batch tasks can receive less resources (slowed down) without QoS degradation (they may still finish before the due date). If the batch slowed down leads to QoS degradation, we search for a new start time.

If the return is null means that there is no power in any start point or empty if there is still power but no node available, which may be added in line 27. If we cannot find such set of PEs we apply in line 18 the slow down/change of resources or increase the start slot (Algorithm 5). Services may only be degraded; batch tasks can receive less resources (slowed down) without degradation (finishing after the due date); if not, we search for a new start time. Other case that can occur is in line 26 where we have no core that match constraint, but still have power to switch on more nodes.

We also propose two other variations that utilize cross-correlation following similar principles as the ones in the previous algorithms:

- The first one is MinCCFirst where the time selection is at the first slot that does not violate the constraints, not differing significantly from the previous algorithm;
- The second variation is MinCCMaxCCPC presented in Algorithm 9. We also use the cross-correlation to find the point where the task's power consumption is as similar as possible with the power curve.

For Algorithm 9, in line 6 is where we get the slots from maximum correlation with the energy available to minimum, according to the expected number of slots used by the task in one of the homogeneous nodes giving the requested

Algorithm 8: Pseudo-code of phases based scheduling algorithm MinC-CMaxE.

```
Data: Phases Tasks, Nodes, Power Profile, Slot Duration.
Result: Tasks execution plan.
slots ← createSlots(powerProfile, durationSlot);
sortByProfit(tasksReceived);
while !tasksReceived.isEmpty do
    startSlot ← t.minimumStartSlot();
    t ← tasksReceived.popFirst();
    slotsEnergy<profilePower,startSlot> ← getMaxEnergySlots(t.durationSlots,slots);
    for currSlot in slotsEnergy do
        validCores ←
        getSmallestCrossCorrelationProcessor(t,nodesOn,startSlot,slots);
        if validCores!=null then
            break;
        end
    end
    if validCores==null and tasksTested < totalTasks then
        tasksReceived.addLast(t);
        tasksTested++;
    end
    else if validCores==null or validCores.isEmpty() and tasksTested > totalTasks
    then
        changeSuccessful ← degradeOrIncreaseSlot(t,nodes,slots);
        if changeSuccessful then
            t.schedule(t.coresSelected,t.startSlot,slots);
        else
            t.remove();
        end
        tasksTested++;
    end
    else if validCores.isEmpty then
        addNewNode(nodesOn);
    end
    else
        schedule(t,validCores,startSlot);
        tasksTested++;
    end
end
```

Algorithm 9: Pseudo-code of phases based scheduling algorithm MinC-CMaxCCPC.

```
Data: Phases Tasks, Nodes, Power Profile, Slot Duration.
Result: Tasks execution plan.
slots ← createSlots(powerProfile, durationSlot);
sortByProfit(tasksReceived);
while !tasksReceived.isEmpty do
    startSlot ← t.minimumStartSlot();
    t ← tasksReceived.popFirst();
    slotsEnergy<CrossCorrelation,startSlot> ←
    getMaxCrossCorrelationSlots(t.durationSlots,slots);
    for currSlot in slotsEnergy do
        validCores ←
        getSmallestCrossCorrelationProcessor(t,nodesOn,startSlot,slots);
        if validCores!=null then
            break;
        end
    end
    if validCores==null and tasksTested < totalTasks then
        tasksReceived.addLast(t);
        tasksTested++;
    end
    else if validCores==null or validCores.isEmpty() and tasksTested > totalTasks
    then
        changeSuccessful ← degradeOrIncreaseSlot(t,nodes,slots);
        if changeSuccessful then
            t.schedule(t.coresSelected,t.startSlot,slots);
        else
            t.remove();
        end
        tasksTested++;
    end
    else if validCores.isEmpty then
        addNewNode(nodesOn);
    end
    else
        schedule(t,validCores,startSlot);
        tasksTested++;
    end
end
```

resources. Similar to the previous approach, in line 8 "getSmallestCrossCorrelationProcessor" returns a set of PEs such that (i) the constraints are fulfilled and (ii) the cross-correlation is the minimum between the load of the task and the load of the processors. To do so, we represent the task and the load of each core as a time series. If the return is null it means that there is no power in any start point. If the return is empty it means that there is still power but no node available, which may be added in line 27. If we cannot find such set of PEs we apply in line 18 the slow down/change of resources or increase the start slot (Algorithm 5). Services may only be degraded; batch tasks can receive less resources (slowed down) without QoS degradation (they may still finish before the due date). If the batch slowed down leads to QoS degradation, we search for a new start time.

DataZERO Adapted Algorithms

To compare the cross-correlation approaches not only with classical algorithm approaches, but also with approaches from the literature that consider a limited power envelope (also in DataZERO project) we also adapted some of the algorithms of Kassab et al. [67]. In this case we re-implemented the variations LPN, LPT, SPT and LPTPN which are the approaches that obtained the best results in the majority of the test cases proposed by the authors. The algorithms have been adapted to utilize slots in order to schedule phases based tasks.

These approaches do not sort the tasks by profit anymore. Following the algorithms list we have (all in a "First" available start time way): (v) LPN which schedules the tasks from the ones with the largest power needed to the ones with the lowest; (vi) LPT which goes from the largest processing time to the lowest; (vii) SPT which schedules the shortest processing time tasks first; and (viii) LPTPN which schedules first the tasks with largest processing time $\times$ power needed (total energy) first.

5.5 Evaluation Methodology

In this section we explain what is the methodology adopted to test all the algorithms variations over different workloads (number of batch and services) and power profile constraints (different sunlight/day). This section is followed by the results evaluation, not only in the profit metric but also a view from the user side, taking a deeper look in the QoS over the different workloads.

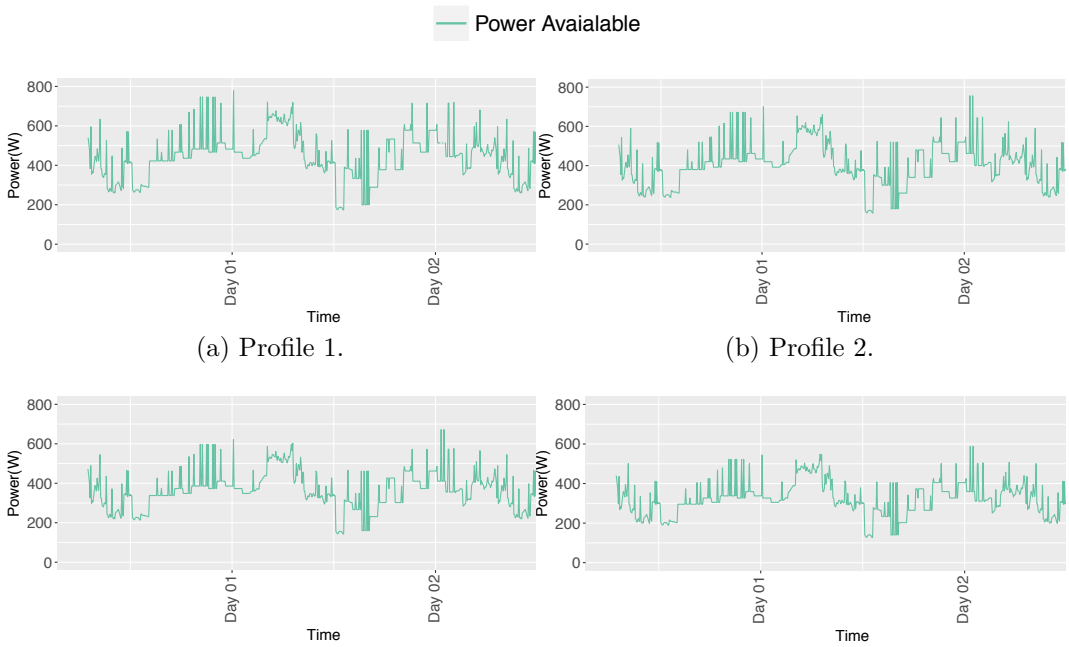
Here we also simulated the IT infrastructure using the DCWoRMS simulator [76] through several adaptations in order to support the new phases based tasks and shared resources (more than one task on each PE). We consider each PE is a core inside a processor, where its usage can vary which could be obtained through cgroups for instance. The slot duration is fixed to 5 minutes, which is based on the minimum phase duration of our workload. The time window considered is of 2 days.

The IT infrastructure inside the simulator is based on Villebonnet et al. [123], more specifically we are using 10 homogeneous hosts and the power consumption values of Paravance cluster from Grid5000[1]. The values for Paravance utilized in the power model presented previously are: $P^{(\mathrm{dyn})} = 0.5904\,W$ (note that in this case $P^{(\mathrm{dyn})}$ value is for each core) and $P^{(\mathrm{idle})} = 69.9\,W$. We also considered $P^{(\mathrm{on})} = 112.91\,W$ over $t^{(\mathrm{on})} = 189\,s$ and for $P^{(\mathrm{off})} = 65.7\,W$ over $t^{(\mathrm{off})} = 10\,s$. The values presented represent the node fully utilized (100%) and the dynamic

[1] https://www.grid5000.fr/

power varies according to the CPU percentage of each core and one node fully utilized results in a peak power consumption of 135.2 W.

The data source presented in Section 3.5 was utilized to generate 4 power envelopes. Each variations consist in reducing the wind power production by approximately 10% on each profile from the highest power production to the lowest we therefore call these profiles 1 to 4. The variations can be seen in Figure 5.13. To summarize we also present a list of all the algorithms that will be evaluated and its characteristics in Table 5.3. The details regarding each workload variation is presented in the next section.



(a) Profile 1.

(b) Profile 2.

(c) Profile 3.

(d) Profile 4.

Figure 5.13: Power production of renewable energy (wind and solar) utilized in the experiments.

Table 5.3: Summary of proposed algorithms characteristics.

Algorithm Name	**Sorting**	**Core Assignment**	**Start Time**
FF	Max. Profit	First	First
AMaxE	Max. Profit	First	MaxE
BFirstMaxE	Max. Profit Batch First	First	MaxE
SFirstMaxE	Max. Profit Service First	First	MaxE
MinCCFirst	Max. Profit	MinCC	First
MinCCMaxE	Max. Profit	MinCC	MaxE
MinCCMaxCCPC	Max. Profit	MinCC	MaxCC
LPN	Largest Power Need	First	First
LPT	Largest Processing Time	First	First
SPT	Shortest Processing Time	First	First
LPTPN	Largest Processing Time x Power Need	First	First

5.6 Results Evaluation

Hereafter we present the results obtained utilizing the variations of power and algorithms presented previously. We tested different workload variations, in order to evaluate different possibilities that could occur in a cloud datacenter. We will first present the results considering a balanced workload where the profit between batch tasks and services is similar. We then follow to an unbalanced workload where services tend to be more profitable (long running and more resources reserved) than batches, to represent a datacenter where the majority of the workload would be services. We then evaluate another unbalanced workload, but with different resources consumption for both batch and services to validate the behavior of the algorithm in similar scenario but different profit/resources consumption for the tasks. Finally we explore the impact of utilizing the concept of tasks degradation and evaluate how it impacts in the profit and QoS.

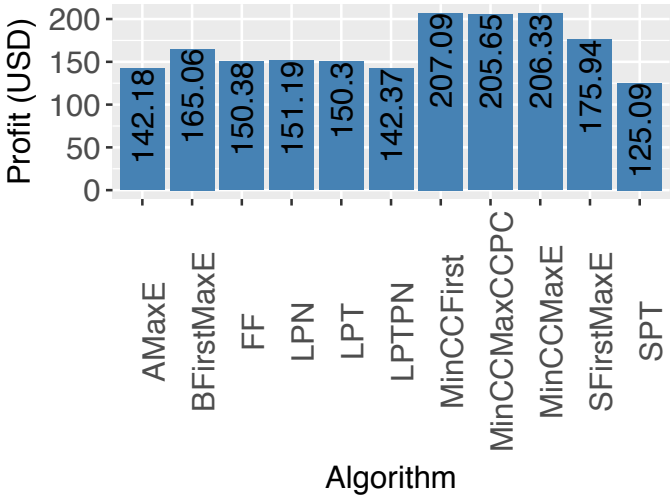
5.6.1 Workload With Balanced Profit

First we are going to introduce the results obtained with a workload variation of 20 services and 114 batch tasks. In particular these are longer time executing batch tasks which allow them to have similar profit when compared to the 20 services utilized. The maximum profit from Batch being $104.138 and from Service $109.629. The total profit is $218.775 where the remainder $ 5.008 comes from network usage. Values which have been obtained by all algorithms when removing the power profile constraints.

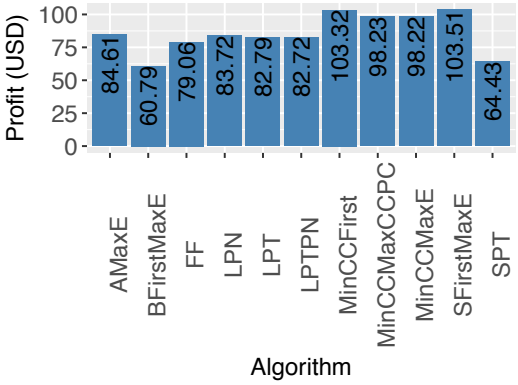
We start by evaluating the profit from each type of task obtained by the algorithms over the 4 different power profiles. Profit results are displayed in Figures 5.14, 5.16, 5.18 and 5.20. In these figures we can see that the cross-correlation variations in a balanced profit scenario obtained the highest total profit for all the cases. Nevertheless, if we look on the separated profits, i.e. profit from batch and profit from service, we can see that the highest batch profit and the highest service profit are always obtained by BFirstMaxE and SFirstMaxE respectively. The cross-correlation variations obtain a higher profit due to the fact that they are able to have the best "balanced" profit among the two types of tasks.

If we look at the QoS for each one of the cases, presented in Figures 5.15, 5.17, 5.19 and 5.21 we can see that the QoS variations can be much more severe, as low as 15.2% for services in BFirstMaxE for Profile 4. For SFirstMaxE we have 45.41% for the QoS of batch tasks, followed by 44% up to 42% for LPN, LPT and LPTPN. These degradations in QoS are just partially translated into profit, since the compensations offered by the defined model and also cloud

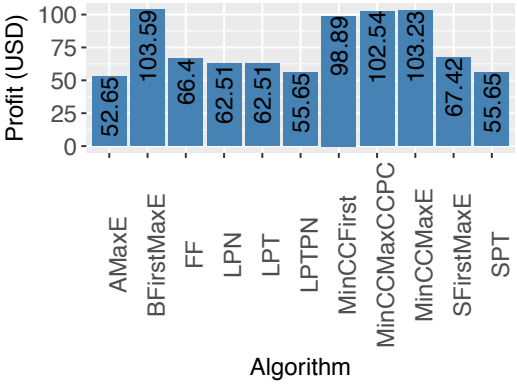
providers have lower bound. For services profit the lowest value profit is 50% for a given job where the user would receive a compensation and batch as low as 40%. There are possibilities that the profit could get lower than this as shown in the profit for service in Profile 4 for batch tasks, where 2 services where removed (QoS 0). We also need to highlight that the relation between average QoS and profit is not always direct since it depends on what tasks had been violated, and what was the profit of these specific tasks.



(a) Total Profit



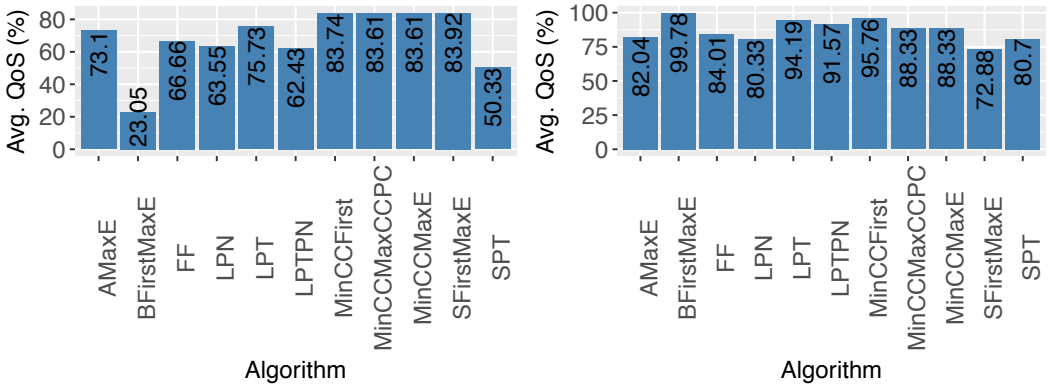
(b) Profit from Service



(c) Profit from Batch

Figure 5.14: Profit of workload with balanced profit for Batch and Service for all algorithms utilizing power profile 1.

The main reason why the cross-correlations have a better profit, even when compared to another algorithms that do not make distinction between batch and service, such as FF and AMaxE can be attributed to the PE choice.

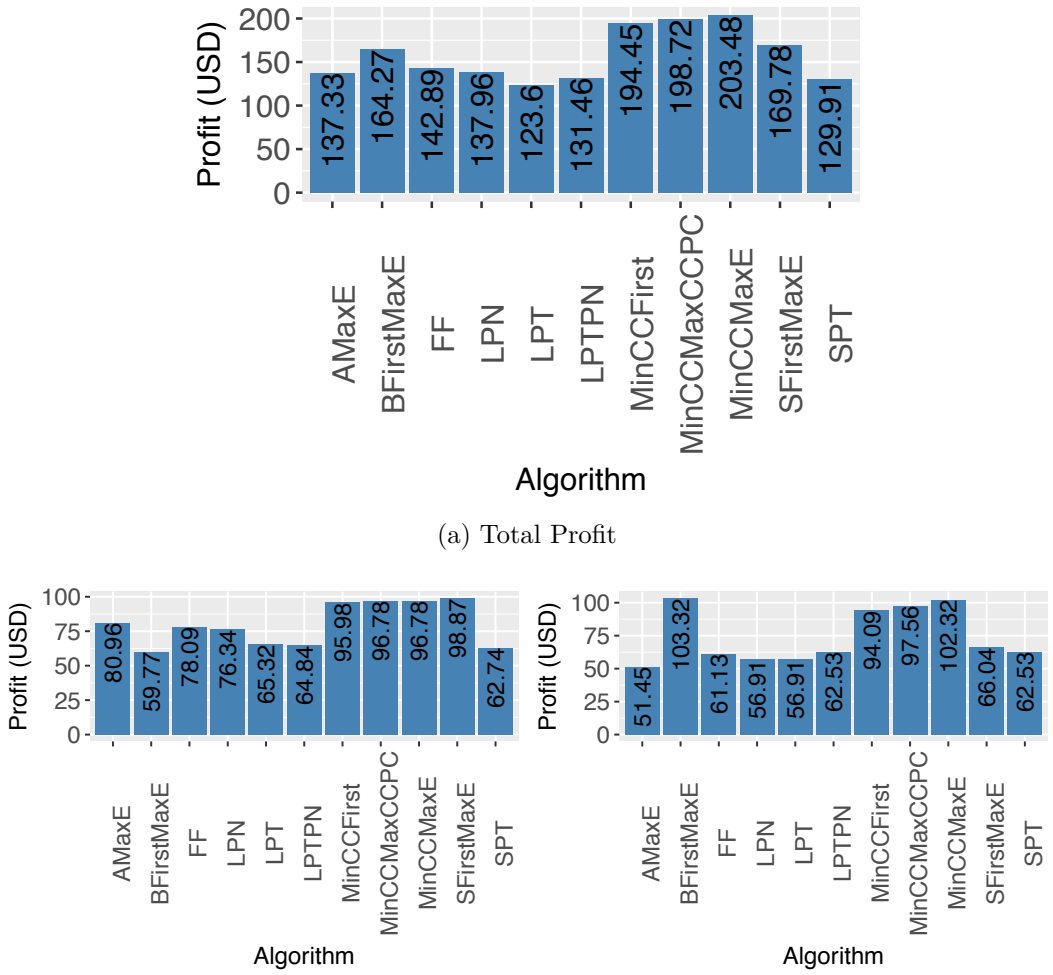


(a) QoS Service (b) QoS Batch

Figure 5.15: QoS of workload with balanced profit for Batch and Service for all algorithms utilizing power profile 1.

Algorithms such as FF and AMaxE place the tasks on the first PE available, with no consideration on what other tasks where placed and the impact that it could have later. On the other hand as shown in the illustration at Figure 5.12 the cross-correlation algorithms try to place the tasks among the PEs that would have the "complementary" workload. This leads to a higher average utilization of the PEs processing capability. We obtain an average of the PEs usage capability of 66% to 68% for the cross-correlation approaches, compared to 32% for the BFirst approach, 48% for FF, 43% for AMaxE and 55% for SFirstMaxE. For the adapted approaches we have 42% and LPTPN 24% for SPT. A lower value for average PE usage and lower profit signifies that the PEs have unused capacity, but due to bad placement the remained processing capacity is not enough to place a task without violating the constraints at some moment. This means that the cross-correlation approaches require less PEs to place the same amount of tasks, which can translate into a smaller energy consumption as well. Despite the relatively well utilization SFirst penalized batch tasks because of the time limitations to schedule the following without violating them. In other words, if a service is consuming all the energy in the time window where the batch could be scheduled, it is not possible to place it. We will see in the next set of experiments that it could perform better depending on the workload characteristics.

This can also be visualized when we compare the profit obtained by the algorithms and the corresponding power curves. In Figure 5.22 we display the power curves for FF and BFirstMaxE to highlight the difference between the PEs usage efficiency. We can see that the power consumption of FF



(a) Total Profit

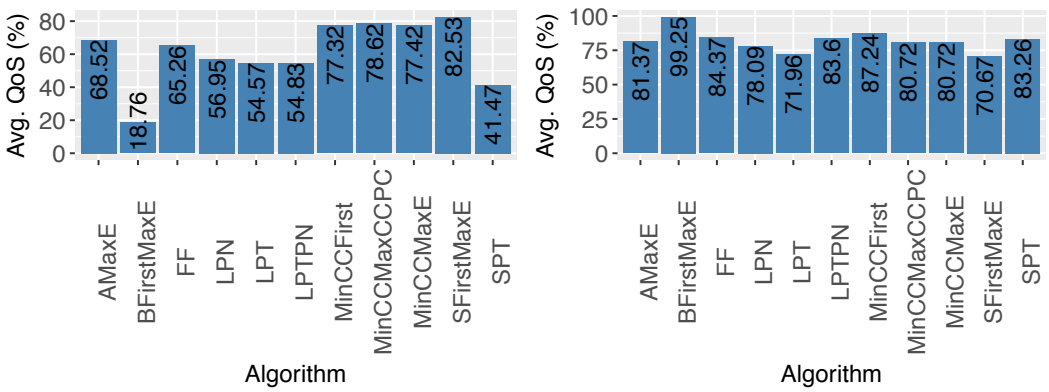
(b) Profit from Service

(c) Profit from Batch

Figure 5.16: Profit of workload with balanced profit for Batch and Service for all algorithms utilizing profile 2.

and BFirstMaxE is higher when compared to the ones in Figure 5.23 where we present the cross-correlation approaches. This means that the PEs on have a higher/more efficient utilization. The same applies for the ones in Figure 5.24 where LPTPN and SPT are displayed. As a drawback, this also means that cross-correlation approach might not be robust. Robustness [59] can be interpreted as the probability of a task to be affected by a failure of a component in the Cloud. In other words, if a node in the cloud fails, the average number of tasks impacted can be higher.

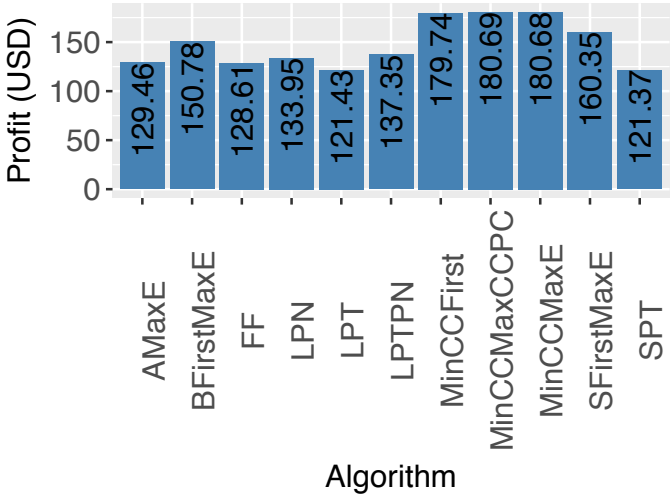
The main drawback of the cross-correlation approaches is in the execution time, as we can see in Figure 5.25. We can also see the increase that occurs as the



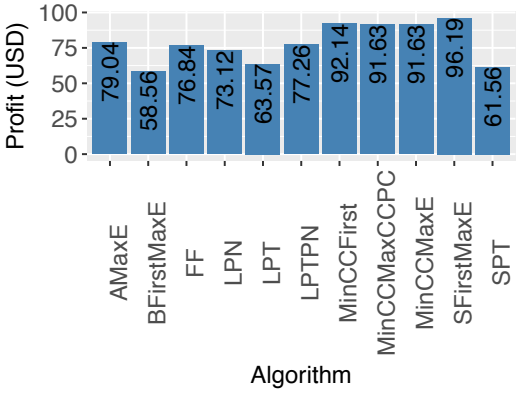
(a) QoS Service (b) QoS Batch

Figure 5.17: QoS of workload with balanced profit for Batch and Service for all algorithms utilizing power profile 2.

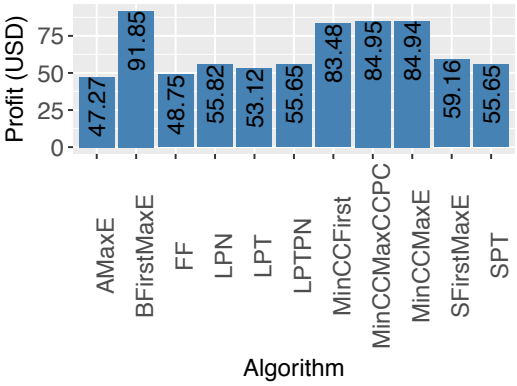
power gets more constrained. This is related to the higher number of interactions needed to find a suitable placement (without violating the constraints). Cross-correlation approach is time consuming and it can increase even more as the slots size decreases, since the calculation according to Equation 5.6 occurs for every time interval of the time series. However, this time is for scheduling 2 days of workload.



(a) Total Profit

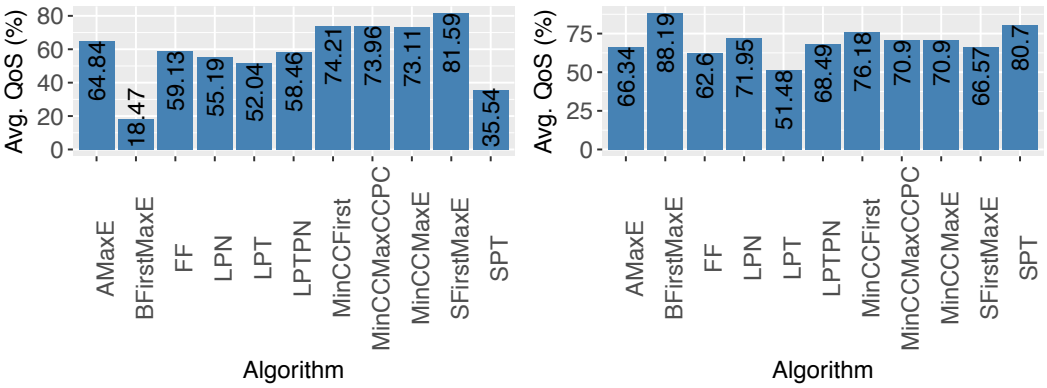


(b) Profit from Service



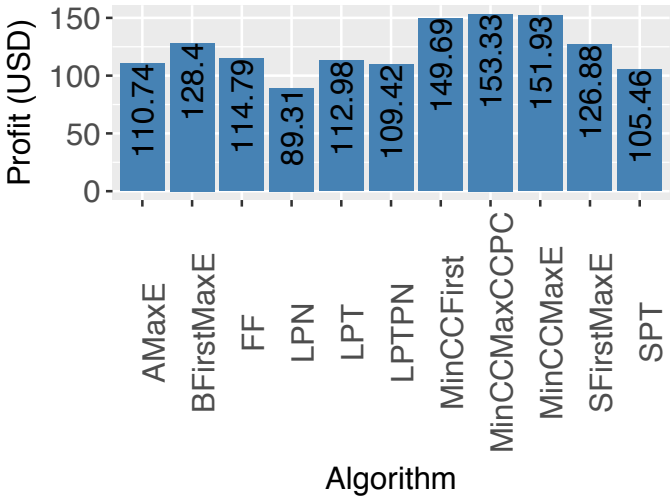
(c) Profit from Batch

Figure 5.18: Profit of workload with balanced profit for Batch and Service for all algorithms utilizing power profile 3.

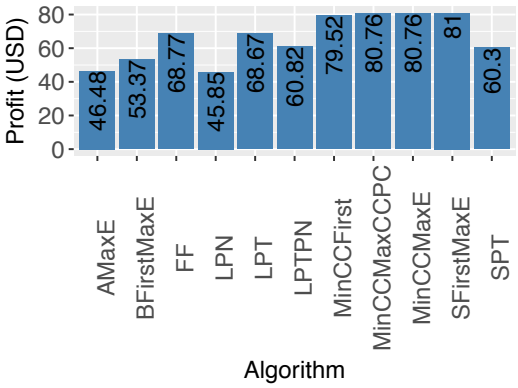


(a) QoS Service (b) QoS Batch

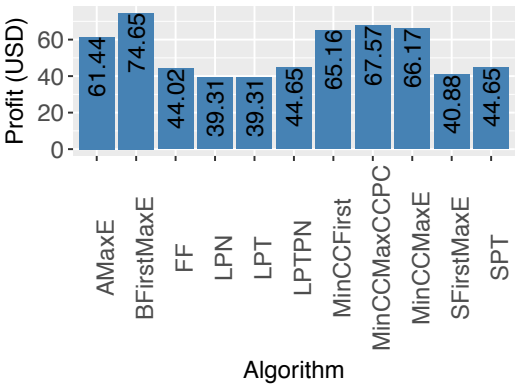
Figure 5.19: QoS of workload with balanced profit for Batch and Service for all algorithms utilizing power profile 3.



(a) Total Profit

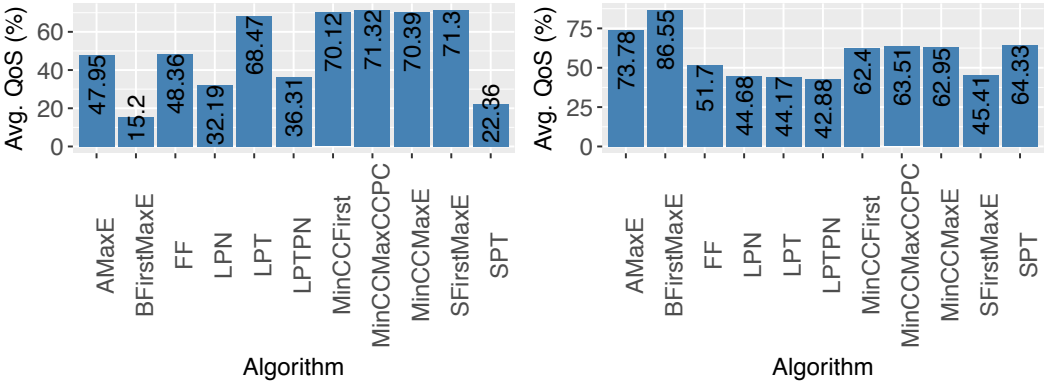


(b) Profit from Service



(c) Profit from Batch

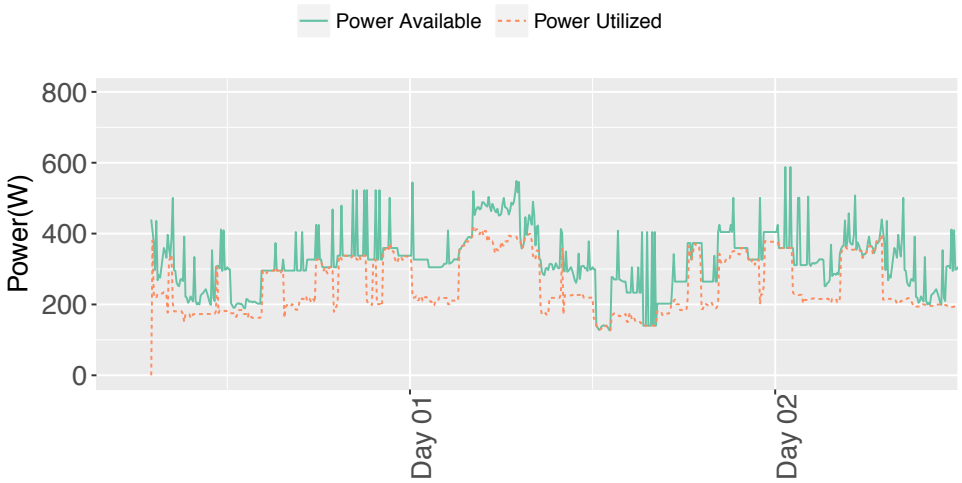
Figure 5.20: Profit of workload with balanced profit for Batch and Service for all algorithms utilizing power profile 4.



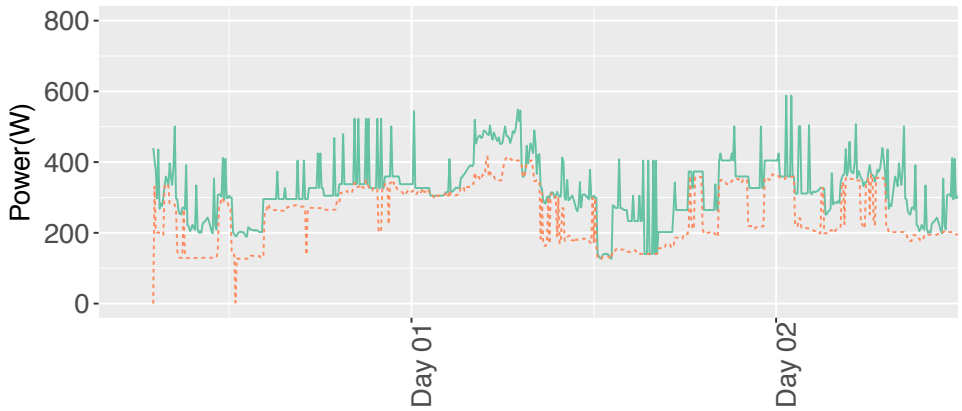
(a) QoS Service

(b) QoS Batch

Figure 5.21: QoS of workload with balanced profit for Batch and Service for all algorithms utilizing power profile 4.

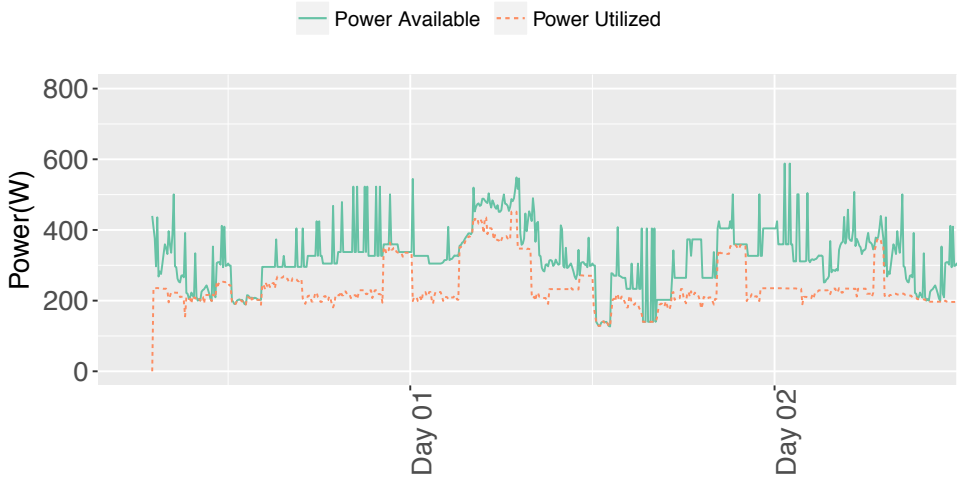


(a) FF

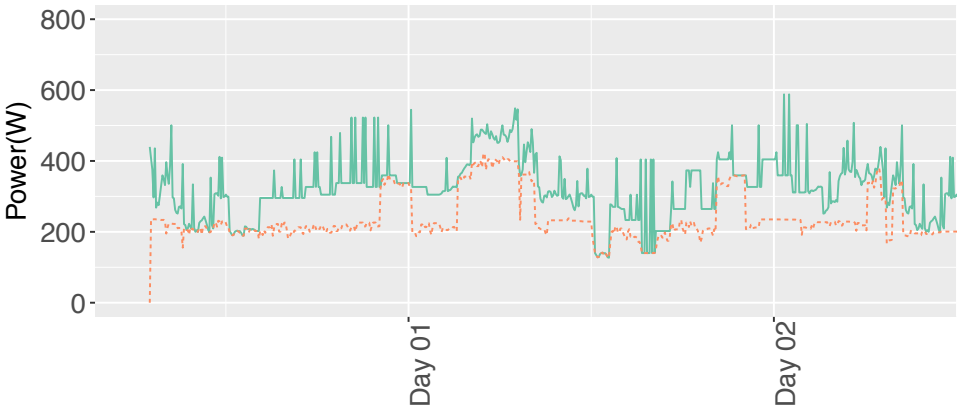


(b) BFirstMaxE

Figure 5.22: Power constraint and power consumption of workload with balanced profit for Batch and Service for FF and BFirstMaxE.

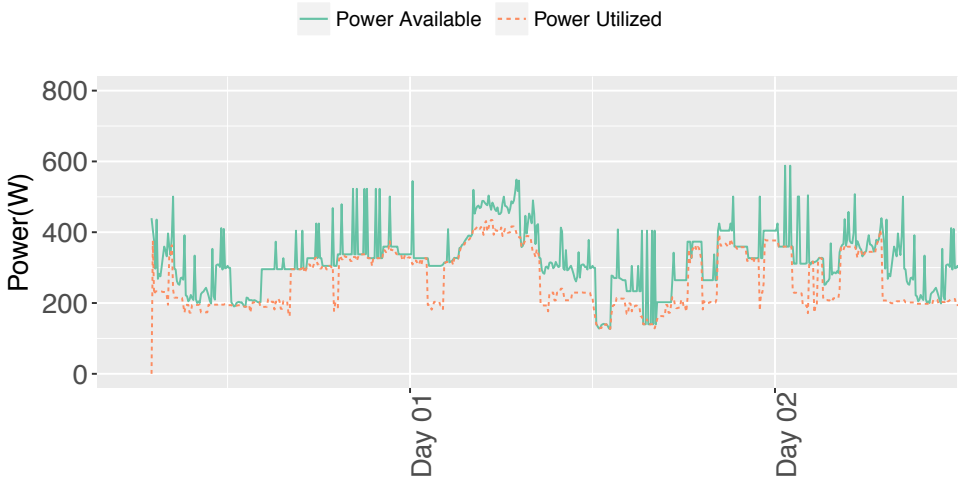


(a) MinCCFirst

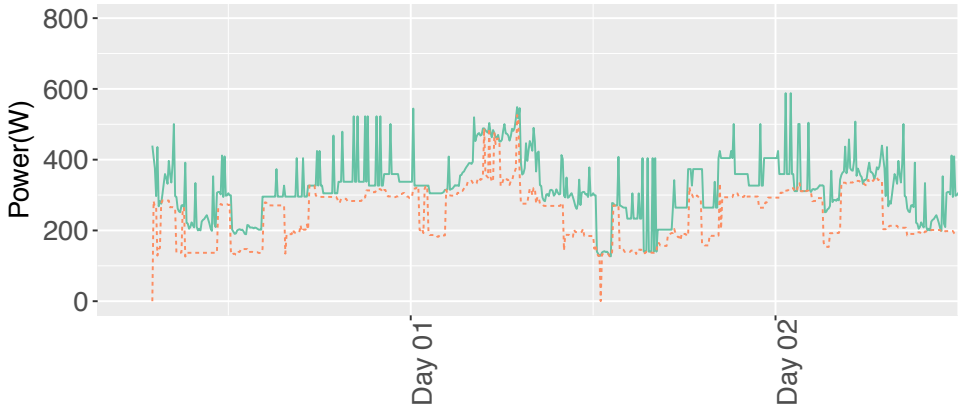


(b) MinCCMaxCCPC

Figure 5.23: Power constraint and power consumption of workload with balanced profit for Batch and Service for MinCCFirst and MinCCMaxCCPC.

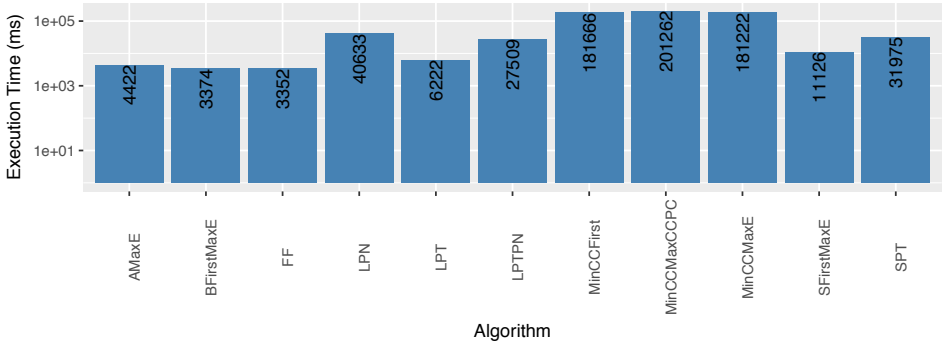


(a) LPTPN

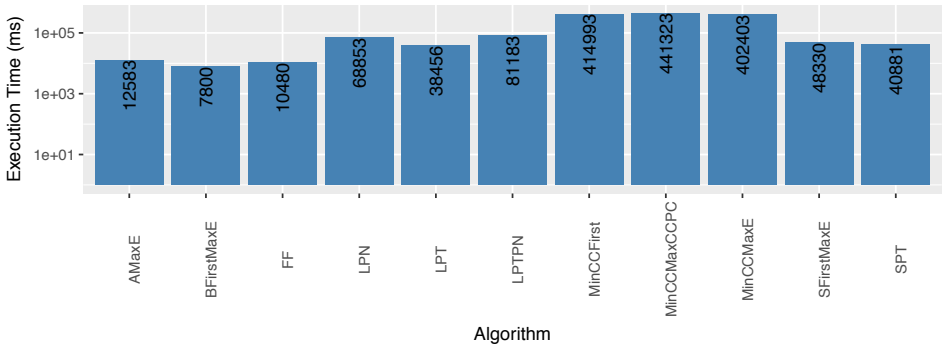


(b) SPT

Figure 5.24: Power constraint and power consumption of workload with balanced profit for Batch and Service for LTPPN and SPT.



(a) Profile 1



(b) Profile 4

Figure 5.25: Execution time of the different algorithms evaluated.

5.6.2 Multiple Workloads With Unbalanced Profit

In this scenario we present the results obtained with the previously presented algorithms but with a different workload. First we are going to introduce the results obtained with all the workload variations up to 211 batch tasks and 55 services. These batches come from the workload generator presented with a lower duration/resources request. According to the profit calculation they will have a lower impact in the overall profit. We then present an overall view of the QoS difference between the cross-correlation algorithms and two other variations that prioritize one type of the task over the other. Later we evaluate the QoS, profit and execution time of some algorithms over the studied power profiles in more details. We then present a closer look on one of the power profiles production and consumption highlighting the point where some of the tasks are violated (even after degradation), and finally we present the loss in profit that could occur if we choose the wrong algorithm for a given workload, in comparison of always using the cross-correlation variation.

In Figures 5.26, 5.27, 5.28 and 5.29 we present one heat map for each power profile where the horizontal axis represents the number of services and the vertical axis the number of batch tasks of the experiment executed. The different colors on the heat map show what are the algorithms (in some cases more than one) that obtained the best profit. We also added a box (black box around some of the results) that represents only an algorithm that uses cross-correlation has the best profit.

What we can see in the aforementioned figures is that cross-correlation algorithms perform as good as the algorithms from the literature that has the best results. In some cases (black boxes), cross-correlation is able to produce a result that is better than all the other algorithms in almost all the workload variations. We also highlight that as we constrained more the power profile, the cases where only cross-correlation variations could obtain the best profit increase. Also, since the impact of batch profit is lower in this scenario we can see that SFirstMaxE can be among the algorithms that have the best profit in several cases.

If we want to study in a closer way what are the reasons why cross-correlation approaches can have a better profit, we need to look at the QoS, which is related to profit penalization. In Figure 5.30 we present the QoS of batch and service separately for all workloads. In the x axis we show the number of batches and each subplot contains a fixed number of services. The power profile 4 is utilized, considering all the algorithm variations. We can see that cross-correlation has a QoS as good as the algorithms that prioritize a given type of task (batch first or service first) in the scheduling process even if it does not make a specific distinction between them, and also better QoS than

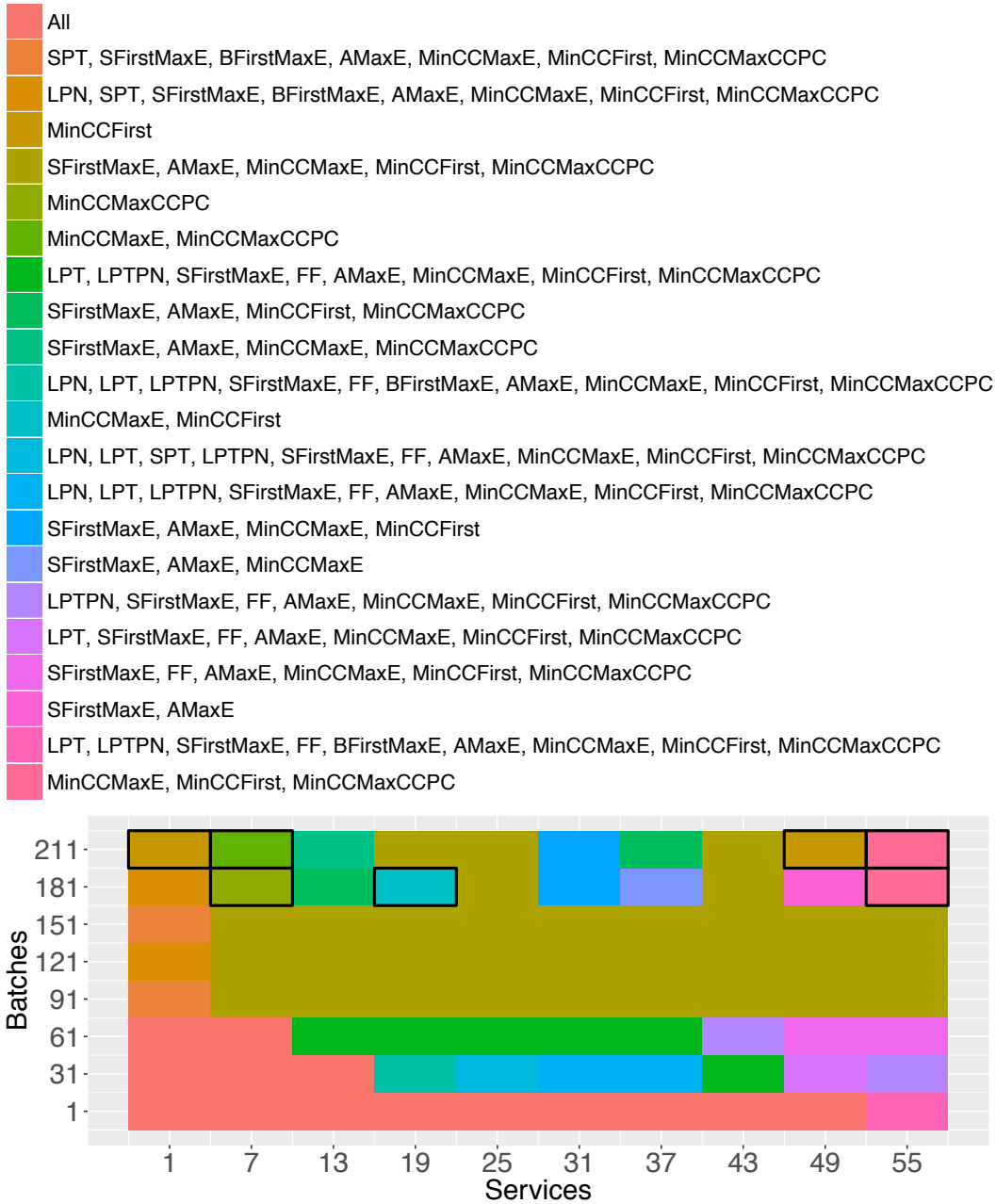


Figure 5.26: Highest total profit comparison with cross-correlation algorithm with power profile 1 for all workload variations.

FF which also does not make distinction between batch and services.

In order to make it easier to visualize we separate in Figure 5.31 four algorithm variations, highlighting with black rectangles the points where the

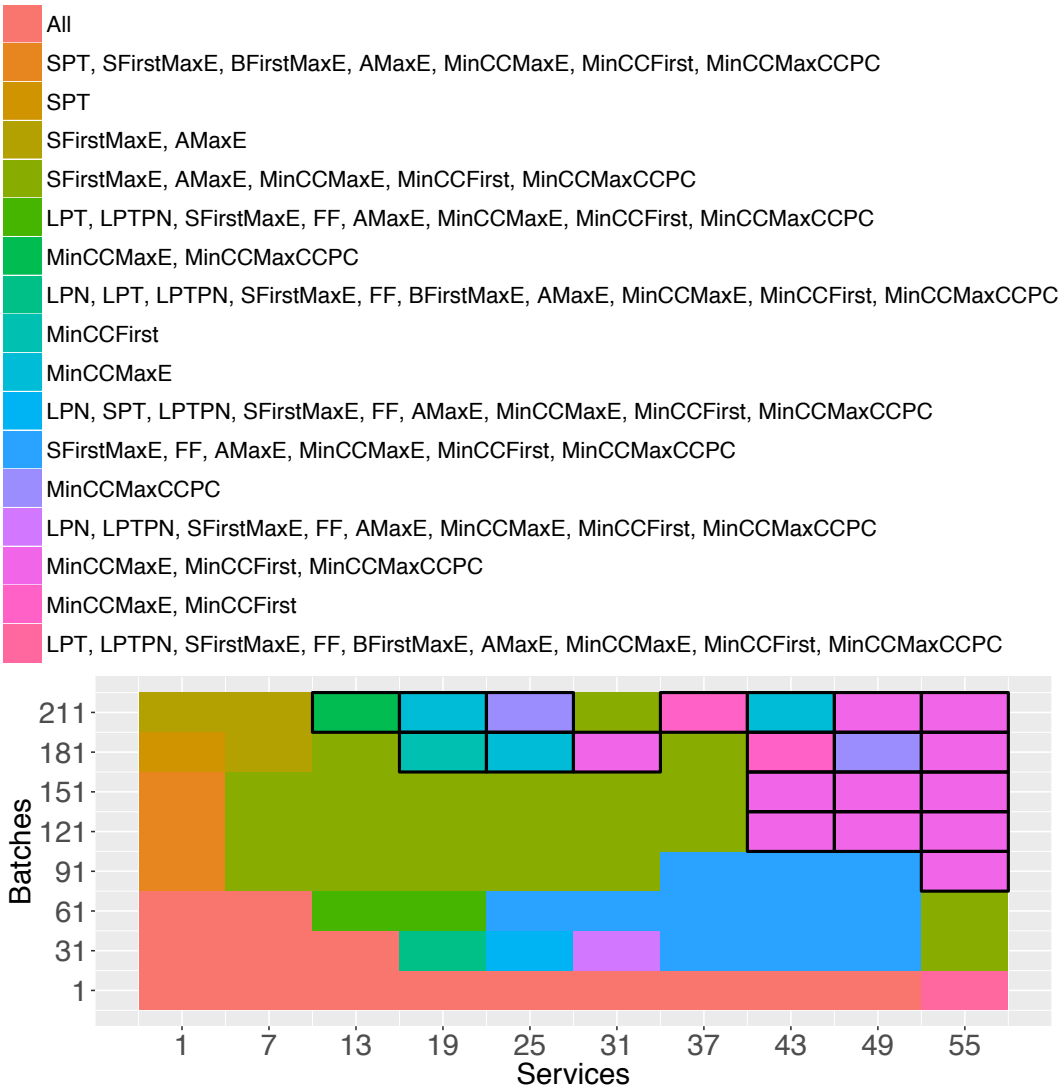


Figure 5.27: Highest total profit comparison with cross-correlation algorithm with power profile 2 for all workload variations.

number of batch increases and the algorithms start to have a drop in the total QoS for batch. In cases where the data center administrator does not know what will be the most profitable kind of task, or does not have a deeper knowledge on what are the proportions of one task over another, cross-correlation algorithms can provide a good QoS for both types of tasks in most of the cases, as seen in the previous balanced workload. We can observe that "MinCCMaxE" has a QoS as good as the "SFirstMaxE" algorithm for services, and as good as "BFirstMaxE" for batch tasks, which also reflects directly in the profit.

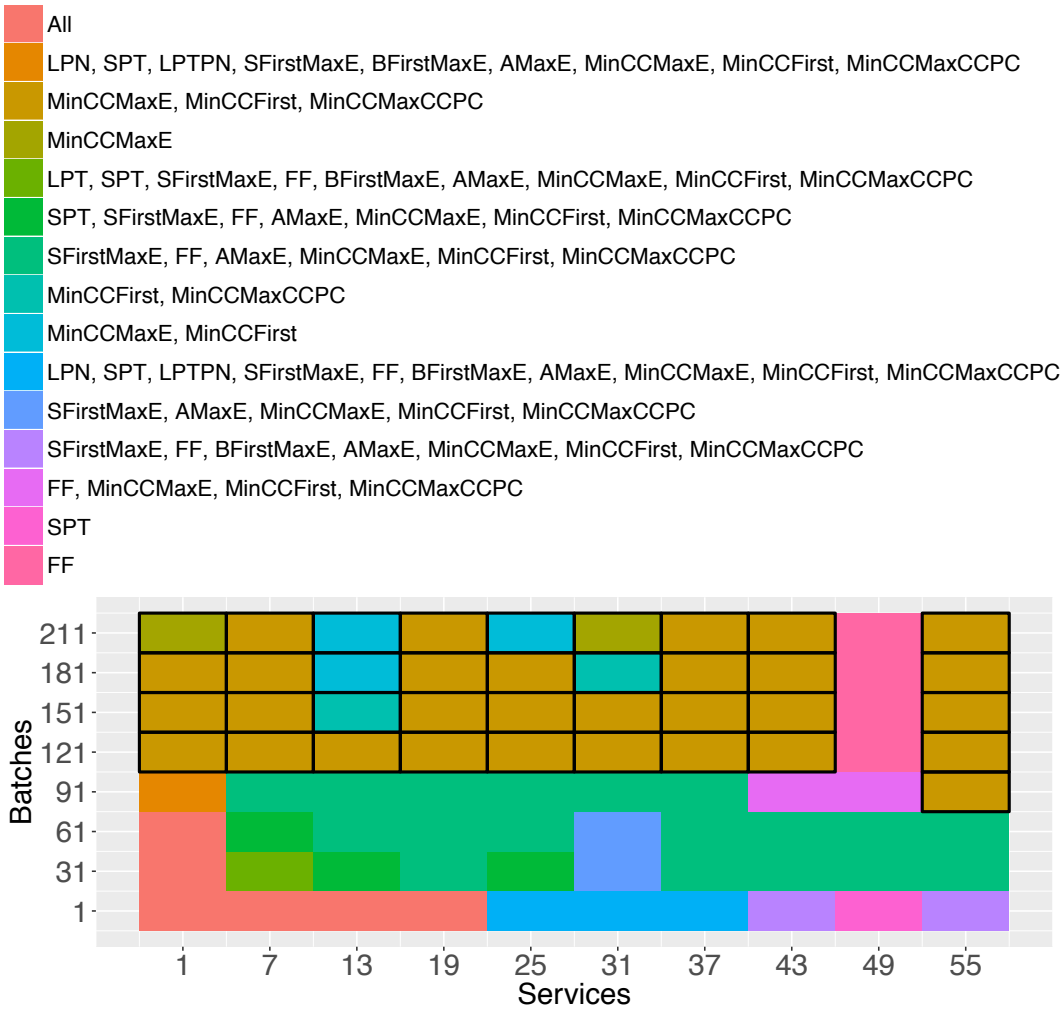


Figure 5.28: Highest total profit comparison with cross-correlation algorithm with power profile 3 for all workload variations.

To have a better view of the different metrics over the proposed power profiles we present in Figure 5.32 one radar plot for each power profile. In this figure we can have a more detailed look in the performance of 5 algorithms from the literature against one of the cross-correlation propositions under the maximum workload (211 batches, 55 services). In the first power profile, even under a more relaxed power constraint LPT, LPTPN and BFirstMaxE already present QoS degradations and a higher impact in profit, and in a more attenuated way First Fit (FF) also has some degradations in the profit. As we go to a more constrained power profile SFirstMaxE can keep a similar profit, but not without reducing the QoS for batch tasks (which reflects in

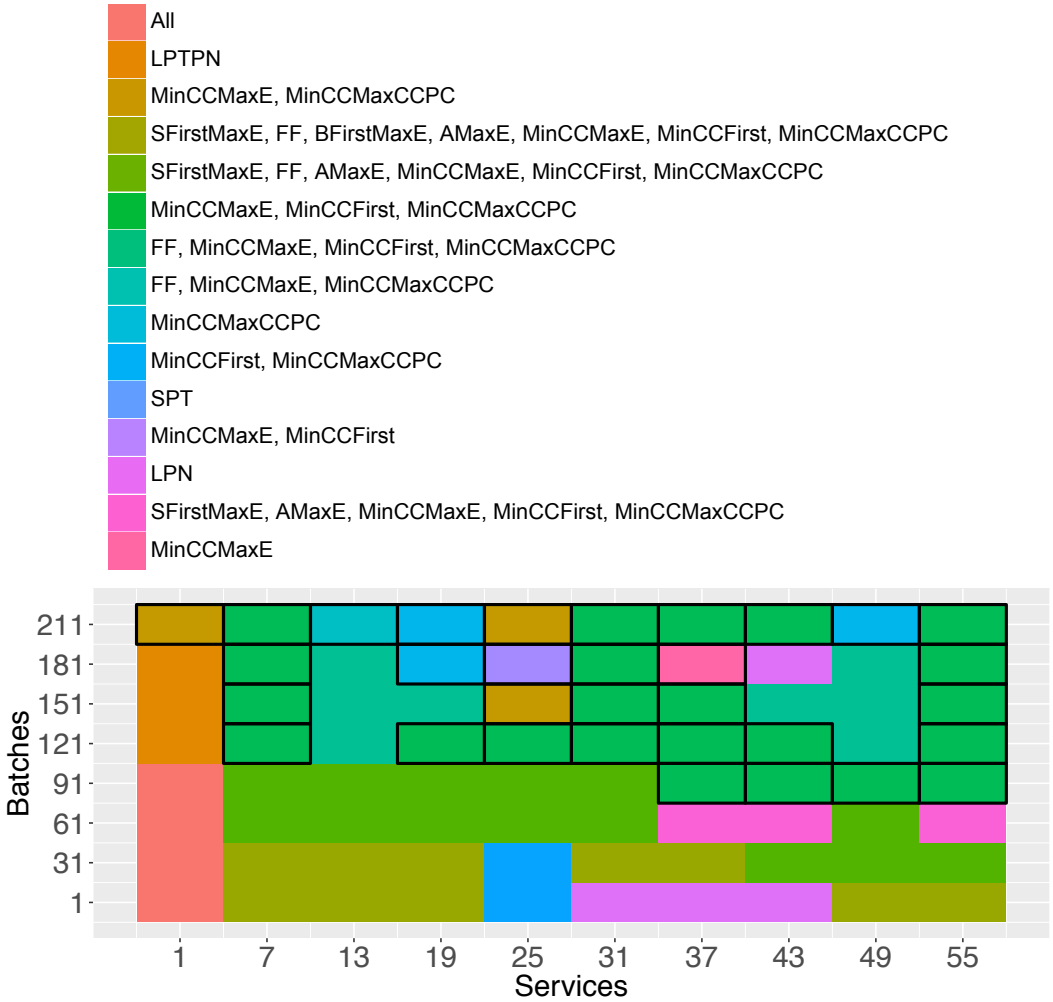


Figure 5.29: Highest total profit comparison with cross-correlation algorithm with power profile 4 for all workload variations.

the reduction of profit). We can see that with exception of MinCCMaxE all the other algorithms oscillate either in QoS for Service or QoS for batch tasks, which reflects directly in the user satisfaction and in the profit that the data center could have (even when the power is constrained like in profile 4). The main drawback again is the execution time of the algorithm as seen in previous experiments.

Regarding the factors that lead to different performance of the algorithms we present in Figure 5.33 the power produced and consumed for one execution of "MinCCMaxCCPC" with some "choke points" (points where we have an abrupt drop in the power production) highlighted with circles where there is a

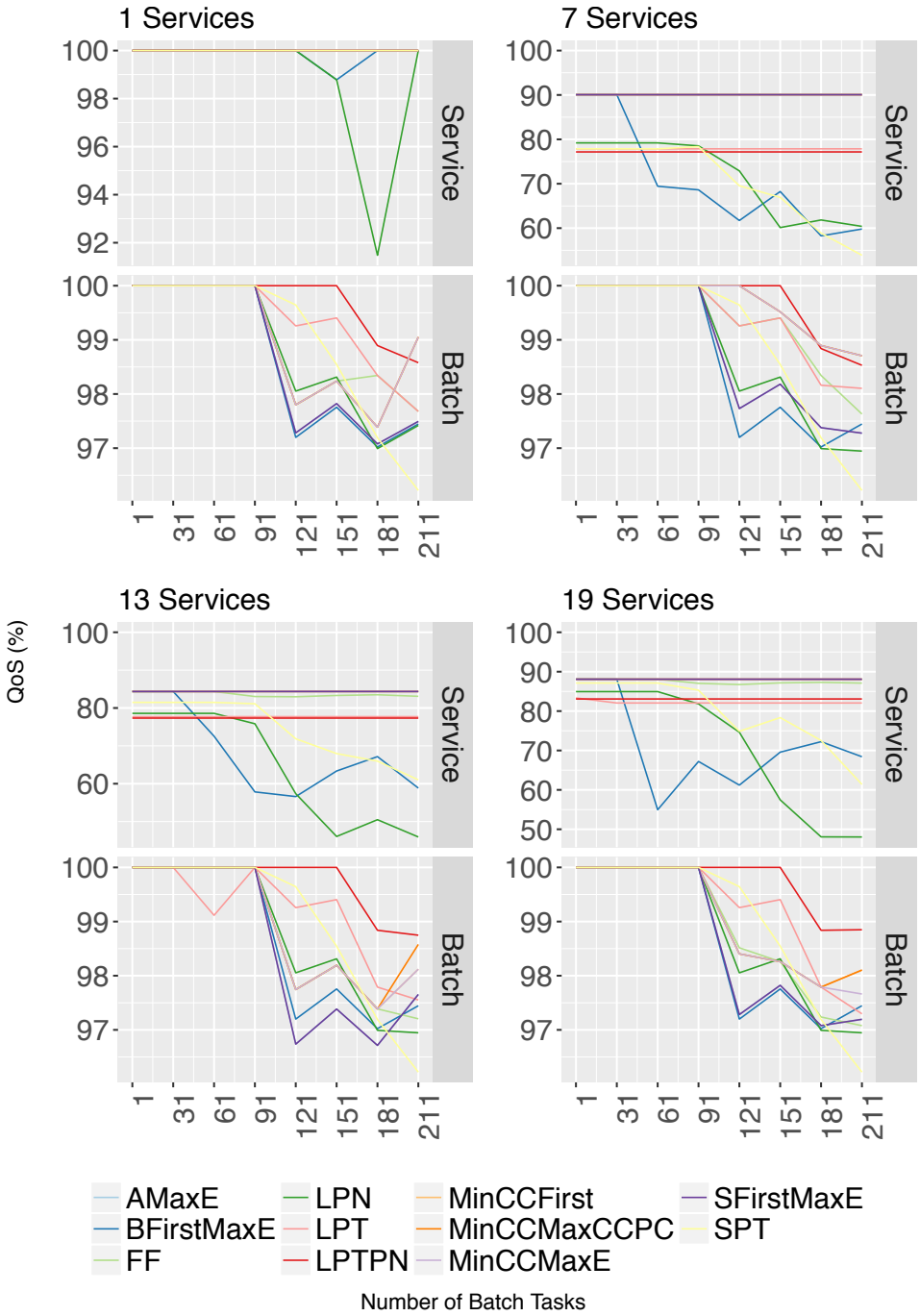


Figure 5.30: QoS distance of different algorithms for both batch and services over all workload variations with power profile 4.

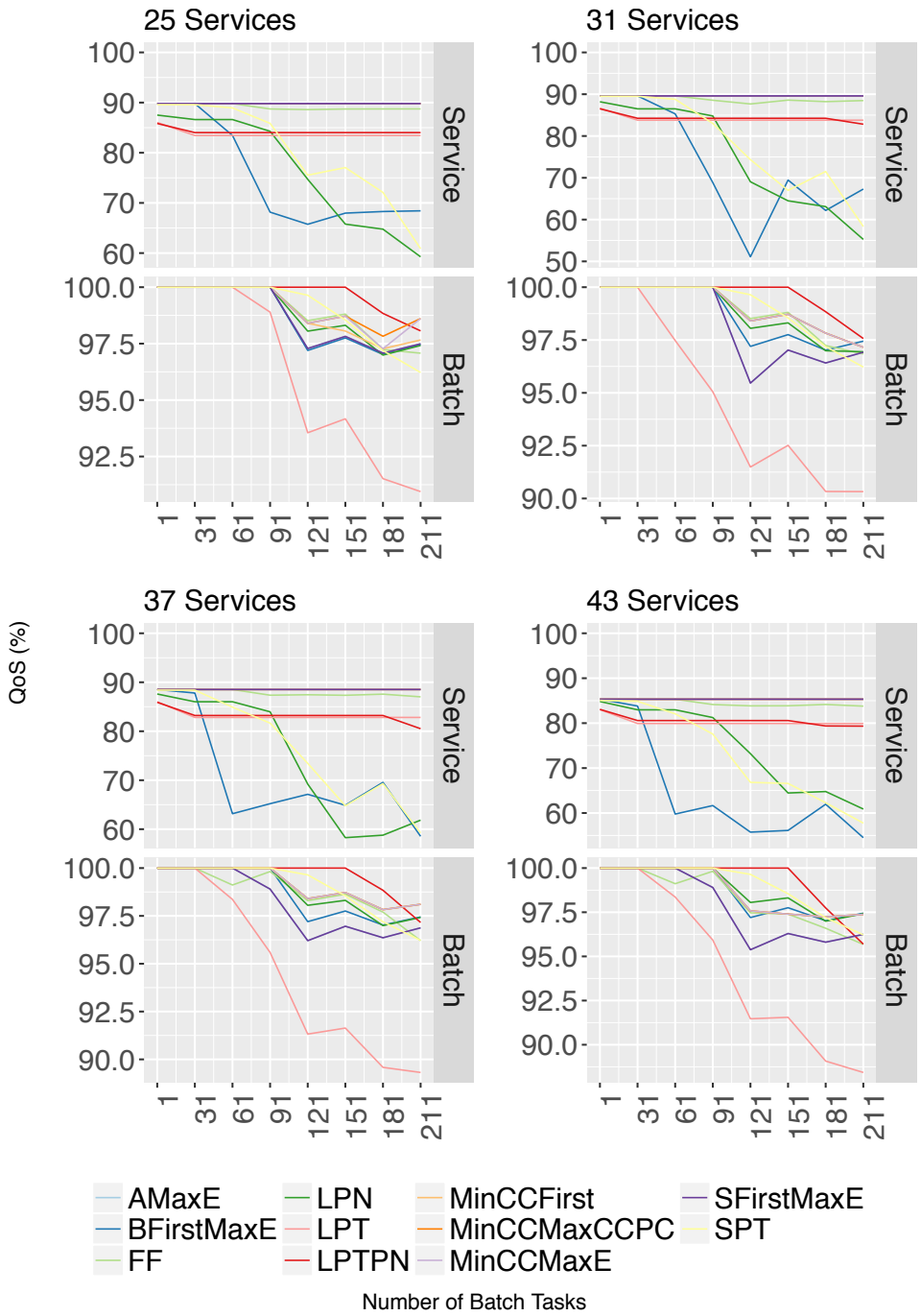


Figure 5.30: QoS distance of different algorithms for both batch and services over all workload variations with power profile 4.

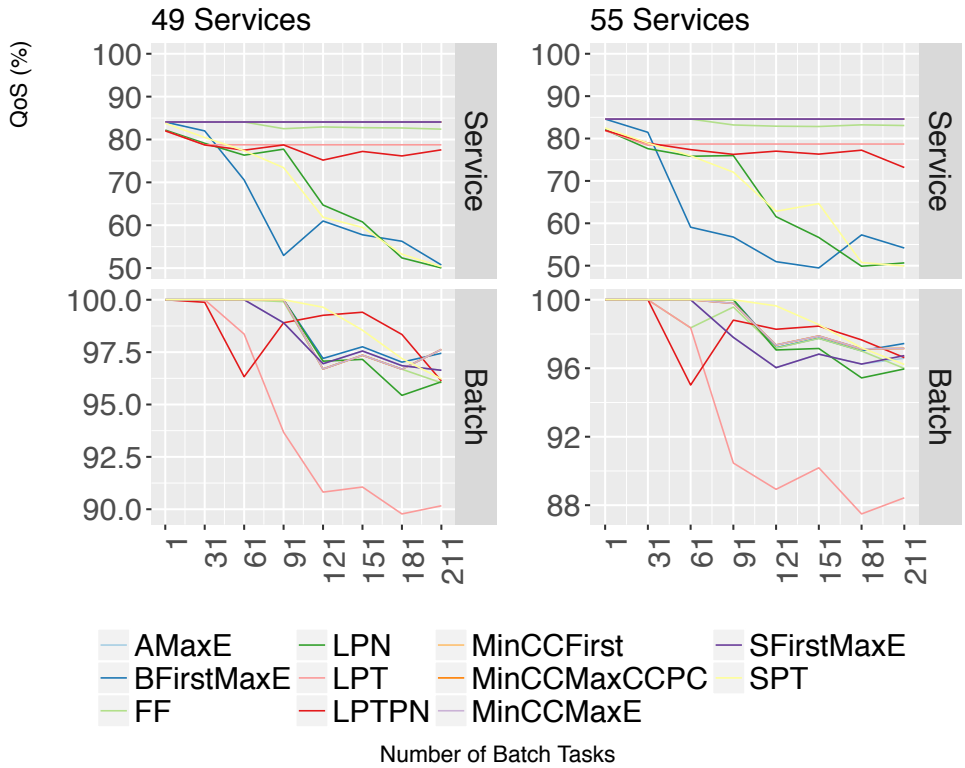


Figure 5.30: QoS distance of different algorithms for both batch and services over all workload variations with power profile 4.

forced reduction in the resources available (nodes/processors). These reductions let impossible the scheduling of the tasks presented underneath (colored lines in the base of Y axis). What these lines represent is the ID of the task, and length refers to its release and due date. The tasks coincide with points where there is a drop in the power available, and since these tasks specifically do not have high flexibility, the tasks finish after the expected due date.

Finally, in Figures 5.34, 5.35, 5.36 and 5.37 we present the impact that a poorly chosen algorithm could have in the profit, when compared to using cross-correlation (MinCCMaxE) for each power profile. These heat maps are interpreted in the same way as the previous ones, but in this case each color represents one or more algorithms that had the worst profit. We present inside each combination the profit percentage difference that could be brought when compared to MinCCMaxE. In general as seen in Figure 5.29, SFirstMaxE is good because it appears many times in the heat map. But, in Figure 5.37, we can see that compared to cross-correlation, the profit obtained with SFirstMaxE is the worst profit among all algorithms. In this case where there is 1 service, between 121 and 181 batches. This chart highlights the losses that a data

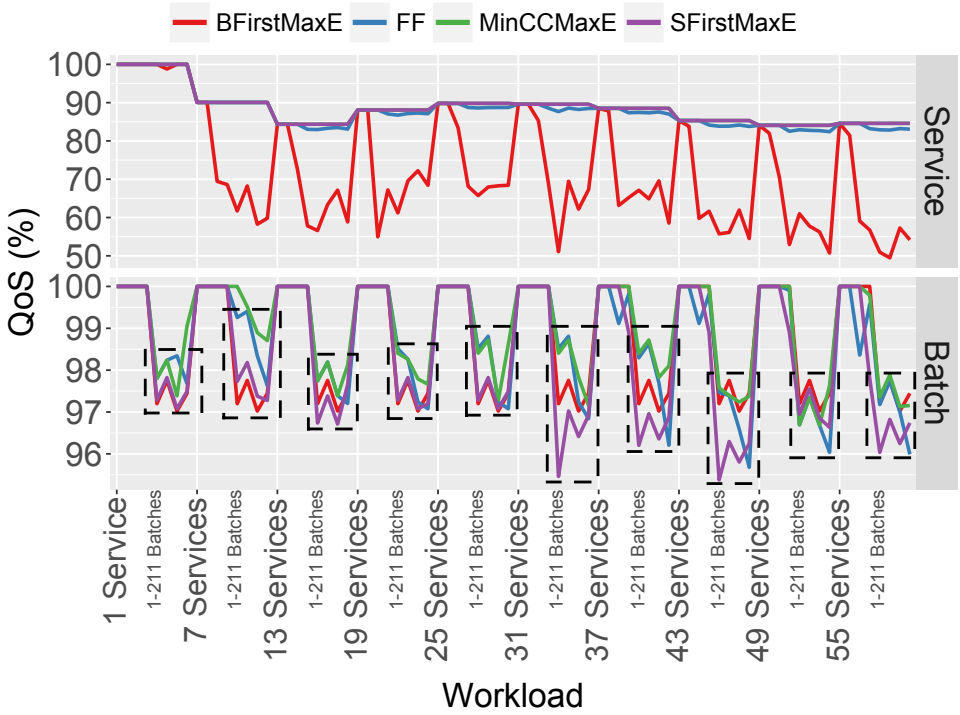
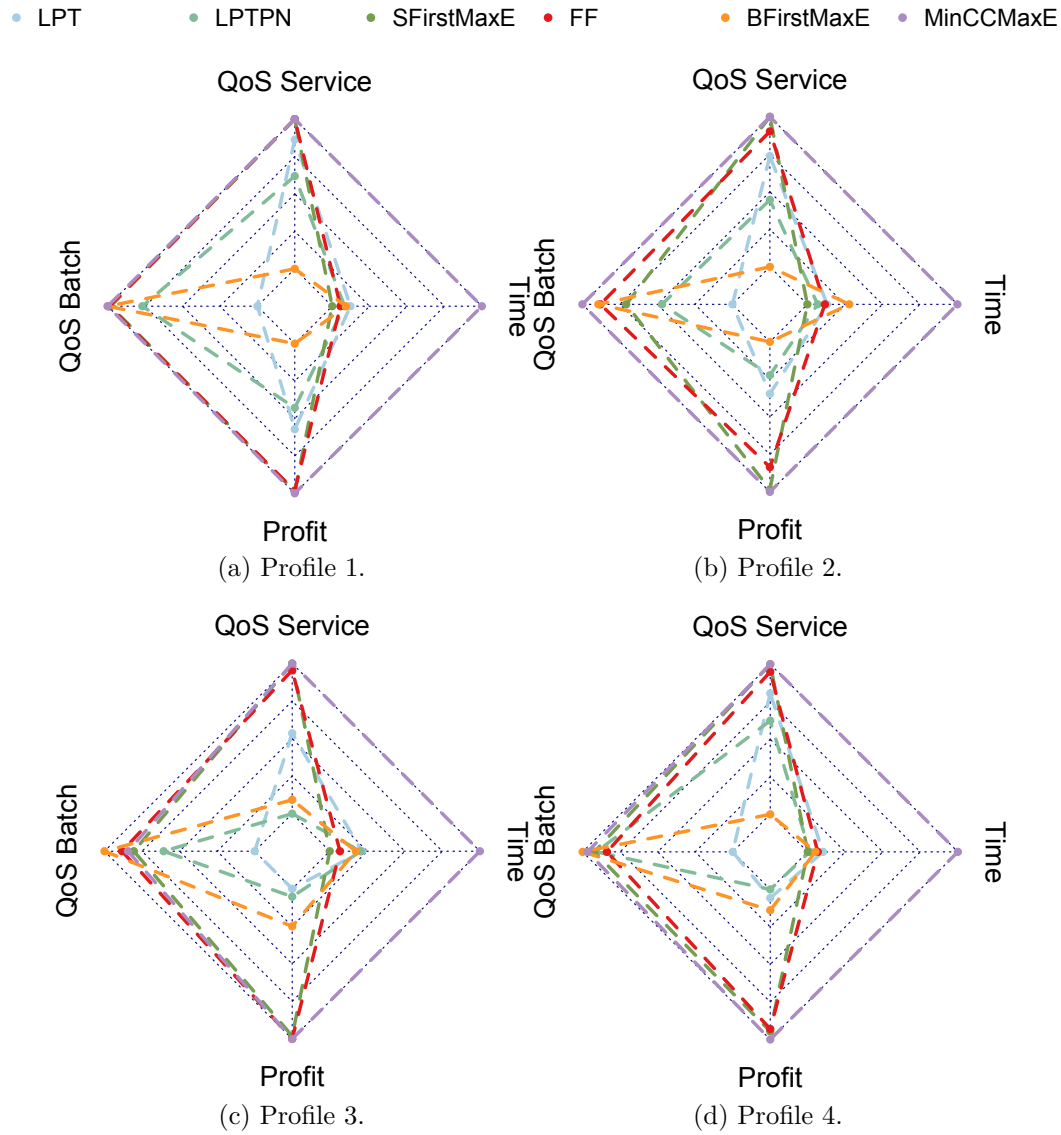


Figure 5.31: QoS distance of different algorithms for both batch and services over all workload variations with power profile 4.

center manager could have if the workload is unknown and the algorithm is poorly chosen. The increase in profit can be up to 44% for profiles 1 and 2, 43% for profile 3 and 34% for profile 4 for instance when utilizing MinCCMaxE. MinCCMaxE provides good results whatever the composition of the workload and the power constraint. One must notice that the profit might also reflect the QoS degradation that occurs in the user side.



(a) Profile 1.

(b) Profile 2.

(c) Profile 3.

(d) Profile 4.

Figure 5.32: QoS for batch and services, profit and execution time of all power profiles and the maximum workload (55 services and 211 batch tasks), with worst in center and best at exterior.

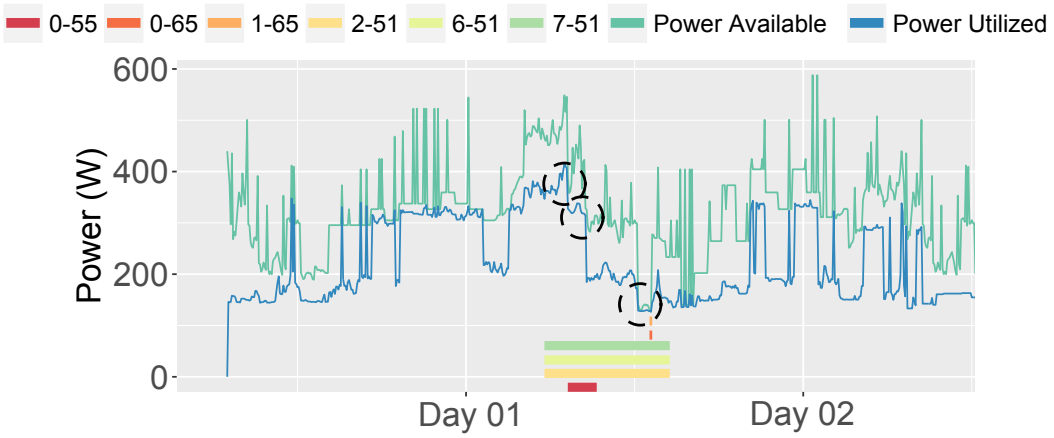


Figure 5.33: Power production and consumption with power profile 4 and the point where tasks where violated using MinCCMaxCCPC.

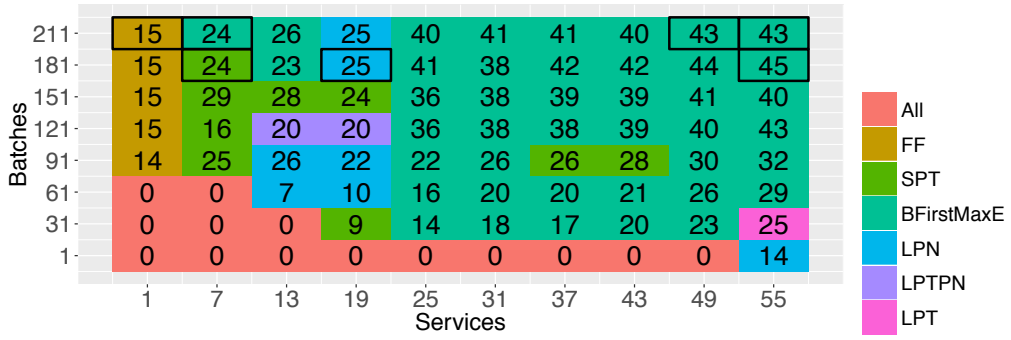


Figure 5.34: Percentage of profit distance between MinCCMaxE algorithm and the worst performing of power profile 1 for all workload variations.

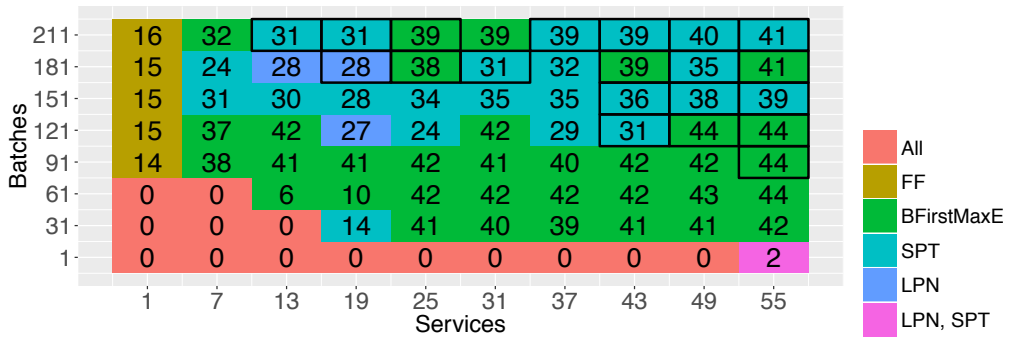


Figure 5.35: Percentage of profit distance between MinCCMaxE algorithm and the worst performing of power profile 2 for all workload variations.

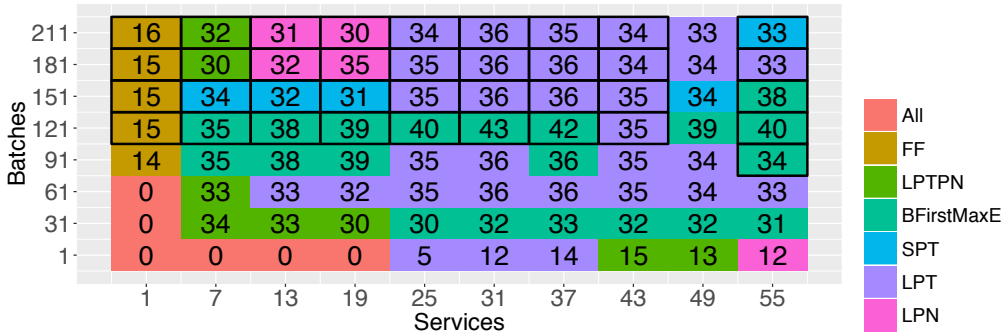


Figure 5.36: Percentage of profit distance between MinCCMaxE algorithm and the worst performing of power profile 3 for all workload variations.

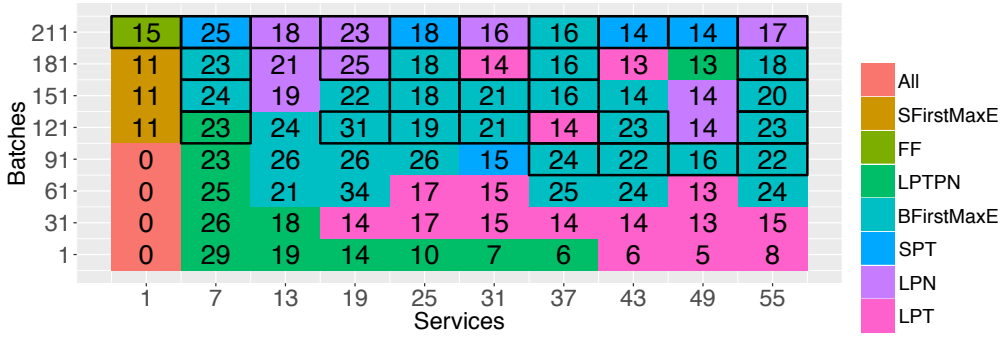


Figure 5.37: Percentage of profit distance between MinCCMaxE algorithm and the worst performing of power profile 4 for all workload variations.

5.6.3 Unbalanced Workload Variation and Degradation Impact Evaluation

In this scenario we present the results obtained with the previously presented algorithms but with a different workload. Again, we chose the same number of services as in the previous experiment, but with different resources consumption and the same number of batch tasks but higher memory consumption (more profitable). These experiments aim to evaluate if the distance between the profits of SFirstMaxE and cross-correlation algorithms increases, according to the previous statements, and how the algorithms behave with a different workload. Also, we aim to evaluate what is the impact of the degradation among some algorithms, which type of tasks are impacted, and how it affects the profit. We evaluate in case the datacenter administrator opts to not utilize it and just reject the tasks that cannot meet the resources requested.

We start by evaluating the total profit for each workload variation over the 4 different power profiles. These results are displayed in Figures 5.38, 5.39, 5.40 and 5.41. In these figures we highlight several points where the profit between SFirstMaxE which obtained good results as well in the previous scenario starts to distantiate even further from the profit of the cross-correlation approaches. This occurs as the number of batches increase, and this variation starts to loose profit. We can also see how the other variations react, and that the more constrained the power profile is sooner the other algorithms start to have degradation in profit.

To make it easier to visualize where the differences come from, we separate in Figure 5.42 the profit from each type of task in the cases with the higher number of services (49 and 55) for Profile 3. In this case we can see that both SFirstMaxE and cross-correlation variations have the same profit for Service, but as the number of batches starts to increase, SFirstMaxE is the algorithm with the highest (and increasing) losses in batch profit. We can see that as the proportionality between batch profit and service profit changes, even algorithms such as SFirstMaxE that had good results in previous workload start to have a higher profit loss.

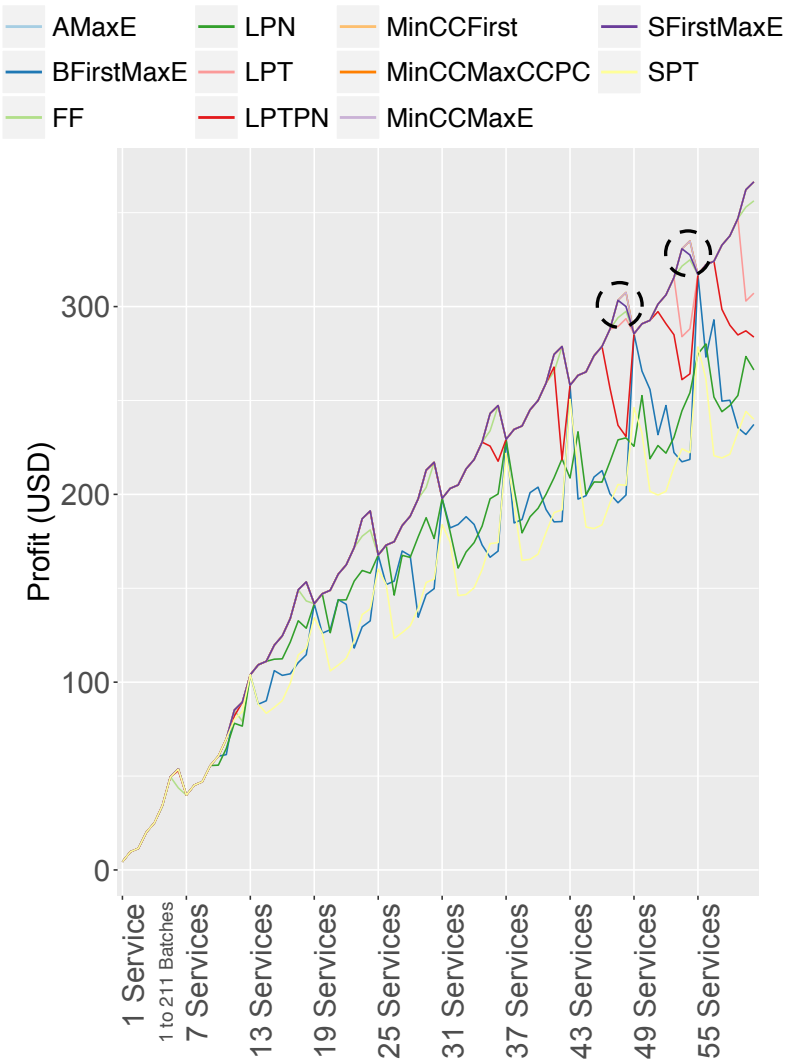


Figure 5.38: Total profit of different algorithms for over all workload variations with power profile 1.

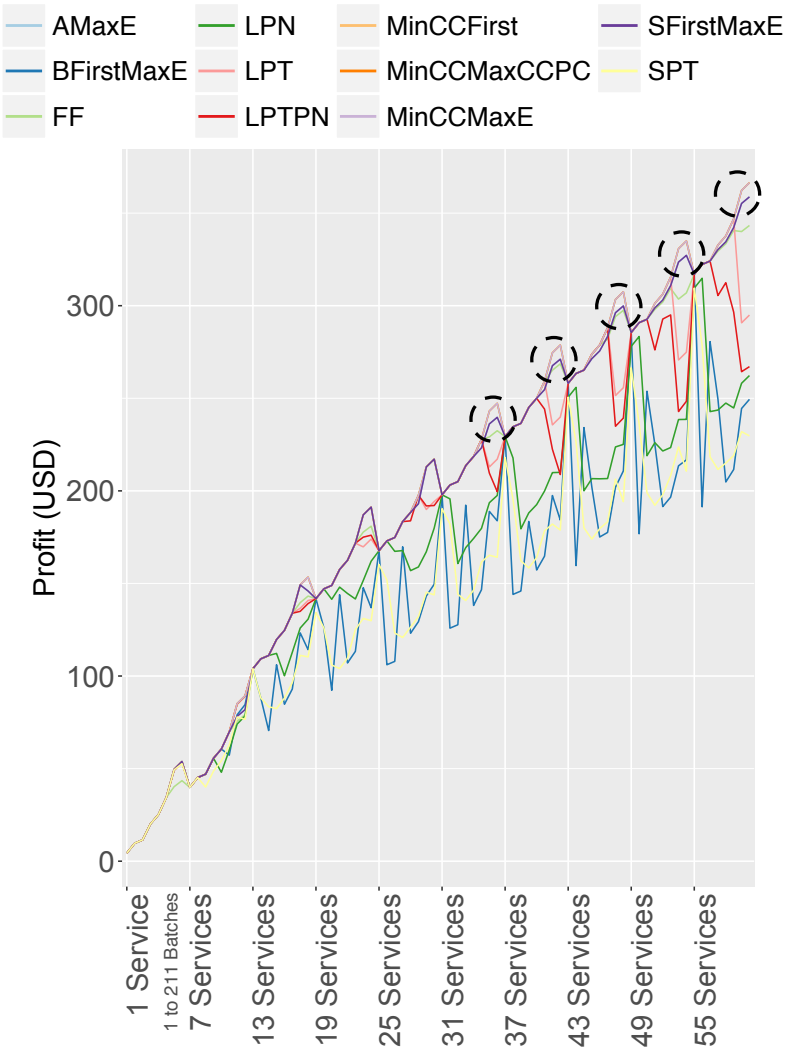


Figure 5.39: Total profit of different algorithms for over all workload variations with power profile 2.

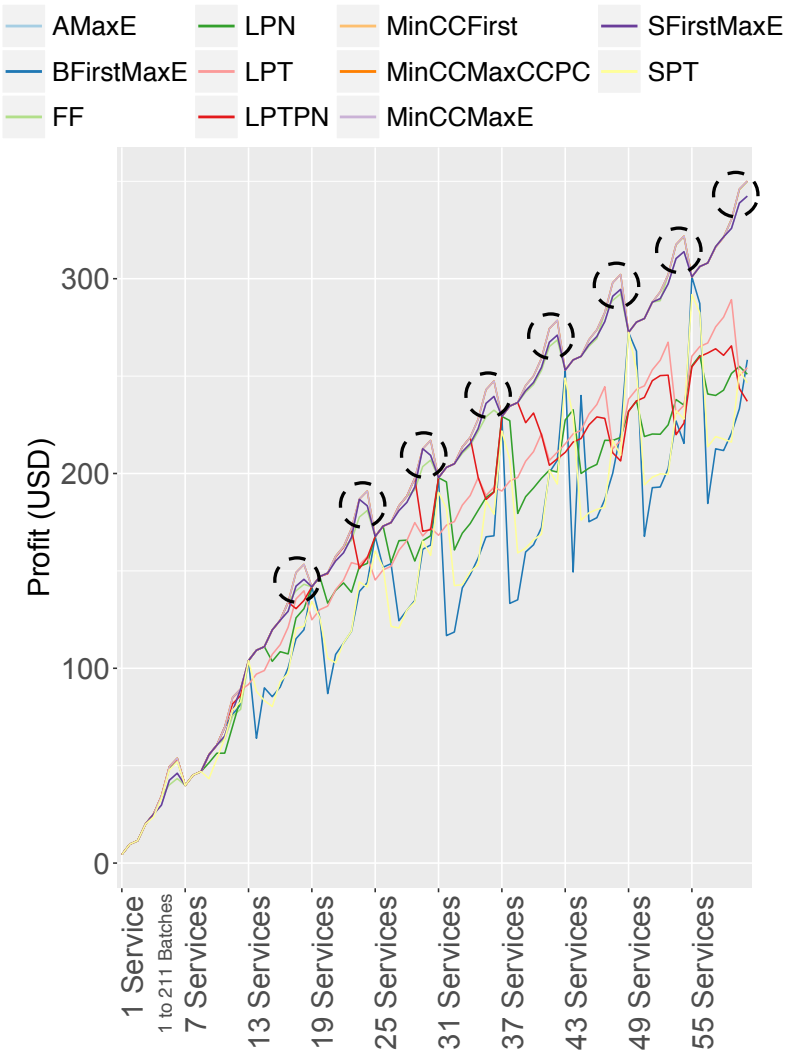


Figure 5.40: Total profit of different algorithms for over all workload variations with power profile 3.

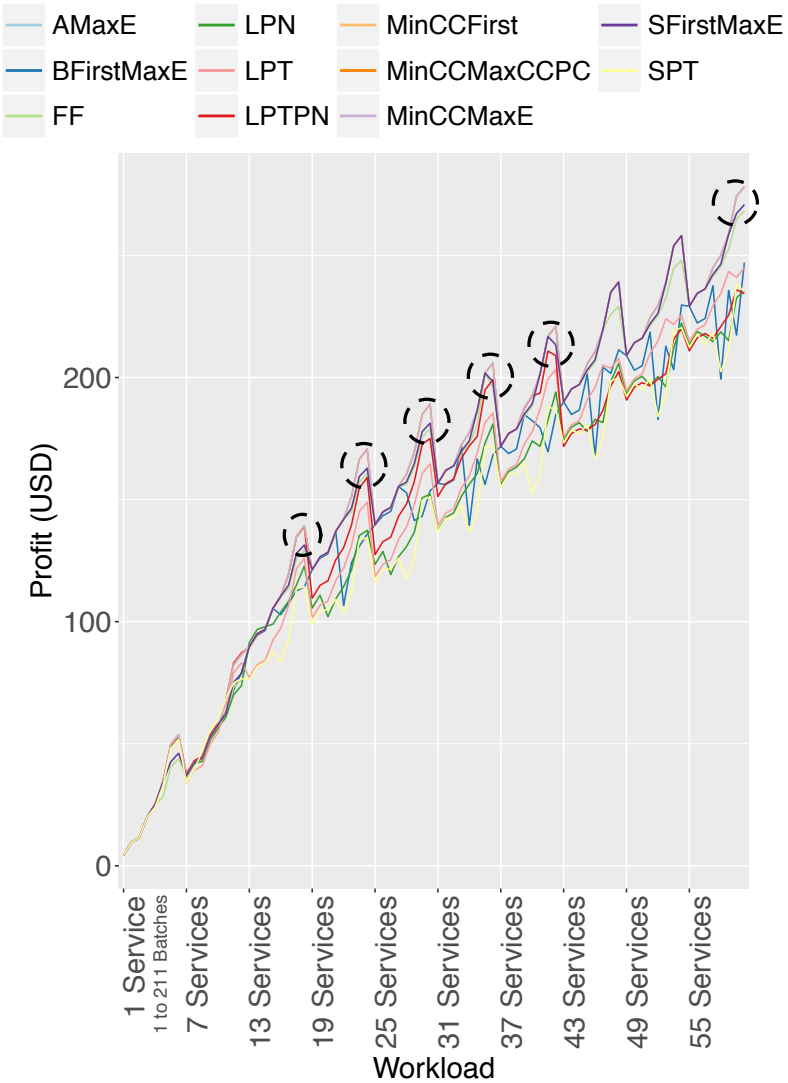


Figure 5.41: Total profit of different algorithms for over all workload variations with power profile 4.

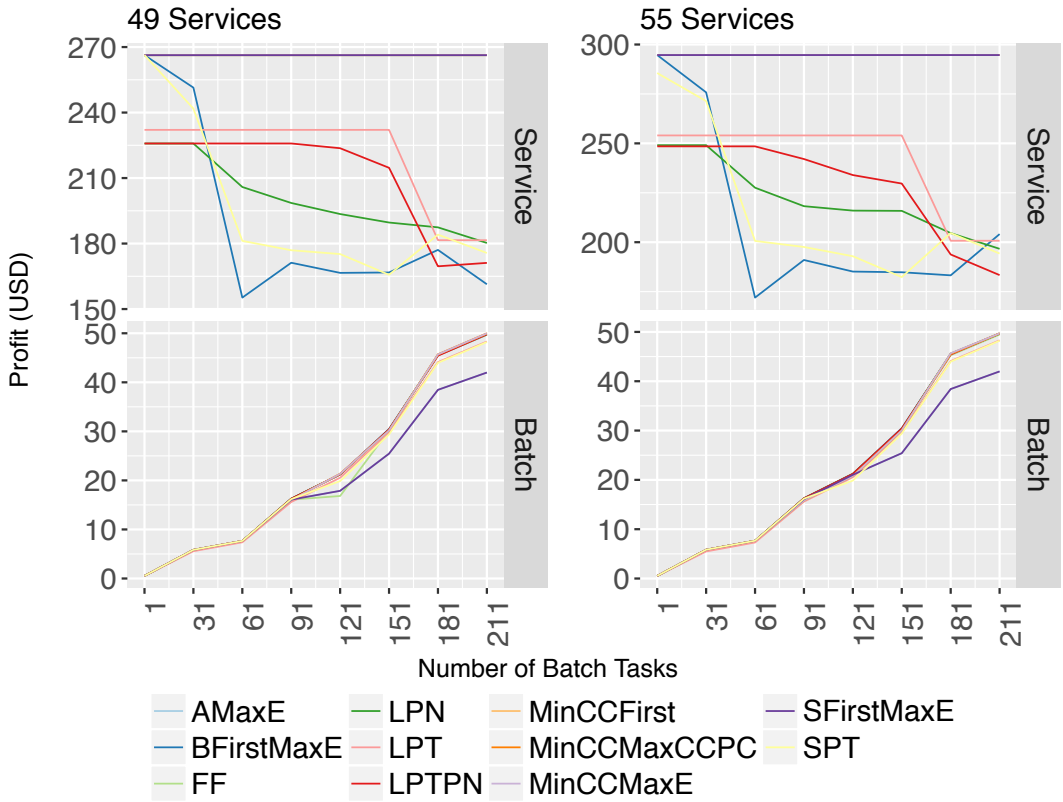


Figure 5.42: Profit distance of different algorithms for both batch and services with power profile 1 for 49 and 55 services workload.

Degradation Removal Impact

Hereafter we show the impact on profit that the removal of degradation would have on some of the algorithms presented. In this case we present the profile with most energy (Profile 1) and the one with the least amount of energy (Profile 4). The workload is the same as in subsection 5.6.3. When removing the degradation the only options would be the removal of services that cannot fit the power/resources constraint, and for batch the change of the execution interval (start/end). No changes in the resources utilization would be possible.

We start by showing the MinCCMaxCCPC profit with/without degradation in Figures 5.43 and 5.44, where the x axis we have the number of services, among which there is all variations of batch tasks (1 to 211). We can see that for Profile 1 there is no impact in not utilizing the degradation. As the power gets more restricted, in Figure 5.44 we see that there would be a significant impact on the profit as we increase the workload. This profit comes from service tasks that would have to be removed, which in the worst case is up to 37 services, representing a loss of 38% of the total profit.

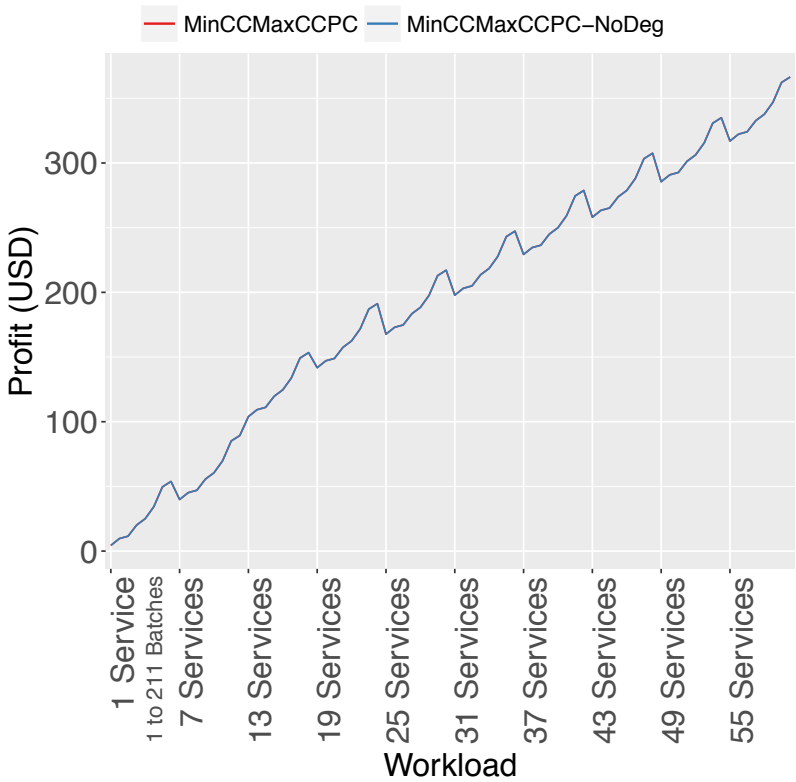


Figure 5.43: Profit impact of degradation removal considering power profile 1 and algorithm MinCCMaxCCPC.

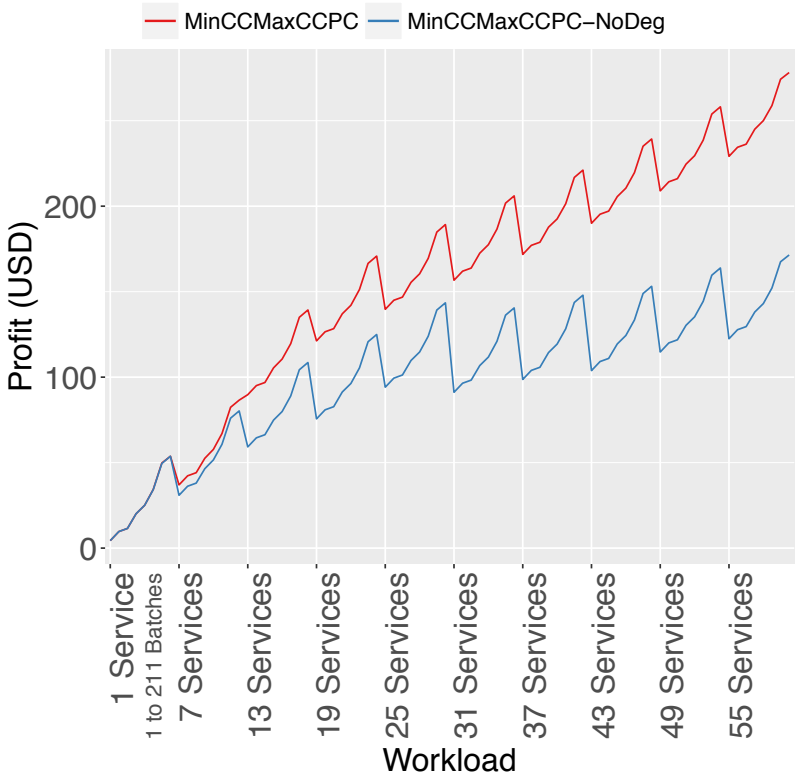


Figure 5.44: Profit impact of degradation removal considering power profile 4 and algorithm MinCCMaxCCPC.

The profit for SFirstMaxE considering with/without degradation is shown in Figures 5.45 and 5.46. We can see that for Profile 1 also there is no impact in not utilizing the degradation. But again, as the power gets more restricted, in Figure 5.46 we see that there would be a significant impact on the profit as we increase the workload, representing 39% of the total profit.

The profit for FF considering with/without degradation is shown in Figures 5.47 and 5.48. We can see that for Profile 1 there is already an impact in not utilizing the degradation when we have 1, 7, 13 and 55 services. And again, as the power gets more restricted, in Figure 5.48 we see that there would be also an impact on the profit, representing 40% of the total profit.

The profit for BFirstMaxE considering both variations with/without degradation is shown in Figures 5.49 and 5.50. In this case, even for Profile 1 there is already an impact in not utilizing the degradation. After placing all batches, there is no possibility in degrading the remained services, similar as shown in the illustration at Figure 5.11. Again, as the power gets more restricted, in Figure 5.50 the impact on the profit represents 44% of the total profit.

Finally, to show the impact in one of the Kassab et al. [67] we evaluate SPT

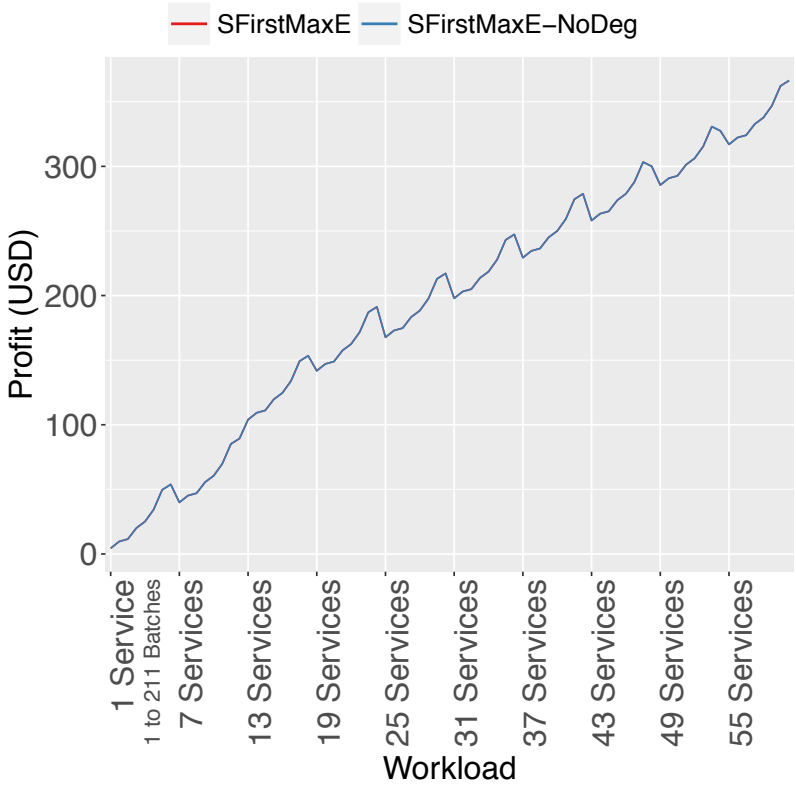


Figure 5.45: Profit impact of degradation removal considering power profile 1 and algorithm SFirstMaxE.

considering with/without degradation in Figures 5.51 and 5.52. In this case, also for Profile 1 there is already an impact in not utilizing the degradation, which increases in Profile 4, representing 41% of the total profit.

Evaluating the difference between the scenario with and without degradation, we can conclude that degradation is necessary to adapt to a power constraint. Also among the evaluated algorithms the cross-correlation variation was the one with the smallest impact of removing degradation.

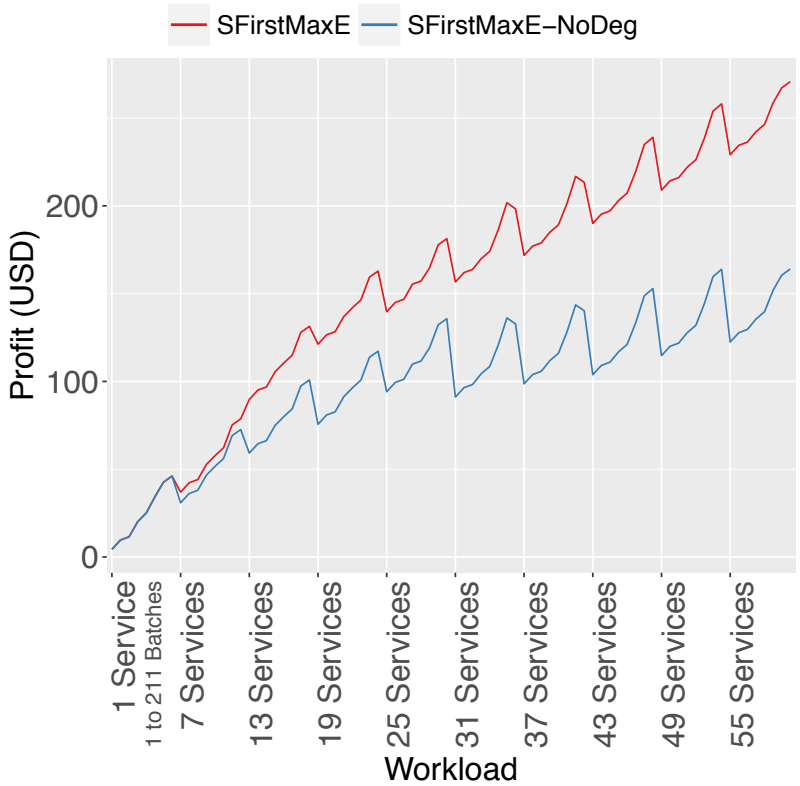


Figure 5.46: Profit impact of degradation removal considering power profile 4 and algorithm SFirstMaxE.

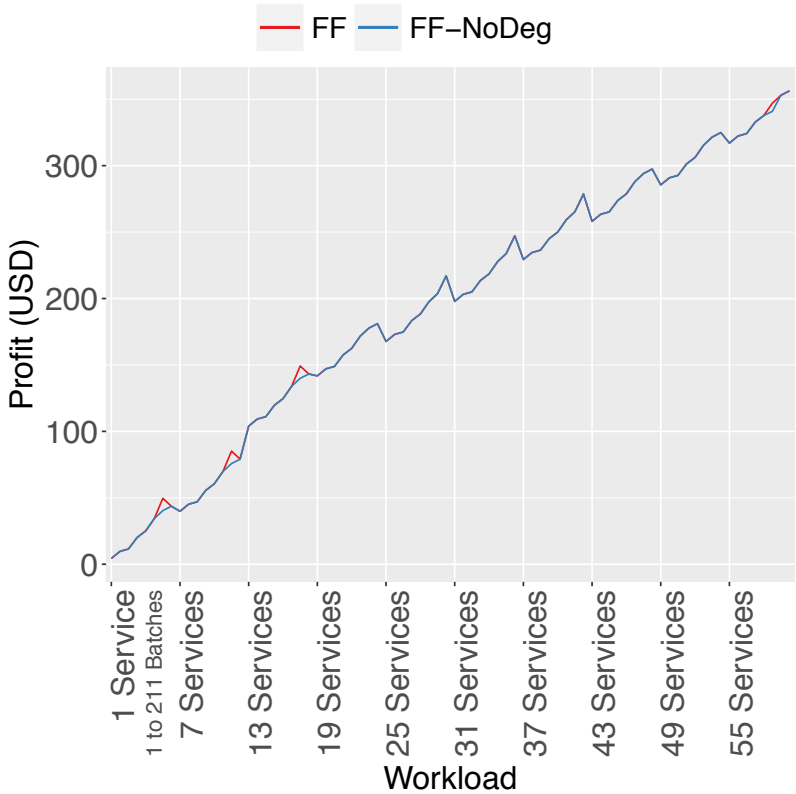


Figure 5.47: Profit impact of degradation removal considering power profile 1 and algorithm FF.

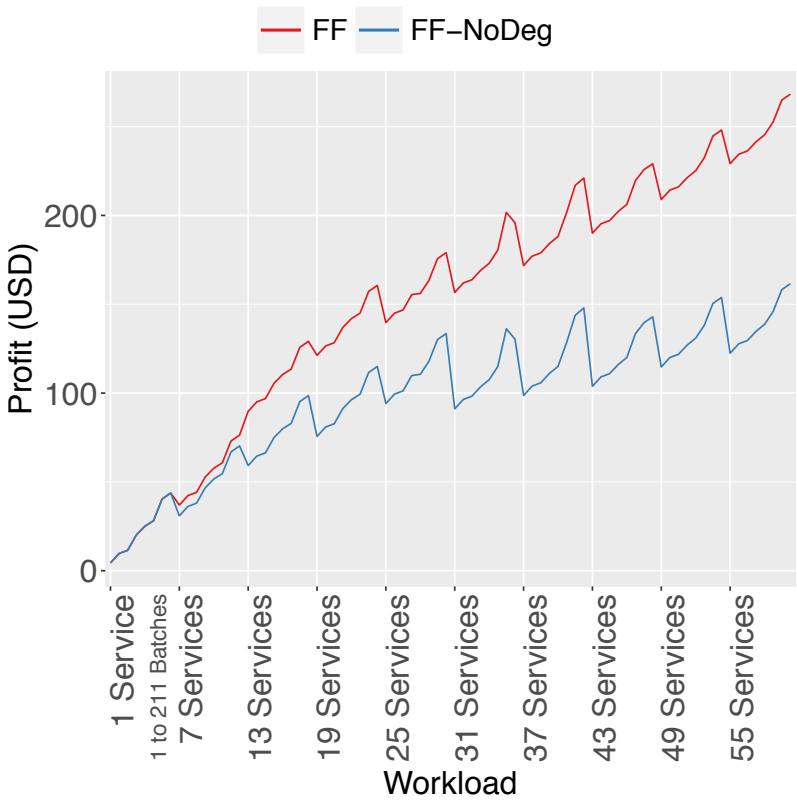


Figure 5.48: Profit impact of degradation removal considering power profile 4 and algorithm FF.

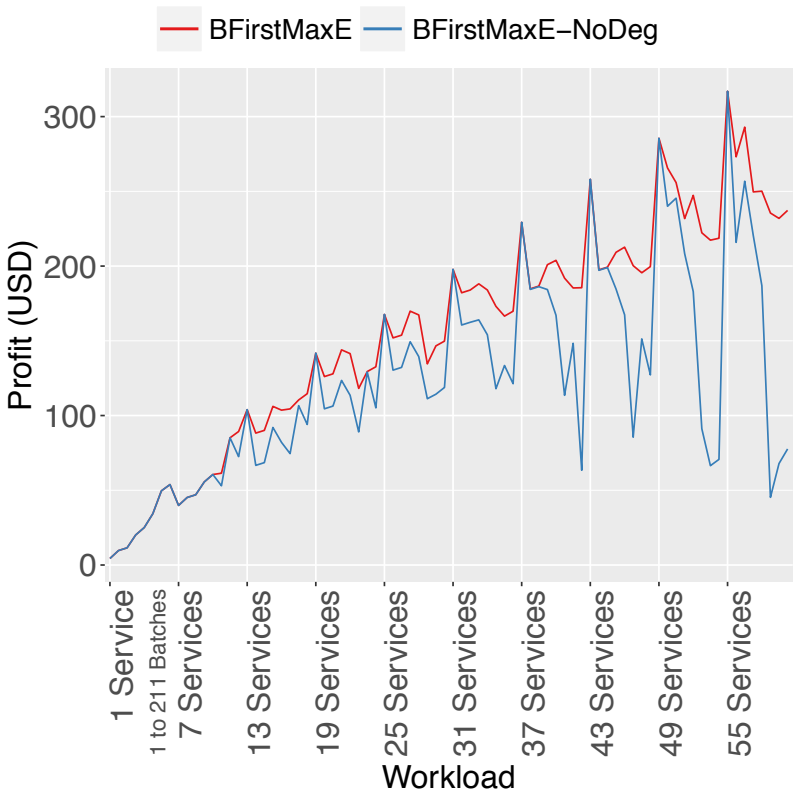


Figure 5.49: Profit impact of degradation removal considering power profile 1 and algorithm BFirstMaxE.

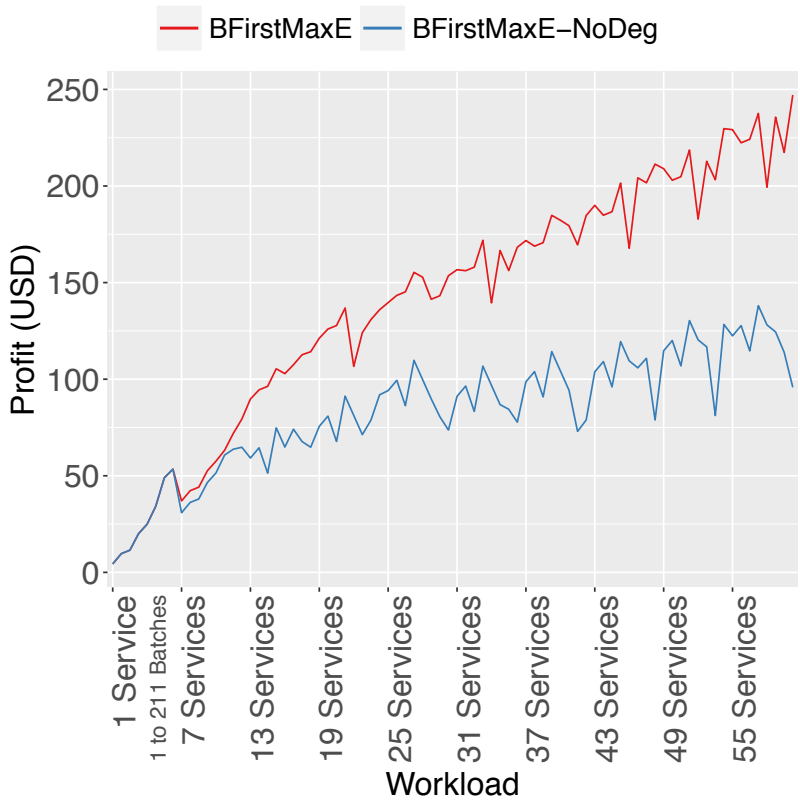


Figure 5.50: Profit impact of degradation removal considering power profile 4 and algorithm BFirstMaxE.

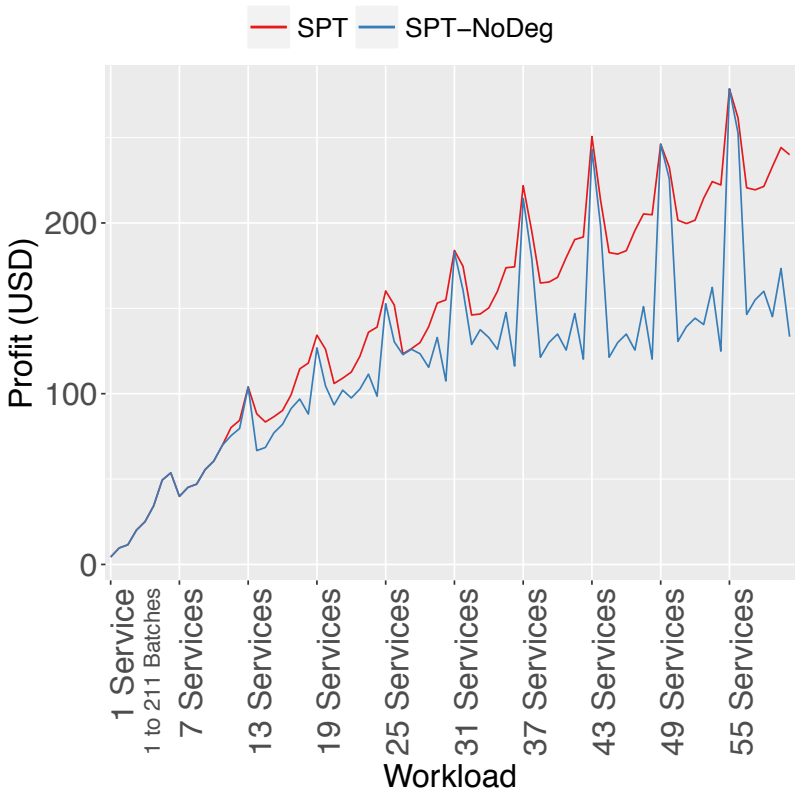


Figure 5.51: Profit impact of degradation removal considering power profile 1 and algorithm SPT.

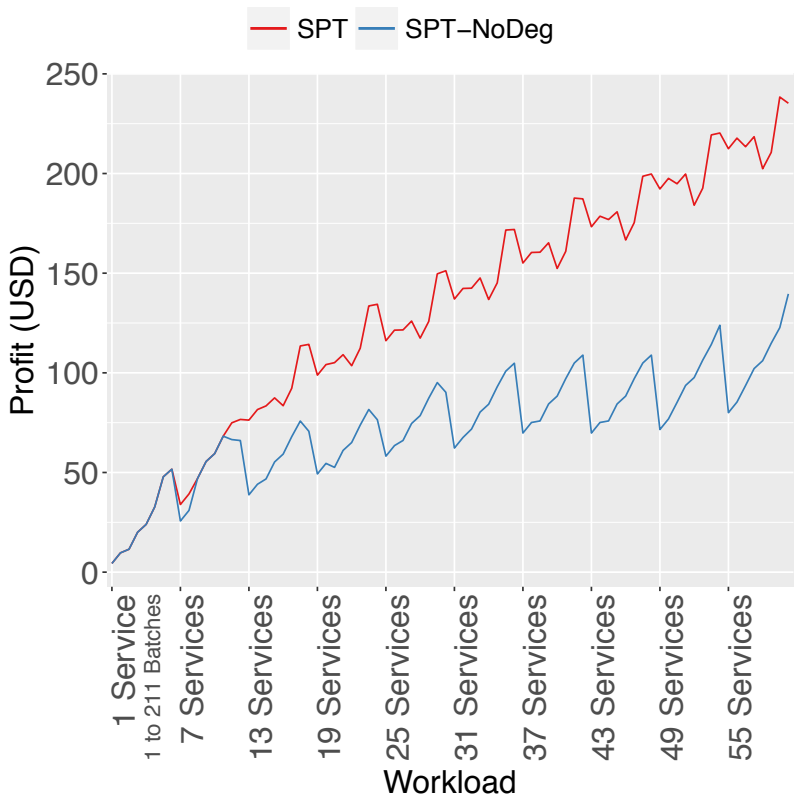


Figure 5.52: Profit impact of degradation removal considering power profile 4 and algorithm SPT.

5.7 Towards Slots Removal

As a validation to the impact of dividing tasks in slots and an initial approach to improve the execution time of the algorithms, we propose a tree search structure. The aim is to find the first valid start point where the constraints (power and IT) would not be violated.

We illustrate in Figure 5.53 a simplified version where each node of the tree would store the minimum and maximum value (of any constraint) of its children. When the node is a leaf, the value stored would be only a single value. The time that each node represents is the number of leaf children under it. This means that each node knows the size of the interval that the minimum and maximum values that it stores represent.

Knowing the size of the task to be placed and the minimum/maximum power available one could rule out several invalid steps faster than searching in a sequential way. In Figure 5.53 we show one task (T1) of duration 4 and the steps of the search. Since the minimum power is 6 and maximum 7 we could directly discard 4 start points in step 2 and validate the constraint in step 4.

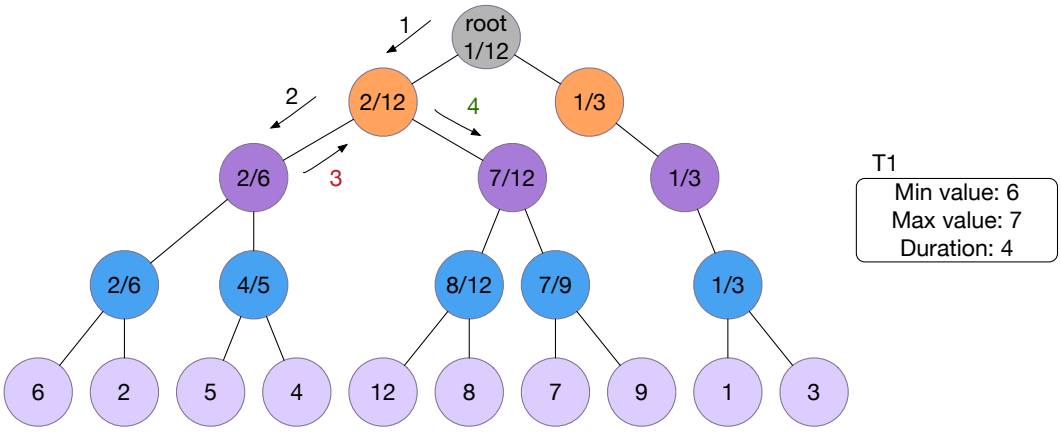


Figure 5.53: Illustration of the tree search proposition for the power constraint.

Since we are working with phases, this validation can occur considering the minimum resource phase and the maximum one, and in the best case scenario reducing significantly the search time. If the validation considering the maximum value of the whole task is not possible we would have to perform it for each phase. In our case we consider that each tree node is "multi-level" allowing us to validate/discard several constraints at each step for both IT and power. This type of structure also allow us to stop at any given layer of the tree if we want to do some kind of aggregation, similar to slots.

5.7.1 Preliminary Results

In the preliminary experiments we explore a single workload with 100 batch tasks and 58 services and slot time of 256s. We consider two algorithms, FF and MinCCFirst. In Figure 5.54 we present the variations "Slots", "Tree W/ Stop" and "Tree", the first being the approach utilized so far. The "Tree W/ Stop" represents the usage of the tree search but keeping the "stop layer" to consider $log2(256)$ i.e. stopping at layer 8. Considering that each leaf has duration of 1 second, the steps below the nodes at the 8th layer have time of 256 seconds. Finally the "Tree" would be the approach without any type of time aggregation.

In Figure 5.54 we show the execution time of each approach. Considering the stop approach we obtained an 8% time reduction for FF and 17% with MinCCFirst. The reduction in time is not as significant due to the fact that we are utilizing phases. In this case when the resources of phase 10, for instance are not valid we need to rollback and start to re-validate the constraints for the whole task again starting from the next valid start point.

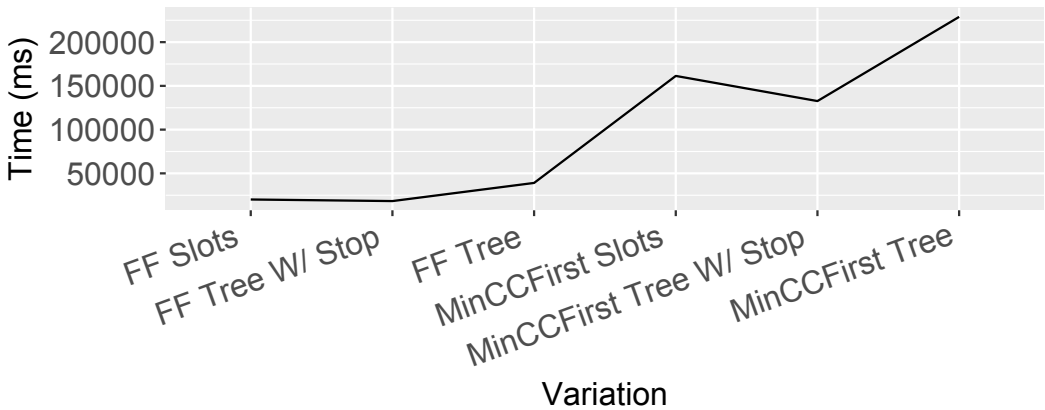


Figure 5.54: Execution time of the scheduling with and without tree search.

For the "Tree" approach without stop, there is an expected increase in the execution time, since in this case we consider no aggregation. Despite of this increase it allows us to validate the impact of slots in the results of the scheduling. In this case, we had a reduction of 1 batch violation for FF, but no difference in the results when utilizing MinCCFirst with or without slots. Another main contributing factor, specially for the cross-correlation related approach is that in this case, the cross-correlation calculation needs to be performed for a bigger number of steps.

5.8 Conclusion

This chapter presented and evaluated a phases based batch and services scheduling approach under renewable energy constraint. We presented different algorithms from literature adapted to consider phases tasks, as well as new approaches that utilize cross-correlation of resources. We also considered in all cases degradation of assigned resources when the data center is not able to deliver them, as well as the impact that this degradation has. We evaluated several different workload compositions, with different balance between profit of a given type of task, and how each algorithm behaves on these scenarios.

The algorithms were evaluated under different workloads, presenting real data center based price metrics and QoS for both kind of tasks. The results show that our approaches based on cross-correlation could maintain the profit and provide a more profitable scheduling when considering the workloads variation. Even when compared to an unbalanced workload where the majority of the profit comes from a single type of task, the cross-correlation based algorithms were able to maintain a profit as good as SFirstMaxE which would benefit in that case. The results also show that cross-correlation based approaches were able to obtain the best profit, specially in cases where less power is available due to the better resources utilization.

We also highlight the importance of the several approaches and the different power profiles and profit metrics, which could be employed in a scenario such as the one in DataZERO [101]. In this case, the negotiation loop, would be able to evaluate the several placements with different quality metrics and power consumption profiles and decide the best one.

Finally we show the impact of removing the degradation to the tasks profit, which can be up to 44%, as well as preliminary results for slots removal. In this case we could observe the challenges of finding a faster search approach to schedule phases based tasks.

Chapter 6

Towards a Multi Phases Mixed Workload Online Scheduling

In this Chapter we consider the same application and cost model as the previous one. We introduce online events such as task arrival, and resources consumption changes that could occur during the application execution. These variations are handled by reactive actions, utilizing an adaptation of some of the algorithms previously presented.

6.1 Introduction

In this Chapter we also focus on task scheduling in data center under a power constraint that comes from renewable energies. Similar to the previous approach, the data center is not connected to the grid which is applicable in the context of projects such as DataZERO[1]. In cloud computing uncertainty plays a big role where a variety of events can occur [13]. As a few we can cite dynamic performance changing, resource provisioning time variation, inaccuracy of applications runtime estimation, variation of processing times and data transmission, workload uncertainty, processing time constraints (deadline, due date) among others [117].

The pattern of instantaneous resource demand varies with time and location, and new tasks arrive constantly [44]. Such variations can lead to changes in the load distribution within the cloud and QoS degradation. This indicates the need for mechanisms that exploit the available resources in an effective way. There are tools, such as multiple cloud broker mechanisms, that explore the heterogeneity to optimize placement of virtual infrastructures across multiple providers. These tools also abstract the deployment and management of the

[1] http://datazero.org

infrastructure components, which facilitates some of the technical issues that arise when allocating resources at cloud providers [87, 120]. Nevertheless they do not tackle the optimization problem of allocating resources in a cost optimal manner, specially considering a power envelope as constraint. Most of the research conducted in this matter is driven by the workload and the power side adapts to it [35]. With this in mind, we aim at online reactive mechanisms to guarantee steady performance/profit of a mixed workload in the cloud.

Considering the literature background and following the previous approach, we consider a realistic workload composed by batch and services, where the resources consumption varies over time. We also consider the same task model that can receive less resources than requested, and the impact on QoS that it has for each task type. In this particular case, we consider a cloud environment where the workload can change dramatically and the service provider needs to react accordingly. Our focus here is to evaluate how a set of particular events that can occur in the cloud side impact metrics such as QoS and the profit. We evaluate how robust some of the previously presented algorithms are under uncertainties utilizing a set of reactive actions. Again, we take a power envelope as constraint, highlighting that the delivery of less resources to tasks is fundamental when dealing with variations in the renewable energy sources [40, 74].

The contributions of this work are the following: (i) we evaluate how the scheduling approaches for phase-based batches and services react to changes in the resources consumption and execution time; (ii) we evaluate how the scheduling approaches react if new tasks would arrive in an online way, and what is the impact on profit and QoS compared to a fully offline scheduling.

The remainder of this Chapter is organized as follows. First we introduce the problem and the optimization objectives in Sections 6.2 and 6.3 respectively. We then exhibit the proposed approach to solve the problem in Section 6.4. After we show the details of the evaluation methodology and what are the test scenarios in Section 6.5. Finally we present the results on each scenario and a comparison with the offline approach in Section 6.6 and the conclusions in Section 6.7.

6.2 Problem Statement

Since we are working with a problem that not only regards the user, but also the cloud data center as a service provider, we will continue utilizing the cost model presented in details in Section 3.6. The optimization takes place on the cloud provider side, where the objective is to schedule tasks so that the profit of the cloud provider is maximized. The main differences from the previous

approach are the uncertainty factors (here also called events) that are added. The factors are listed below:

1. Arrival of new tasks at any moment of the scheduling window;
2. Tasks that could finish earlier than predicted;
3. Tasks that could finish later than predicted;
4. Tasks that could request more resources than expected;
5. Tasks that request less resources than expected.

Considering the shared cloud resources presented in the previous Chapter and the overcommitment that is employed, a single task that does not follow the expected behavior can have an impact on several others. The challenge in this case is how to address each of the unexpected event variations and assess what is the impact on the previous scheduling decisions.

The processing capacity of a single virtual machine is time-varying due to the degree of interaction of co-located VMs. This interaction overhead (with some exceptions) is task-dependent, meaning that the changes in one virtual machine will have an impact on all the co-located ones [35]. Several efforts have been conducted to predict the tasks behavior, ranging from estimating the maximum and minimum load in a given time slot [26, 29]. Other authors develop models to well describe statistical features and the trend of the workload [13] up to hourly predictions [121]. In particular, researchers [61] were able to reduce up to 75% of the QoS violations pro-actively changing the resources provision.

Based on the premise that the forecasting of the load changes is highly predictable (able to achieve an average accuracy of up to 91% [61]), we assume that at the beginning of the task execution we are able to predict a divergence between the submitted resource consumption and the actual consumption. Therefore we react to the events at the beginning of tasks. In this approach we aim to tackle the 5 aforementioned uncertainty events that could occur in a regular cloud provider after a scheduling.

6.3 Optimization Objective

The same way as in the previous chapter, the aim is to find where and when to run every task, *i.e.* to find assignment functions σ_{core} , σ_{host} and σ_{freq} expressing that phase $\mathcal{TP}_{j,t,\varphi}$ of $\mathcal{T}_{j,t}$ runs on PE $\sigma_{\text{core}}(j,t,\varphi)$ of host $\sigma_{\text{host}}(j,t,\varphi)$, and a starting point function st expressing that $\mathcal{T}_{j,t}$ starts at time $st(j,t)$, while respecting the phases sequence order, the IT resources and the power constraints.

The problem can then be formulated as follows: maximize $\sum_j Profit_j$, while fulfilling the power constraints. The additional challenge in this case, is how to address each one of the 5 uncertainty variations in order to have the least impact on profit possible.

6.4 Proposed Approach

In this section we present the problem resolution, starting by the actions that are taken on each one of the 5 events that are addressed in this chapter. We re-utilize the proposed scheduling approaches (algorithms) shown in Subsection 5.4.4 for phase-based batches and services. In this particular case we utilize them for both scheduling the initial tasks in the bag (off-line resource allocation) and also for the new tasks that arrive (online resources allocation). The algorithms re-utilized for the online approach are: (i) FF; (ii) AMaxE; (iii) BFirstMaxE; (iv) SFirstMaxE; (v) MinCCFirst; (vi) MinCCMaxCCPC; (vii) MinCCMaxE. All the names for the algorithms which are able to handle the events are re-named with the prefix "online".

The difference from the previous approaches is that in this case we need to consider the previously scheduled tasks on the new placement i.e. several resources are already in use. In DCWoRMS simulator we create simulation triggers for all the 5 events. So whenever a tasks arrives, more or less resources are requested, or an unexpected finishing time, a new action is triggered. The way that we handle these events is the same for all the algorithms. Below we list the set of actions that occur on each case:

- Online Task Arrival: When a new task arrives we place the task utilizing the same logic of the algorithm in use, without changing the tasks that are currently placed.

- Tasks Finishing Earlier: When a task finishes earlier the first thing is to verify all the tasks running in same slots where that task would run in the original end. The aim is to remove the degradation of the tasks that would represent the highest profit gain, in case some tasks were degraded.

- Tasks Finishing Later: If a task does not finish in the expected time, we verify if it is possible to keep the placement without degradation (i.e without impacting the next tasks). If there is degradation we check if this implies deadline violation or profit loss. If this is the case we re-schedule the task utilizing the same algorithm aiming to place where the best profit could be obtained.

- Tasks Requesting More Resources: For tasks that request more resources than the expected, first we try to keep the placement without changes if there is no impact to other tasks. If there is some impact we reduce the resource usage to the maximum allowed in each slot. In case of batch, where the execution time would also change, we evaluate if it impacts tasks that are scheduled later. If there is an impact we re-schedule the current task.

- Tasks Requesting Less Resources: For tasks requesting less resources we aim to adjust the other tasks in the same slot. In this case we try to remove possible degradation that could be occurring in order to increase the profit of degraded tasks.

6.5 Evaluation Methodology

Here we also simulated the IT infrastructure using the DCWoRMS simulator [76] through several adaptations in order to support the new phases based tasks, shared resources and online events. The slot duration is fixed in 5 minutes, which is based on the minimum phase duration of our workload. The time window considered is of 2 days. The IT infrastructure inside the simulator is the same as the one presented in the previous Chapter in Section 5.5. The power production variations are also kept the same and can be seen in Figure 5.13.

The main difference in this case are the events generated when a task does not follow the resource consumption that was expected on the initial placement. We created a set of 5 events variation that are triggered on each particular case, introduced in Section 6.4. The workload utilized is the balanced one evaluated in the previous chapter in subsection 5.6.1 containing 20 services and 114 batch tasks. First we do an offline placement of the tasks in the workload. We then randomly choose a set of 15 tasks comprising a mix of 12 batches and 3 services which are utilized to test all the events variations. The goal of these experiments is to evaluate the robustness of the algorithms when resources variations occur, and how the profit can change in comparison to the offline version previously shown.

Hereafter we present the results obtained utilizing the variations of power presented previously and each one of the 5 events variations. We will first present the results considering the balanced workload (where the profit between batch tasks and services is similar) where we separate the 15 randomly selected tasks to place online. In other words, we first do an initial placement of the workload, and as the tasks execution goes by, the remainder 15 tasks will trigger interruption events to be placed as shown in Figure 6.1. These tasks

must be placed in order to not degrade the profit nor the QoS of the tasks that were already running. We then test a scenario where the same 15 tasks would have a shorted duration. In this case when the tasks are executing, they will end earlier and trigger an action notifying the scheduler. This event will then be handled according to what was detailed in the previous section. In the same way, we evaluate the same tasks having a longer duration, requesting less resources that the forecasted and finally requesting more resources than the expected. For the duration variations and for the resources consumption, we defined a fixed value of 20% arbitrarily, as to overcome the worst case forecast of Herbst et al. [61] and Tran et al. [121].

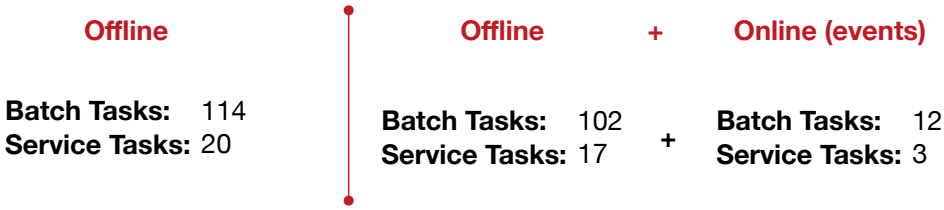


Figure 6.1: Comparison between Offline experiments methodology and Online.

6.6 Results

In this section we show the results obtained by the 5 variations proposed. First we are going to introduce the results obtained with the online task arrival. We then evaluate the tasks with shorter and longer duration, and finally the ones with more and less resources requested. The workload contains 20 services and 114 batch tasks, where 3 services and 12 batches are randomly selected as the affected by events.

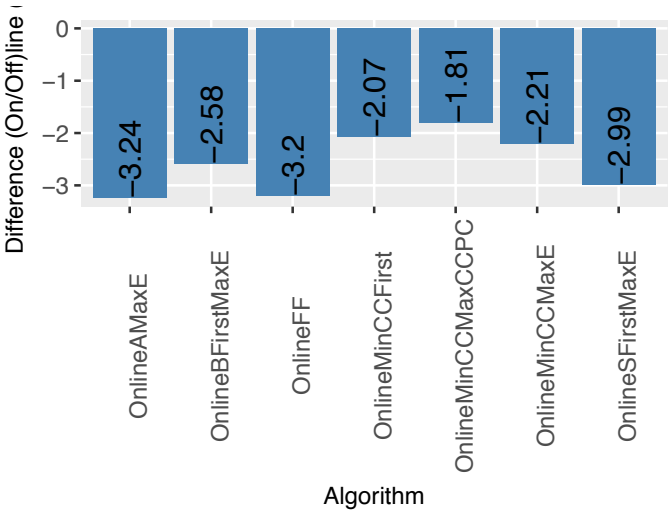
6.6.1 Online Task Arrival

We start by evaluating the profit from each type of task obtained by the algorithms.

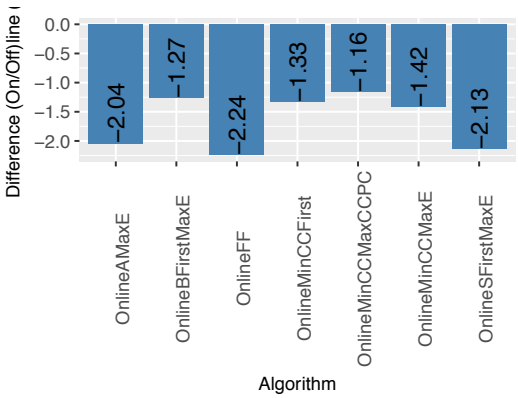
As we go from the results with the highest power, to the lower one (Profile 4) we were able to see that the lower the power, the highest is the impact on profit when placing the tasks online. More specifically, in Profile 1 the impact is lower than 1% in the total profit, but as we display in Figure 6.2 for Profile 4 the losses can be up to 3.2% for FF and AMaxE. Despite the percentage of loss seems to be low, we need to consider that this is cumulative to the losses already presented when offline scheduling took place. In other words, the algorithms that tend to perform poorly in offline schedule also perform poorly when placing the online tasks.

We also show the total profit over the 4 different power profiles when placing the selected tasks with online arrival. These results are displayed in Figures 6.3, 6.5, 6.6 and 6.7. The profit levels are kept the same, i.e. with cross-correlation variations having the highest profit in this balanced workload.

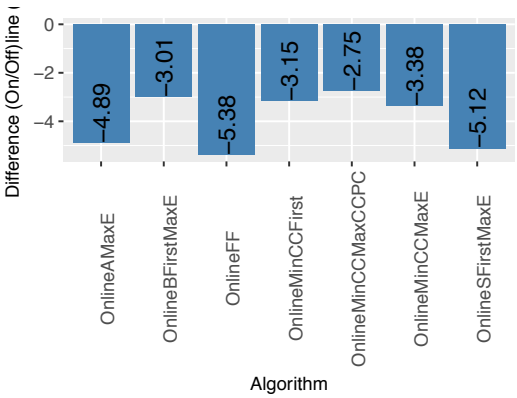
If we look at the QoS for Profile 1 and Profile 4 displayed in Figures 6.4 and 6.8 again the same trend observed in the offline approach follows. In other words BFirstMaxE having the worst QoS for services and SFirstMaxE having the worst QoS for batch tasks. Again, these degradations in QoS are just partially translated into profit, since the compensations offered by the defined model and also cloud providers have lower bound. The percentage of reduction compared to the offline approach for Profile 4 averages 4.26% for batch tasks and 6.87% for services.



(a) Reduction Total Profit

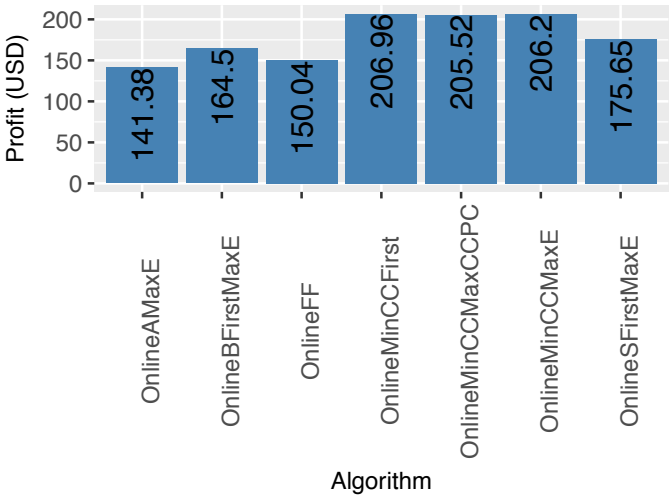


(b) Reduction Services Profit

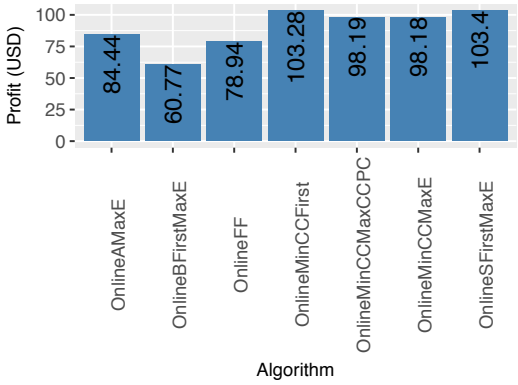


(c) Reduction Batch Profit

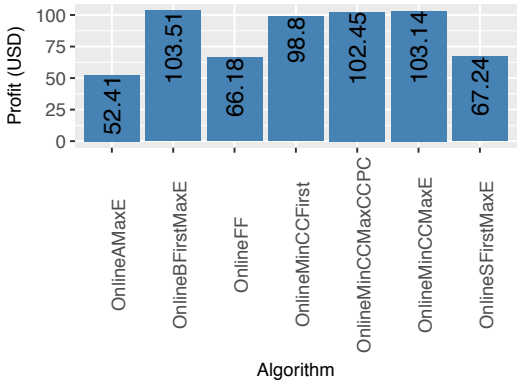
Figure 6.2: Percentage of profit reduction of workload with balanced profit for Batch and Service for all online algorithms considering online task arrival utilizing power profile 4.



(a) Total Profit

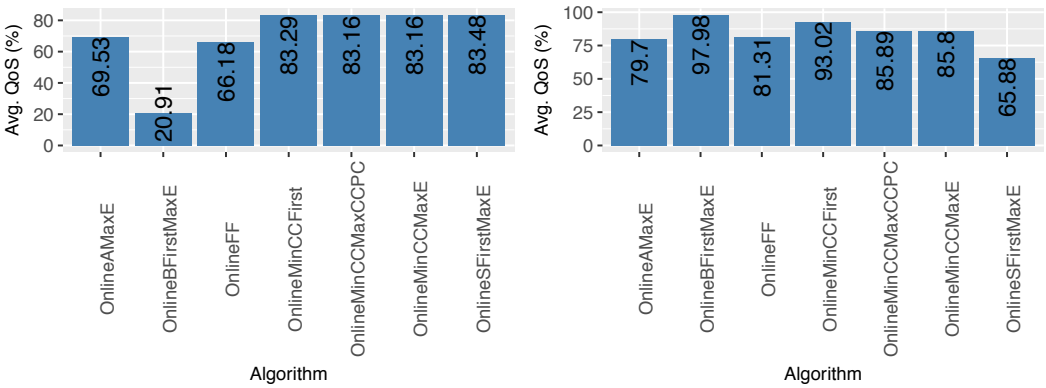


(b) Profit from Service



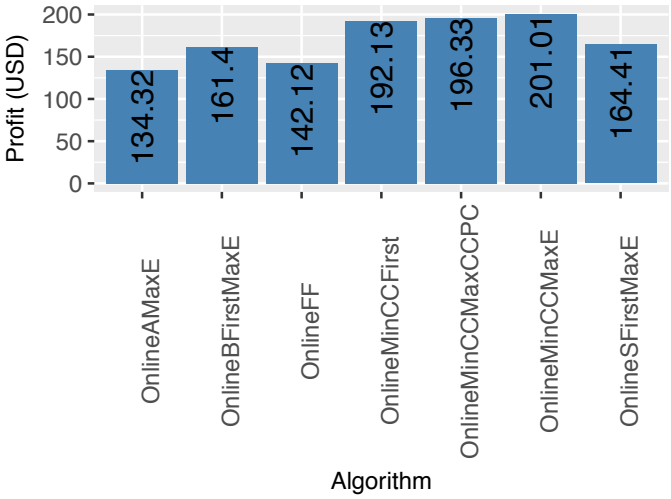
(c) Profit from Batch

Figure 6.3: Profit of workload with balanced profit for Batch and Service for all online algorithms considering online task arrival utilizing power profile 1.

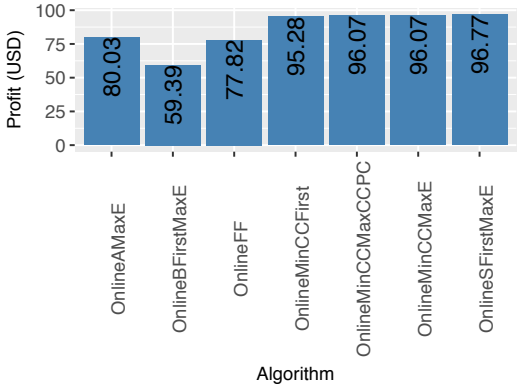


(a) QoS Service (b) QoS Batch

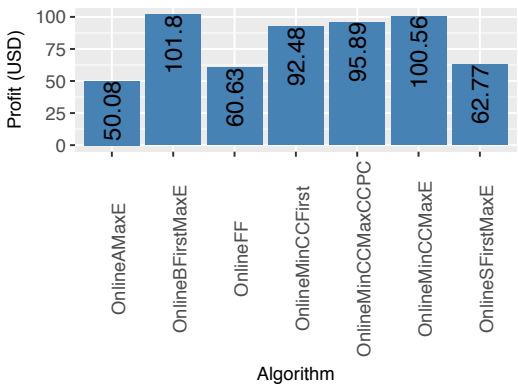
Figure 6.4: QoS of workload with balanced profit for Batch and Service for all online algorithms considering online task arrival utilizing power profile 1.



(a) Total Profit

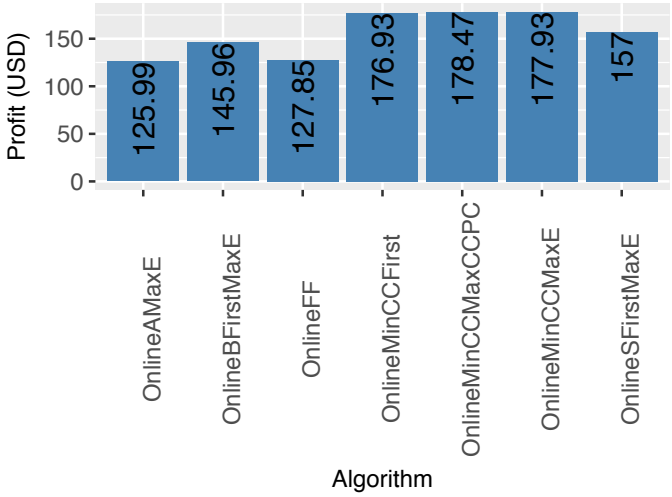


(b) Profit from Service

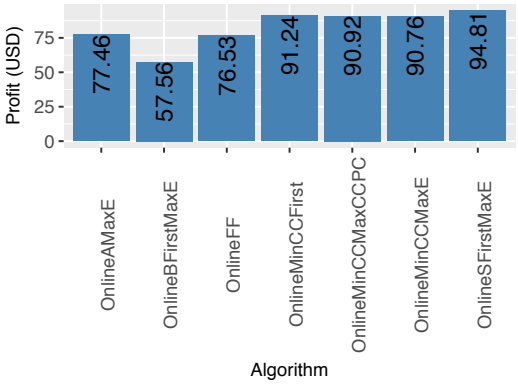


(c) Profit from Batch

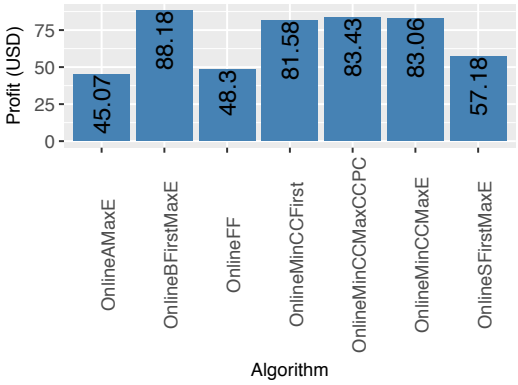
Figure 6.5: Profit of workload with balanced profit for Batch and Service for all online algorithms considering online task arrival utilizing power profile 2.



(a) Total Profit

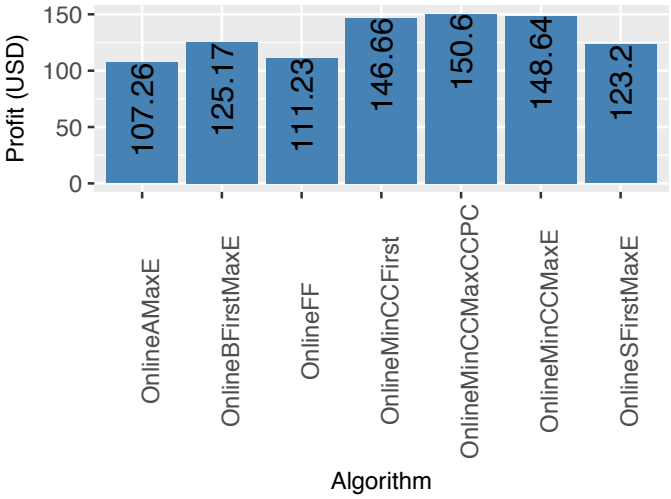


(b) Profit from Service

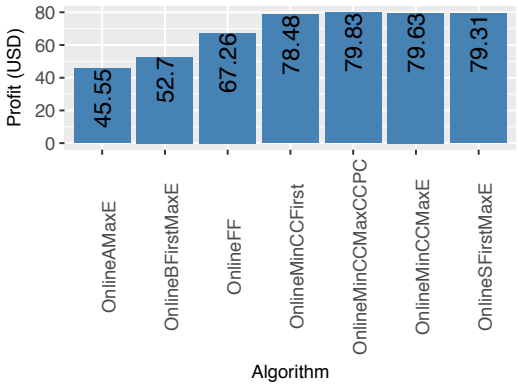


(c) Profit from Batch

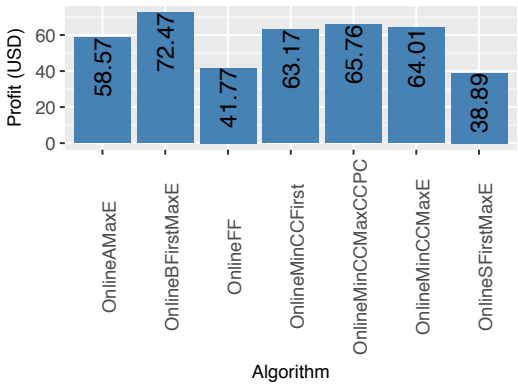
Figure 6.6: Profit of workload with balanced profit for Batch and Service for all online algorithms considering online task arrival utilizing power profile 3.



(a) Total Profit

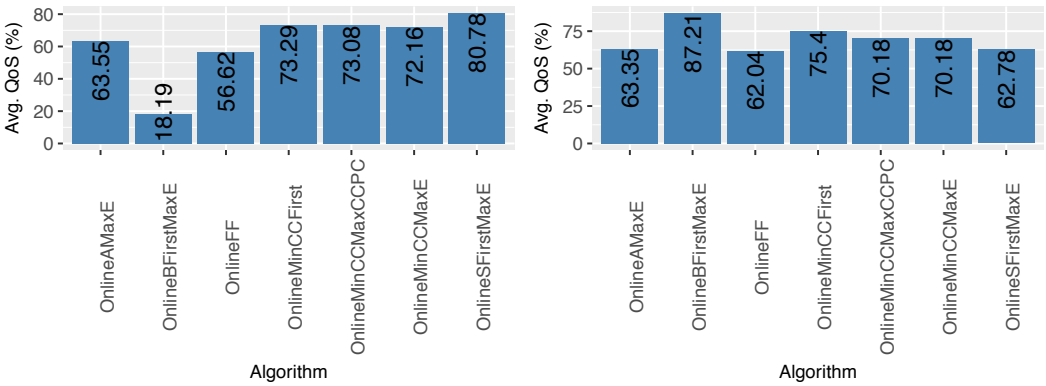


(b) Profit from Service



(c) Profit from Batch

Figure 6.7: Profit of workload with balanced profit for Batch and Service for all online algorithms considering online task arrival utilizing power profile 4.



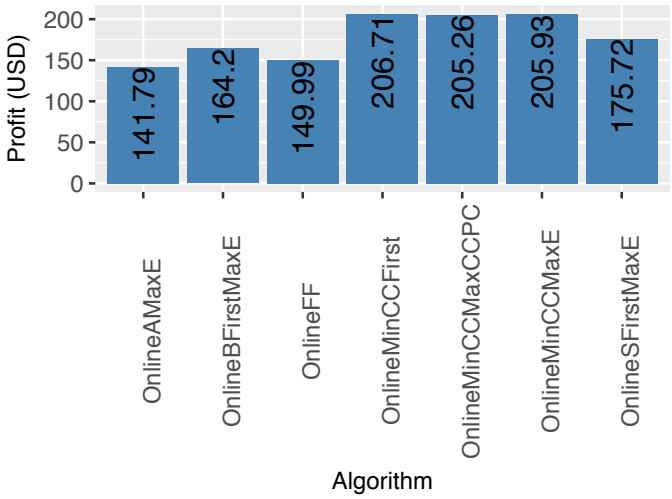
(a) QoS Service (b) QoS Batch

Figure 6.8: QoS of workload with balanced profit for Batch and Service for all online algorithms considering online task arrival utilizing power profile 4.

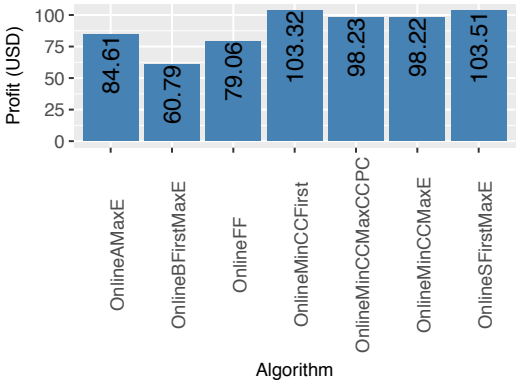
6.6.2 Tasks Have Shorter or Longer Duration

Below we evaluate the profit obtained by the algorithms over when placing the selected tasks with changes in the tasks duration. First we explore the results with reduction in the execution time of the tasks, followed by the increase in the execution time. The results with the reduction are presented in Figures 6.9 and 6.10 for Profiles 1 and 4. In Profile 1 we can see that the tasks have a reduction in the profit when compared to the offline approach, as we reduce the tasks duration. The reduction is approximately 1%, and the same behavior was observed in Profile 2. This occurs due to the natural reduction in time, since the tasks execute for a shorter period of time. According to the profit model utilized, the users pay less due to this reduction in the tasks duration. Nevertheless, we see a shift when evaluating Profiles 3 and 4, where the profit increases when the execution time of the tasks was reduced. The increase in profit has a similar low impact of less than 1% on average. In this case we attributed the increase in profit to tasks that where able to be "undegraded", which was occurring due to the more constrained power profile.

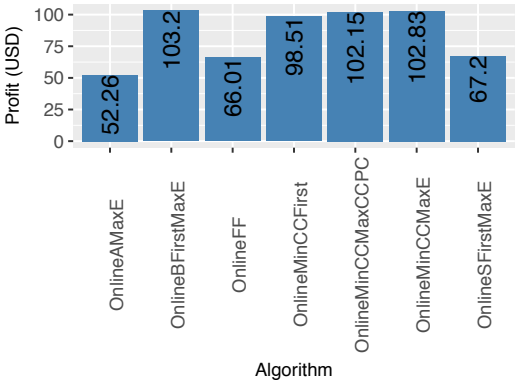
The exact opposite can be observed in the results with the increase of the execution time. We display in Figures 6.9 and 6.10 Profiles 1 and 4 respectively. Profiles 1 and 2 showed an increase in the profit, due to the increase in the tasks duration, but a reduction, specially in Profile 4. Again, due to the increase in time in a more constrained power profile, and the subsequent impact in the tasks profit.



(a) Total Profit

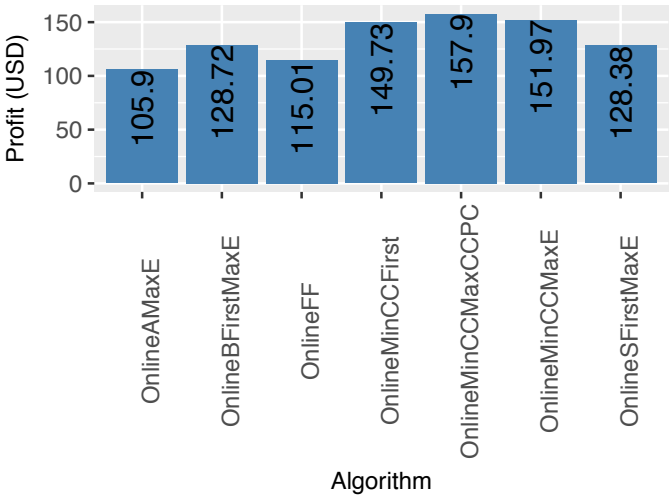


(b) Profit from Service

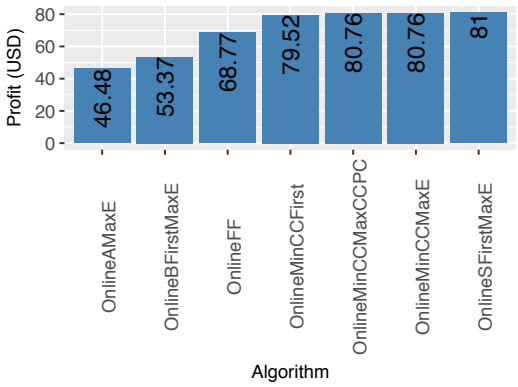


(c) Profit from Batch

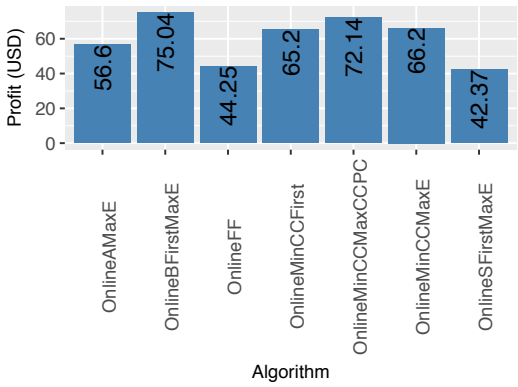
Figure 6.9: Profit of workload with balanced profit for Batch and Service for all algorithms considering less time event utilizing power profile 1.



(a) Total Profit

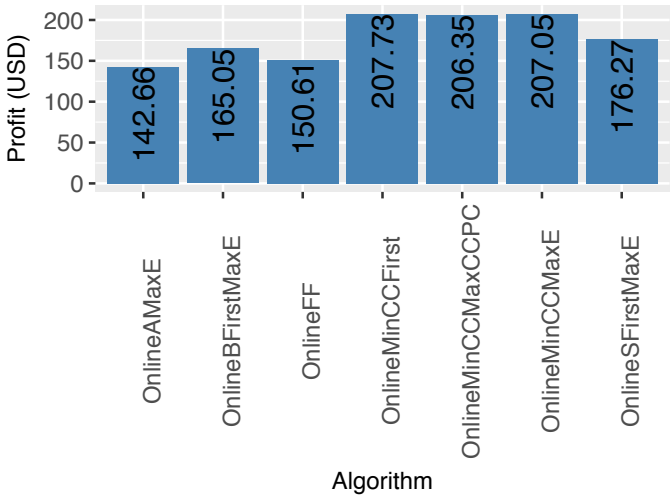


(b) Profit from Service

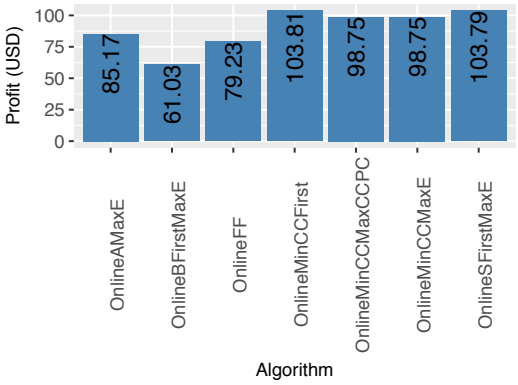


(c) Profit from Batch

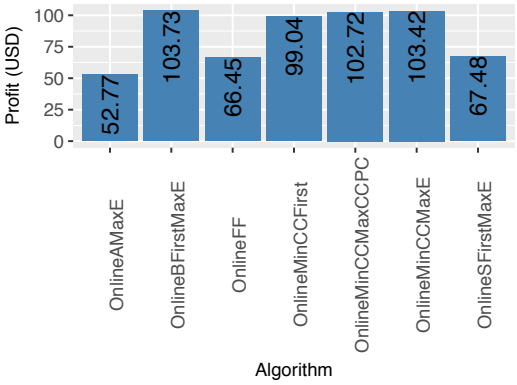
Figure 6.10: Profit of workload with balanced profit for Batch and Service for all algorithms considering less time event utilizing power profile 4.



(a) Total Profit

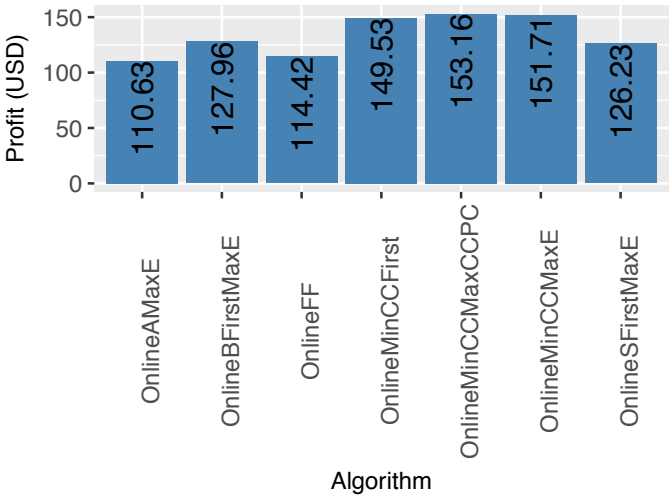


(b) Profit from Service

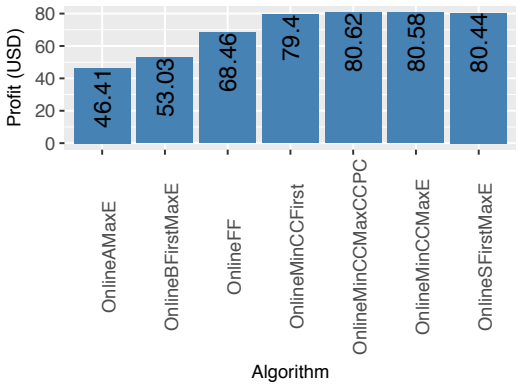


(c) Profit from Batch

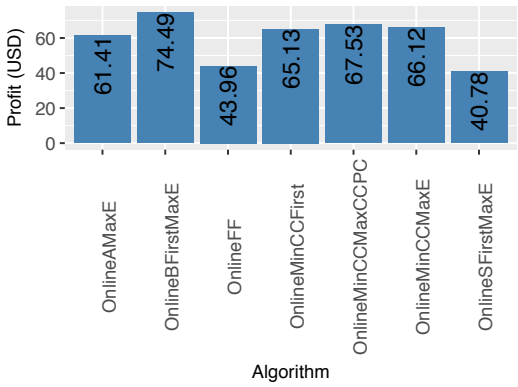
Figure 6.11: Profit of workload with balanced profit for Batch and Service for all algorithms considering more time event utilizing power profile 1.



(a) Total Profit



(b) Profit from Service



(c) Profit from Batch

Figure 6.12: Profit of workload with balanced profit for Batch and Service for all algorithms considering more time event utilizing power profile 4.

6.6.3 Tasks Request Less or More Resources

Finally we evaluate the profit obtained by the algorithms over when placing the selected tasks with changes in the tasks resource consumption. This change occur with the same percentage for all resources (20%). First we explore the results with reduction in the resources utilized by the tasks, followed by the increase in the resource consumption. The results with the reduction are presented in Figures 6.13 and 6.14 for Profiles 1 and 4. In all the cases where there is a reduction in the resources being consumed, there was a reduction in the profit obtained, when compared to the offline approach. The reduction that is near 1% in Profile 1 can be up to 7% for SFirstMaxE in Profile 4, again due to the impact that this increase/reduction have in our profit calculation.

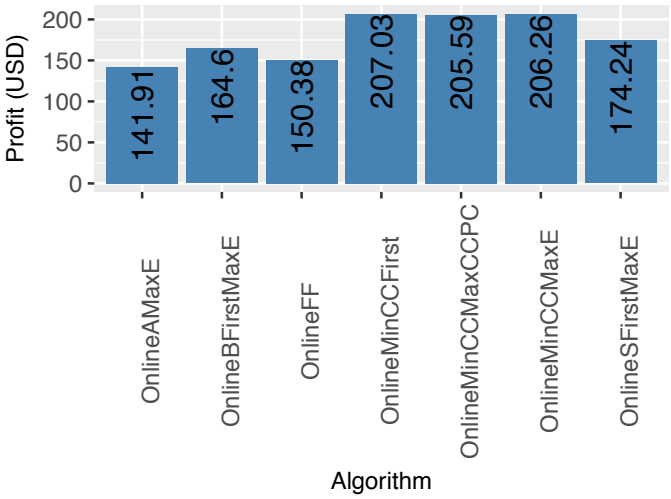
When increasing the resources consumption, we observed a small increase of up to 0.4% for AMaxE in Profile 1, displayed in Figure 6.15. Regarding the other cases, when the resources consumed increase there was also a loss in the total profit but again not significant, being up to 0.72% for BFirstMaxE in Profile 4 displayed in 6.12.

6.7 Conclusion

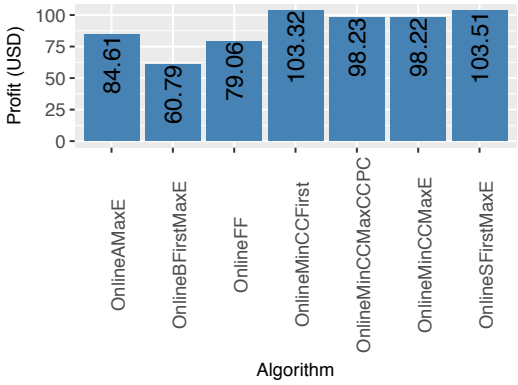
This chapter presented and evaluated the approach towards an online phase-based batch and services scheduling approach under renewable energy constraint. We presented reactive actions for different algorithms from literature adapted to consider online events, as well as new approaches that utilize cross-correlation of resources.

The algorithms were evaluated considering 5 different events that could occur in a real data center, presenting real data center based price metrics for both kind of tasks. The results show that our approaches based on cross-correlation could maintain the profit and provide a more profitable scheduling when considering the evaluated workload variation. Moreover we present the impact when comparing the whole workload being scheduled with previous knowledge of all tasks, and another approach were some of the tasks have online arrival. In this case the losses can vary from nearly 1% of the total profit up to 3.2% depending on the algorithm and power profile considered.

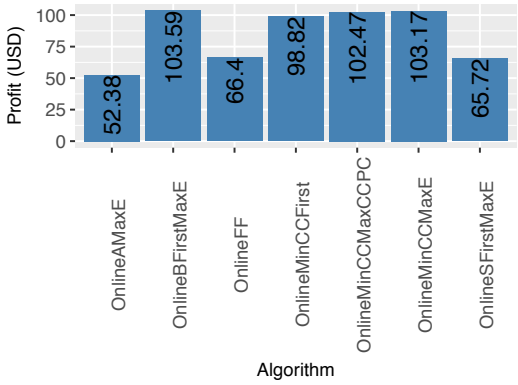
Finally we show the impact of changes in both duration and resources consumption after the tasks have been scheduled in the total profit, and that these variations did not cause a significant profit degradation for the evaluated workload and tasks selected.



(a) Total Profit

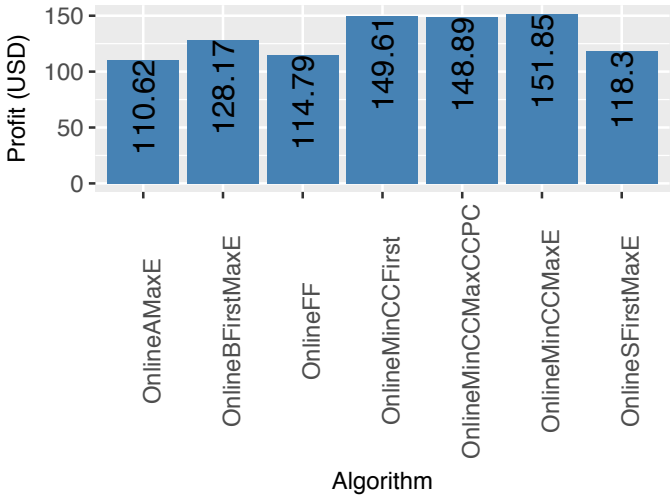


(b) Profit from Service

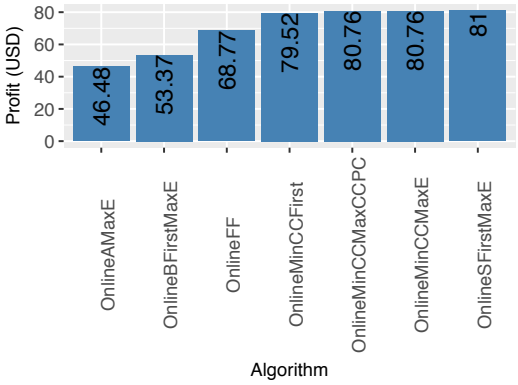


(c) Profit from Batch

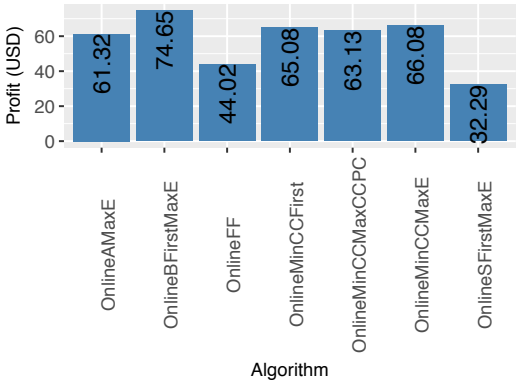
Figure 6.13: Profit of workload with balanced profit for Batch and Service for all algorithms considering less resources event utilizing power profile 1.



(a) Total Profit

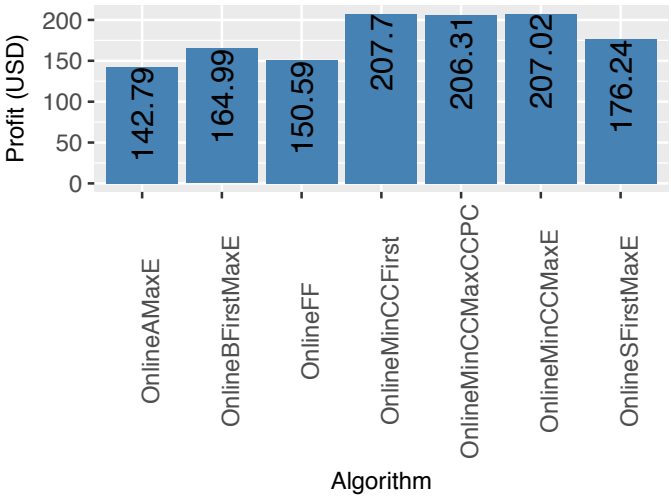


(b) Profit from Service

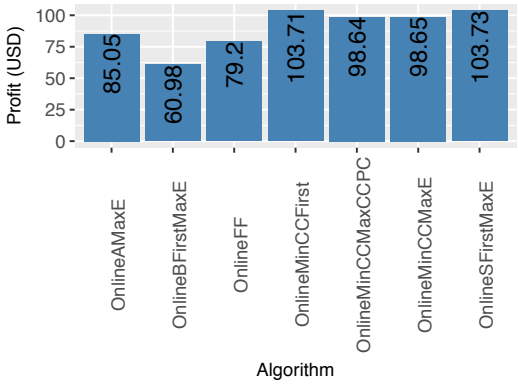


(c) Profit from Batch

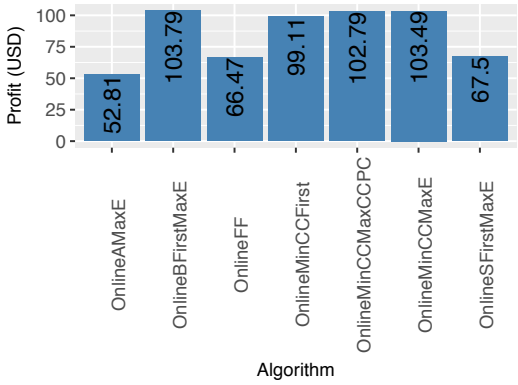
Figure 6.14: Profit of workload with balanced profit for Batch and Service for all algorithms considering less resources event utilizing power profile 4.



(a) Total Profit

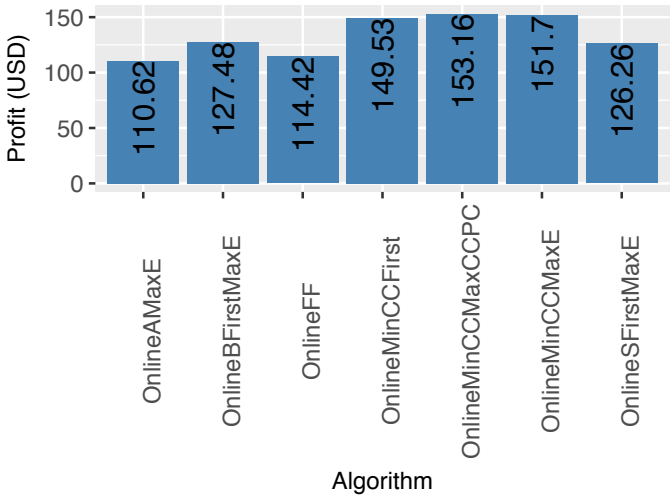


(b) Profit from Service

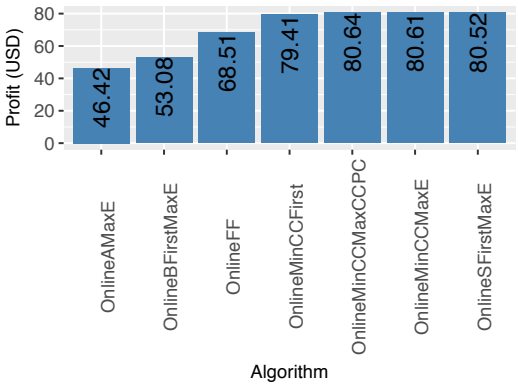


(c) Profit from Batch

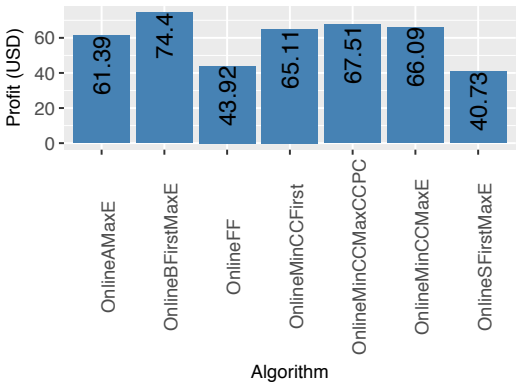
Figure 6.15: Profit of workload with balanced profit for Batch and Service for all algorithms considering more resources event utilizing power profile 1.



(a) Total Profit



(b) Profit from Service



(c) Profit from Batch

Figure 6.16: Profit of workload with balanced profit for Batch and Service for all algorithms considering more resources event utilizing power profile 4.

Chapter 7

Conclusions and Perspectives

In this chapter we make an overall conclusions and summarize the contributions of this work, along with the discussion of future work.

7.1 Conclusions

The accelerated growth of technologies that rely in cloud data centers forced an increase in its infrastructure. As we continue scaling this infrastructure, there is also a significant increase in its energy consumption. While the increases in efficiency are promising, they are yet to catch up with the growth in data traffic and consumption of cloud service [103]. It is very important to make sure this growth of cloud infrastructures is done in a way that allows the transition from traditional power sources to greener ones. In the literature, still very few works approach data centers with these breakthrough infrastructures, with only renewable energy sources, which impose several new challenges. This thesis deals with the problem of scheduling different types of tasks in a data center powered only by renewable energy, so without connection to the grid.

Different approaches have been explored, where contrary to the traditional adaptation of the power production to the workload, we adapt the workload to the power that is available. Through the presented work we approach the problem always considering metrics such as QoS and profit, while respecting the power envelopes received. This allow us to explore new infrastructures, such as the one proposed by the DataZERO project, and move towards 100% renewable data centers where the energy can be produced on site. In our approaches we utilized several techniques such as DVFS and phase-based tasks degradation, in a variety of greedy and meta-heuristic algorithms over several variations of power envelopes.

First we addressed the IT load scheduling while considering the renewable

energy available with a simplified task model. We start our approach by proposing a module to schedule single phase batch tasks, which are characterized by their release time, due date and resource demand. The goal was to maximize the Quality of Service, and to do so we propose several algorithms ranging from greedy algorithms to meta-heuristics (named MPGA and MPGA-MO). These were the first steps towards an interaction between data center electrical consumption and renewable power sources. In this case we evaluated both homogeneous and heterogeneous data centers where we were able to obtain an economy of up to 14.12% in the energy consumption, and reductions from 188 violations (First Fit) to 11 violations (MPGA).

After, in our second proposal we consider a more realistic workload composed by batch and services, where the resources consumption varies over time. We consider a phase-based task model that can receive less resources than requested and the impact on QoS that it has for each task type. To evaluate this approach we also introduced a cost model based on real cloud providers, considering the impacts that degradations have on it as well. Moreover, we include the concept of cross-correlation to schedule this mix of tasks, as well as several approaches adapted from the literature. We evaluate extensively this approach under several workloads with different compositions such as the number of batch and services, and different power profiles. Also, in this chapter we show the beneficial impact that the phases degradation can have in the profit and the steps towards improving the execution time of some of the algorithms.

Finally, following the previous approach, we take in consideration the fact that the workload in a cloud environment can be changed and the service provider needs to react accordingly. In this case we focus in evaluating how a set of particular events that can occur in the cloud side impact metrics such as QoS and the profit. We evaluate tasks not having the same duration, nor the resources consumption that was expected. And, most importantly we evaluate the difference between the offline placement of all tasks versus the placement of several tasks arriving in real time mixed with some tasks already scheduled. This allow us to see how robust some of the previously presented algorithms are under uncertainties utilizing, and the effectiveness of a set of reactive actions. Again, we take a power envelope as constraint, highlighting that the delivery of less resources to tasks is fundamental when dealing with variations in the renewable energy sources.

7.2 Perspectives

The work presented in this thesis is the primary steps towards having data centers powered only by renewable energy sources. As perspectives there are

several remaining directions to further improve our approach as well as new challenges that have to be addressed.

Execution Time Related Research

The majority of the algorithms utilized in the phases based approach rely on the usage of slots. This can be a bottleneck in the performance as the time window evaluated increase or the slots time is reduced. The utilization of slots provides aggregated data which may hide some variations in resources consumption. One of the possible solutions is to utilize tree search algorithms, such as the one presented in the end of our corresponding chapter. Another possibilities that could be explored is the change in the slots duration dynamically according to the size of the phases, or the interval between the power or tasks variation. Another aspect that could be improved is the time that the cross-correlation algorithms take to calculate it, or utilizing image processing techniques which have similar goal [130] (finding similarity/difference between images).

Infrastructure Related Research

Even though we evaluated the impact of heterogeneity in the initial approach without phases, there is still need to further evaluate it in the phases based approaches. Cross-Correlation and other phase-based algorithms need to be tested to verify how heterogeneity would impact them. This impact is significant specially when considering batch tasks, where the duration and the degradation impact would be different depending on the processing capability of the node. All the simulated experiments would also greatly benefit from a cross-validation when executing them in a real infrastructure. Another key aspect is to take in consideration the other elements that consume energy in a data center such as cooling, storage and network equipments. The objective regarding the power consumption could also be further explored. Assuming a scenario where all the energy storage devices are full, the objective could change to consume as much as possible in certain intervals of time.

Algorithms and Workload Related Research

As to our current evaluation we compared as if only one of the algorithm was utilized to provide the IT scheduling. Another approach to the problem could rely on the utilization of machine learning techniques to pre-evaluate the workload and choose the best algorithm similar to the one in Gudu et al. [57]. The implementation of an optimal solution for phases with degradation would also be beneficial, since it could be utilized as comparison base. This would allow us to evaluate how far the meta-heuristics and heuristics are from the optimal solution. A thorough evaluation of all uncertainty aspects of a data center powered only with renewable energies, as well as the evaluation of forecast errors impact would also provide a better knowledge on the robustness of the algorithms.

Regarding the workload, our evaluation comprise only Google traces based batch tasks and services from a business-critical data center. Workloads with different characteristics could also be explored, such as the ones coming from new IoT/Smart City applications, once those have been characterized. Another key point would be the evaluation in a real environment the feasibility to obtain such detailed knowledge of tasks (know each phase). Diouri et al. [38] evaluate the partial phase recognition and system adaptation in the HPC scenario, yet further evaluation on a more diverse system, such as the one presented with a mix of batch and service tasks, is necessary. One possible alternative would be to utilize methods such as Markov-Chain or others listed in [13, 26, 29] to predict and take reactive actions in case of mis-estimation of the resources in a phase. Finally, for the degradation of phases we assume that a degradation implies a proportional impact on resources, which also should be validated in a real infrastructure.

Physical Location of Processing Elements Related Research
To the best of our knowledge, there is no evaluation of feasibility of geographically distributed data centers powered only by renewable energy (no grid as backup). The closest to a geographical load balancing with renewable energy is presented in Liu et al. [85]. This kind of scenario can be specially interesting when considering no connection to the grid, since the workload could be migrated among different geographical locations, according to its renewable energy production forecast. As we move more towards significant amounts of data coming from smart cities/IoT devices, another key aspect that should be integrated is Edge and Fog computing [14] where those devices should also be in charge of processing part of its data, offloading the data center.

Appendices

Appendix A

Workload Format Description

A.1 Single Phase Batch Tasks

```
<job id="0">
   <task id="0">
      <requirements>
         <resourceRequirements>
            <computingResource>
               <hostParameter name="cpucount">
               <value>1</value>
               </hostParameter>
               <hostParameter name="memory">
               <value>1024</value>
               </hostParameter>
            </computingResource>
         </resourceRequirements>
      </requirements>
      <executionTime>
         <executionDuration>PT2M17S</executionDuration>
         <timePeriod>
            <periodStart>2016-06-01T05:59:17+00:00</periodStart>
            <periodEnd>2016-06-01T06:06:34+00:00</periodEnd>
         </timePeriod>
      </executionTime>
   </task>
   <task id="1">
      <requirements>
         <resourceRequirements>
            <computingResource>
               <hostParameter name="cpucount">
               <value>1</value>
               </hostParameter>
               <hostParameter name="memory">
```

```
                <value>1024</value>
                </hostParameter>
            </computingResource>
        </resourceRequirements>
    </requirements>
    <executionTime>
        <executionDuration>PT3M10S</executionDuration>
        <timePeriod>
            <periodStart>2016-06-01T05:59:17+00:00</periodStart>
            <periodEnd>2016-06-01T06:06:34+00:00</periodEnd>
        </timePeriod>
    </executionTime>
  </task>
</job>
```

A.2 Multi Phase Batch Tasks

```
<job id="0">
  <task id="0">
    <requirements>
        <resourceRequirements>
            <computingResource>
                <hostParameter name="cpucount">
                <value>1</value>
                </hostParameter>
                <hostParameter name="memory">
                <value>1024</value>
                </hostParameter>
            </computingResource>
        </resourceRequirements>
    </requirements>
    <executionTime>
        <executionDuration>PT3H29M49S</executionDuration>
        <timePeriod>
            <periodStart>2016-06-01T06:13:12+00:00</periodStart>
            <periodEnd>2016-06-01T22:54:36+00:00</periodEnd>
        </timePeriod>
    </executionTime>
  <execution>
    <resourceConsumptionProfile>
            <resourceConsumption>
                <duration>PT1H44M54S</duration>
                <behaviour name="RefCores">
                    <value>1.0</value>
                </behaviour>
                <behaviour name="RefFreq">
```

```
30                  <value>1000000.0</value>
                 </behaviour>
32               <behaviour name="PM_CPU_Usage">
                    <value>0.220286485236</value>
34               </behaviour>
                 <behaviour name="PM_Memory_Usage">
36                  <value>1024</value>
                 </behaviour>
38               <behaviour name="PM_Download">
                    <value>535.666667</value>
40               </behaviour>
                 <behaviour name="PM_Upload">
42                  <value>2.366667</value>
                 </behaviour>
44            </resourceConsumption>
              <resourceConsumption>
46               <duration>PT1H44M54S</duration>
                 <behaviour name="RefCores">
48                  <value>1.0</value>
                 </behaviour>
50               <behaviour name="RefFreq">
                    <value>1000000.0</value>
52               </behaviour>
                 <behaviour name="PM_CPU_Usage">
54                  <value>0.612593847338</value>
                 </behaviour>
56               <behaviour name="PM_Memory_Usage">
                    <value>1024</value>
58               </behaviour>
                 <behaviour name="PM_Download">
60                  <value>235.666667</value>
                 </behaviour>
62               <behaviour name="PM_Upload">
                    <value>5.366667</value>
64               </behaviour>
              </resourceConsumption>
66       </resourceConsumptionProfile>
    </execution>
68  </task>
    <task id="1">
70      <requirements>
           <resourceRequirements>
72            <computingResource>
                 <hostParameter name="cpucount">
74               <value>1</value>
                 </hostParameter>
76               <hostParameter name="memory">
                 <value>1024</value>
78               </hostParameter>
```

```
            </computingResource>
80        </resourceRequirements>
      </requirements>
82    <executionTime>
        <executionDuration>PT3H29M49S</executionDuration>
84      <timePeriod>
            <periodStart>2016-06-01T06:13:12+00:00</periodStart>
86          <periodEnd>2016-06-01T22:54:36+00:00</periodEnd>
        </timePeriod>
88    </executionTime>
  <execution>
90    <resourceConsumptionProfile>
            <resourceConsumption>
92              <duration>PT1H44M54S</duration>
                <behaviour name="RefCores">
94                  <value>1.0</value>
                </behaviour>
96              <behaviour name="RefFreq">
                    <value>1000000.0</value>
98              </behaviour>
                <behaviour name="PM_CPU_Usage">
100                 <value>0.369871486176</value>
                </behaviour>
102             <behaviour name="PM_Memory_Usage">
                    <value>1024</value>
104             </behaviour>
                <behaviour name="PM_Download">
106                 <value>435.666667</value>
                </behaviour>
108             <behaviour name="PM_Upload">
                    <value>234.366667</value>
110             </behaviour>
            </resourceConsumption>
112         <resourceConsumption>
                <duration>PT1H44M54S</duration>
114             <behaviour name="RefCores">
                    <value>1.0</value>
116             </behaviour>
                <behaviour name="RefFreq">
118                 <value>1000000.0</value>
                </behaviour>
120             <behaviour name="PM_CPU_Usage">
                    <value>0.402574923247</value>
122             </behaviour>
                <behaviour name="PM_Memory_Usage">
124                 <value>1024</value>
                </behaviour>
126             <behaviour name="PM_Download">
                    <value>15.666667</value>
```

```
                </behaviour>
                <behaviour name="PM_Upload">
                    <value>232.366667</value>
                </behaviour>
            </resourceConsumption>
        </resourceConsumptionProfile>
    </execution>
    </task>
</job>
```

A.3 Multi Phase Services

```
<job id="0">
 <task id="0">
    <requirements>
       <resourceRequirements>
          <computingResource>
             <hostParameter name="cpucount">
             <value>4</value>
             </hostParameter>
             <hostParameter name="memory">
             <value>65536</value>
             </hostParameter>
          </computingResource>
       </resourceRequirements>
    </requirements>
   <execution>
   <resourceConsumptionProfile>
    <resourceConsumption>
        <duration>PT10M</duration>
        <behaviour name="RefCores">
            <value>4.000000</value>
        </behaviour>
        <behaviour name="RefFreq">
            <value>2400000.000000</value>
        </behaviour>
        <behaviour name="PM_CPU_Usage">
            <value>0.931417</value>
        </behaviour>
        <behaviour name="PM_Memory_Usage">
            <value>6291.454297</value>
        </behaviour>
        <behaviour name="PM_Download">
            <value>0.000000</value>
        </behaviour>
        <behaviour name="PM_Upload">
```

```
            <value>2.366667</value>
        </behaviour>
    </resourceConsumption>
    <resourceConsumption>
        <duration>PT5M</duration>
        <behaviour name="RefCores">
            <value>4.000000</value>
        </behaviour>
        <behaviour name="RefFreq">
            <value>2400000.000000</value>
        </behaviour>
        <behaviour name="PM_CPU_Usage">
            <value>0.891500</value>
        </behaviour>
        <behaviour name="PM_Memory_Usage">
            <value>8738.131250</value>
        </behaviour>
        <behaviour name="PM_Download">
            <value>535.666667</value>
        </behaviour>
        <behaviour name="PM_Upload">
            <value>23.933333</value>
        </behaviour>
    </resourceConsumption>
    <resourceConsumption>
        <duration>PT5H15M2S</duration>
        <behaviour name="RefCores">
            <value>4.000000</value>
        </behaviour>
        <behaviour name="RefFreq">
            <value>2400000.000000</value>
        </behaviour>
        <behaviour name="PM_CPU_Usage">
            <value>0.125787</value>
        </behaviour>
        <behaviour name="PM_Memory_Usage">
            <value>275.291351</value>
        </behaviour>
        <behaviour name="PM_Download">
            <value>0.001062</value>
        </behaviour>
        <behaviour name="PM_Upload">
            <value>1.005294</value>
        </behaviour>
    </resourceConsumption>
   </resourceConsumptionProfile>
   </execution>
 </task>
</job>
```

Appendix B

Cloud Price List

B.1 Amazon EC2

B.2 Azure

Table B.1: Resouces and price of On-Demand Amazon EC2 instances evaluated.

Label	**vCPU**	**ECU**	**Memory(GB)**	**Price(USD/Hour)**	**Instance Type**
t2.nano	1	N/A	0.5	0.0058	General
t2.micro	1	N/A	1	0.0116	General
t2.small	1	N/A	2	0.023	General
t2.medium	2	N/A	4	0.0464	General
t2.large	2	N/A	8	0.0928	General
t2.xlarge	4	N/A	16	0.1856	General
t2.2xlarge	8	N/A	32	0.3712	General
m5.large	2	10	8	0.096	General
m5.xlarge	4	15	16	0.192	General
m5.2xlarge	8	31	32	0.384	General
m5.4xlarge	16	61	64	0.768	General
m5.12xlarge	48	173	192	2.304	General
m5.24xlarge	96	345	384	4.608	General
m4.large	2	6.5	8	0.1	General
m4.xlarge	4	13	16	0.2	General
m4.2xlarge	8	26	32	0.4	General
m4.4xlarge	16	53.5	64	0.8	General
m4.10xlarge	40	124.5	160	2	General
m4.16xlarge	64	188	256	3.2	General
m3.medium	1	3	3.75	0.067	General
m3.large	2	6.5	7.5	0.133	General
m3.xlarge	4	13	15	0.266	General
m3.2xlarge	8	26	30	0.532	General
c5.large	2	8	4	0.085	Computing
c5.xlarge	4	16	8	0.17	Computing
c5.2xlarge	8	31	16	0.34	Computing
c5.4xlarge	16	62	32	0.68	Computing
c5.9xlarge	36	139	72	1.53	Computing
c5.18xlarge	72	278	144	3.06	Computing
c4.large	2	8	3.75	0.1	Computing
c4.xlarge	4	16	7.5	0.199	Computing
c4.2xlarge	8	31	15	0.398	Computing
c4.4xlarge	16	62	30	0.796	Computing
c4.8xlarge	36	132	60	1.591	Computing
c3.large	2	7	3.75	0.105	Computing
c3.xlarge	4	14	7.5	0.21	Computing
c3.2xlarge	8	28	15	0.42	Computing
c3.4xlarge	16	55	30	0.84	Computing
c3.8xlarge	32	108	60	1.68	Computing
x1.16xlarge	64	174.5	976	6.669	Memory
x1.32xlarge	128	349	1952	13.338	Memory
x1e.xlarge	4	12	122	0.834	Memory
x1e.2xlarge	8	23	244	1.668	Memory
x1e.4xlarge	16	47	488	3.336	Memory
x1e.8xlarge	32	91	976	6.672	Memory
x1e.16xlarge	64	179	1952	13.344	Memory
x1e.32xlarge	128	340	3904	26.688	Memory
r3.large	2	6.5	15	0.166	Memory
r3.xlarge	4	13	30.5	0.333	Memory
r3.2xlarge	8	26	61	0.665	Memory
r3.4xlarge	16	52	122	1.33	Memory
r3.8xlarge	32	104	244	2.66	Memory
r4.large	2	7	15.25	0.133	Memory
r4.xlarge	4	13.5	30.5	0.266	Memory
r4.2xlarge	8	27	61	0.532	Memory
r4.4xlarge	16	53	122	1.064	Memory
r4.8xlarge	32	99	244	2.128	Memory
r4.16xlarge	64	195	488	4.256	Memory

Table B.2: Resouces and price of Pay-as-you-go Azure Cloud instances evaluated.

Label	**vCPU**	**ECU**	**Memory(GB)**	**Price(USD/Hour)**	**Instance Type**
A0	1	50	0.75	0.018	General
A1	1	100	1.75	0.025	General
A2	2	200	3.5	0.085	General
A3	4	400	7	0.188	General
A4	8	800	14	0.376	General
A1 v2	1	100	2	0.043	General
A2 v2	2	200	4	0.091	General
A4 v2	4	400	8	0.191	General
A8 v2	8	800	16	0.4	General
A2m v2	2	200	16	0.129	General
A4m v2	4	400	32	0.27	General
A8m v2	8	800	64	0.568	General
D2 v3	2	350	8	0.11	General
D4 v3	4	700	16	0.22	General
D8 v3	8	1400	32	0.44	General
D16 v3	16	2800	64	0.88	General
D32 v3	32	5600	128	1.76	General
D64 v3	64	11200	256	3.52	General
D1 v2	1	230	3.5	0.073	General
D2 v2	2	460	7	0.146	General
D3 v2	4	920	14	0.293	General
D4 v2	8	1840	28	0.585	General
D5 v2	16	3680	56	1.17	General
E2 v3	2	350	16	0.146	Memory
E4 v3	4	700	32	0.293	Memory
E8 v3	8	1400	64	0.585	Memory
E16 v3	16	2800	128	1.17	Memory
E32 v3	32	5600	256	2.341	Memory
E64 v3	64	11200	432	4.412	Memory
D11 v2	2	460	14	0.185	Memory
D12 v2	4	920	28	0.371	Memory
D13 v2	8	1840	56	0.741	Memory
D14 v2	16	3680	112	1.482	Memory
D15 v2	20	4600	140	1.853	Memory
DS11 v2	2	460	14	0.185	Memory
DS12 v2	4	920	28	0.371	Memory
DS13 v2	8	1840	56	0.741	Memory
DS14 v2	16	3680	112	1.482	Memory
DS15 v2	20	4600	140	1.853	Memory
D2 v3	2	350	8	0.11	General
D4 v3	4	700	16	0.22	General
D8 v3	8	1400	32	0.44	General
D16 v3	16	2800	64	0.88	General
D32 v3	32	5600	128	1.76	General
D64 v3	64	11200	256	3.52	General
F1	1	230	2	0.055	Computing
F2	2	460	4	0.11	Computing
F4	4	920	8	0.219	Computing
F8	8	1840	16	0.438	Computing
F16	16	3680	32	0.876	Computing
F2 v2	2	405	4	0.085	Computing
F4 v2	4	810	8	0.169	Computing
F8 v2	8	1620	16	0.338	Computing
F16 v2	16	3240	32	0.677	Computing
F32 v2	32	6480	64	1.353	Computing
F64 v2	64	12960	128	2.706	Computing
F72 v2	72	14580	144	3.045	Computing

Bibliography

[1] Alemayehu Addisu, Laurent George, Pierre Courbin, and Vincent Sciandra. Smoothing of renewable energy generation using gaussian-based method with power constraints. *Energy Procedia*, 134:171 – 180, 2017. ISSN 1876-6102. doi: https://doi.org/10.1016/j.egypro.2017.09.555. URL http://www.sciencedirect.com/science/article/pii/S1876610217346866. Sustainability in Energy and Buildings 2017: Proceedings of the Ninth KES International Conference, Chania, Greece, 5-7 July 2017.

[2] Pragati Agrawal and Shrisha Rao. Energy-efficient scheduling: Classification, bounds, and algorithms. *CoRR*, abs/1609.06430, 2016. URL http://arxiv.org/abs/1609.06430.

[3] Baris Aksanli, Jagannathan Venkatesh, Liuyi Zhang, and Tajana Rosing. Utilizing green energy prediction to schedule mixed batch and service jobs in data centers. In *Proceedings of the 4th Workshop on Power-Aware Computing and Systems*, HotPower '11, pages 5:1–5:5, NY, NY, USA, 2011. ACM. ISBN 978-1-4503-0981-3. doi: 10.1145/2039252.2039257. URL http://doi.acm.org/10.1145/2039252.2039257.

[4] Alibaba Cloud. Alibaba cloud, 2019. https://us.alibabacloud.com/.

[5] Amazon. Amazon compute service level agreement, 2019. URL https://aws.amazon.com/compute/sla/.

[6] Amazon EC2. Amazon ec2, 2019. https://aws.amazon.com/ec2/.

[7] Khosravi Atefeh, Nadjaran Toosi Adel, and Buyya Rajkumar. Online virtual machine migration for renewable energy usage maximization in geographically distributed cloud datacenters. *Concurrency and Computation: Practice and Experience*, 29(18):e4125, 2017. doi: 10.1002/cpe.4125. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4125.

[8] G. S. Aujla, N. Kumar, S. Garg, K. Kaur, R. Rajan, and S. Garg. Renewable energy-based multi-indexed job classification and container management scheme for sustainability of cloud data centers. *IEEE Transactions on Industrial Informatics*, pages 1–1, 2018. ISSN 1551-3203. doi: 10.1109/TII.2018.2800693.

[9] Elaine Barker, Miles Smid, and Dennis Branstad. A profile for us federal cryptographic key management systems. *NIST Special Publication*, 800: 152, 2014.

[10] Salman A. Baset. Cloud slas: Present and future. *SIGOPS Oper. Syst. Rev.*, 46(2):57–66, July 2012. ISSN 0163-5980. doi: 10.1145/2331576.2331586. URL http://doi.acm.org/10.1145/2331576.2331586.

[11] Nicolas Beldiceanu, Bárbara Dumas Feris, Philippe Gravey, Sabbir Hasan, Claude Jard, Thomas Ledoux, Yunbo Li, Didier Lime, Gilles Madi-Wamba, Jean-Marc Menaud, Pascal Morel, Michel Morvan, Marie-Laure Moulinard, Anne-Cécile Orgerie, Jean-Louis Pazat, Olivier Roux, and Ammar Sharaiha. Towards energy-proportional clouds partially powered by renewable energy. *Computing*, 99(1):3–22, 2017. ISSN 1436-5057. doi: 10.1007/s00607-016-0503-z. URL http://dx.doi.org/10.1007/s00607-016-0503-z.

[12] J. L. Berral, I. Goiri, T. D. Nguyen, R. Gavaldá, J. Torres, and R. Bianchini. Building green cloud services at low cost. In *2014 IEEE 34th International Conference on Distributed Computing Systems*, pages 449–460, June 2014. doi: 10.1109/ICDCS.2014.53.

[13] J. Bi, H. Yuan, M. Zhou, and Q. Liu. Time-dependent cloud workload forecasting via multi-task learning. *IEEE Robotics and Automation Letters*, 4(3):2401–2406, July 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2899224.

[14] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1519-7. doi: 10.1145/2342509.2342513. URL http://doi.acm.org/10.1145/2342509.2342513.

[15] D. Borgetto and P. Stolf. An energy efficient approach to virtual machines management in cloud computing. In *2014 IEEE 3rd Int. Conference on*

Cloud Networking (CloudNet), pages 229–235, Oct 2014. doi: 10.1109/CloudNet.2014.6968997.

[16] D. Borgetto, R. Chakode, B. Depardon, C. Eichler, J. M. Garcia, H. Hbaieb, T. Monteil, E. Pelorce, A. Rachdi, A. Al Sheikh, and P. Stolf. Hybrid approach for energy aware management of multi-cloud architecture integrating user machines. *Journal of Grid Computing*, 14(1): 91–108, Mar 2016. ISSN 1572-9184. doi: 10.1007/s10723-015-9342-y. URL https://doi.org/10.1007/s10723-015-9342-y.

[17] Paul Bourke. *Cross correlation*. 1996.

[18] Ilhem Boussaad, Julien Lepagnot, and Patrick Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237: 82 – 117, 2013. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2013.02.041. URL http://www.sciencedirect.com/science/article/pii/S0020025513001588. Prediction, Control and Diagnosis using Advanced Neural Computations.

[19] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, January 2011. ISSN 1097-024X. doi: 10.1002/spe.995.

[20] H. Casanova. Simgrid: A toolkit for the simulation of application scheduling. In *First IEEE/ACM International Symposium on Cluster Computing and the Grid, 2001. Proceedings*, pages 430–437, 2001. doi: 10.1109/CCGRID.2001.923223.

[21] Stéphane Caux, Paul Renaud-Goud, Gustavo Rostirolla, and Patricia Stolf. IT optimization for datacenters under renewable power constraint. In *Euro-Par 2018: Parallel Processing - 24th International Conference on Parallel and Distributed Computing, Turin, Italy, August 27-31, 2018, Proceedings*, pages 339–351, 2018. doi: 10.1007/978-3-319-96983-1_24. URL https://doi.org/10.1007/978-3-319-96983-1_24.

[22] Stephane Caux, Gustavo Rostirolla, and Patricia Stolf. Smart Datacenter Electrical Load Model for Renewable Sources Management (regular paper). In *nternational Conference on Renewable Energies and Power Quality (ICREPQ 2018), Salamanca, Spain, 21/03/18-23/03/18*, volume 16, pages 127–132, http://www.icrepq.com, avril 2018. European Association

for the Development of Renewable Energies, Environment and Power Quality. URL https://doi.org/10.24084/repqj16.231.

[23] Stéphane Caux, Paul Renaud-Goud, Gustavo Rostirolla, and Patricia Stolf. Phase-based tasks scheduling in data centers powered exclusively by renewable energy. In *Symposium on Computer Architecture and High Performance Computing (SBAC-PAD) , Bucharest, Romania, September 29 - October 02, 2019, Proceedings*, pages 0–8, 2019.

[24] Berk Celik, Gustavo Rostirolla, Stéphane Caux, Paul Renaud-Goud, and Patricia Stolf. Analysis of demand response for datacenter energy management using ga and time-of-use prices. In *IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe) , Campo Grande, MS, Brazil, September 15-18, 2019, Proceedings*, pages 0–8, 2019.

[25] Patrik Cerwall, Anette Lundvall, Peter Jonsson, Stephen Carson, Richard Moller, Andres Torres, Per Lindberg, Kati Ohman, and Athanasios Karapantelakis. Ericsson mobility report. Technical report, Ericsson, 2019.

[26] Yao-Chung Chang, Ruay-Shiung Chang, and Feng-Wei Chuang. A predictive method for workload forecasting in the cloud environment. In Yueh-Min Huang, Han-Chieh Chao, Der-Jiunn Deng, and James J. (Jong Hyuk) Park, editors, *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, pages 577–585, Dordrecht, 2014. Springer Netherlands. ISBN 978-94-007-7262-5.

[27] Muhammad Tayyab Chaudhry, Teck Chaw Ling, Atif Manzoor, Syed Asad Hussain, and Jongwon Kim. Thermal-aware scheduling in green data centers. *ACM Comput. Surv.*, 47(3):39:1–39:48, February 2015. ISSN 0360-0300. doi: 10.1145/2678278. URL http://doi.acm.org/10.1145/2678278.

[28] Niangjun Chen, Xiaoqi Ren, Shaolei Ren, and Adam Wierman. Greening multi-tenant data center demand response. *Performance Evaluation*, 91: 229–254, 2015.

[29] Zhijia Chen, Yuanchang Zhu, Yanqiang Di, and Shaochong Feng. Self-adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network. *Intell. Neuroscience*, 2015:17:17–17:17, January 2015. ISSN 1687-5265. doi: 10.1155/2015/919805. URL https://doi.org/10.1155/2015/919805.

[30] Tudor Cioara, Ionut Anghel, Marcel Antal, Sebastian Crisan, and Ioan Salomie. Data center optimization methodology to maximize the usage of locally produced renewable energy. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2015*, pages 1–8. IEEE, 2015.

[31] VNI Cisco. Cisco visual networking index: Forecast and trends, 2017–2022. Technical report, Cisco, 2019.

[32] Compaq. Internet solutions division strategy for cloud computing, 1996.

[33] Inès De Courchelle, Tom Guérout, Georges Da Costa, Thierry Monteil, and Yann Labit. Green energy efficient scheduling management. *Simulation Modelling Practice and Theory*, 2018. ISSN 1569-190X. doi: https://doi.org/10.1016/j.simpat.2018.09.011. URL http://www.sciencedirect.com/science/article/pii/S1569190X18301382.

[34] Georges Da Costa, Léo Grange, and Inès de Courchelle. Modeling, classifying and generating large-scale google-like workload. *Sustainable Computing: Informatics and Systems*, 2018.

[35] Ruben Van den Bossche, Kurt Vanmechelen, and Jan Broeckhove. Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds. *Future Generation Computer Systems*, 29 (4):973 – 985, 2013. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2012.12.012. URL http://www.sciencedirect.com/science/article/pii/S0167739X12002324. Special Section: Utility and Cloud Computing.

[36] W. Deng, F. Liu, H. Jin, B. Li, and D. Li. Harnessing renewable energy in cloud datacenters: opportunities and challenges. *IEEE Network*, 28(1): 48–55, January 2014. ISSN 0890-8044. doi: 10.1109/MNET.2014.6724106.

[37] X. Deng, D. Wu, J. Shen, and J. He. Eco-aware online power management and load scheduling for green cloud datacenters. *IEEE Systems Journal*, 10(1):78–87, 2016. doi: 10.1109/JSYST.2014.2344028.

[38] Mehdi Diouri, Ghislain Landry Tsafack Chetsa, Olivier Gluck, Laurent Lefèvre, Jean-Marc Pierson, Patricia Stolf, and Georges Da Costa. Energy efficiency in HPC: with or without knowledge on applications and services. *Int. Journal of High Performance Computing Applications*, doi:10.1177/1094342013497990:232–243, Aug 2013. URL http://oatao.univ-toulouse.fr/12431/.

[39] Corentin Dupont, Mehdi Sheikhalishahi, Federico M Facca, and Fabien Hermenier. An energy aware application controller for optimizing renewable energy consumption in data centres. In *Utility and Cloud Computing (UCC), 2015 IEEE/ACM 8th International Conference on*, pages 195–204. IEEE, 2015.

[40] Ben Elliston, Iain MacGill, and Mark Diesendorf. Least cost 100% renewable electricity scenarios in the australian national electricity market. *Energy Policy*, 59:270 – 282, 2013. ISSN 0301-4215. doi: https://doi.org/10.1016/j.enpol.2013.03.038. URL http://www.sciencedirect.com/science/article/pii/S0301421513002164.

[41] Dror G. Feitelson and Larry Rudolph. Toward convergence in job schedulers for parallel supercomputers. In Dror G. Feitelson and Larry Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 1–26, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. ISBN 978-3-540-70710-3.

[42] Gartner. Gartner says worldwide iaas public cloud services market grew 31.3in 2018, 2019. https://www.gartner.com/en/newsroom/press-releases/2019-07-29-gartner-says-worldwide-iaas-public-cloud-services-market-grew-31point3-percent-in-2018.

[43] Mahdi Ghamkhari and Hamed Mohsenian-Rad. Energy and performance management of green data centers: A profit maximization approach. *IEEE Transactions on Smart Grid*, 4(2):1017–1025, 2013.

[44] L. Gkatzikis and I. Koutsopoulos. Migrate or not? exploiting dynamic task migration in mobile cloud computing systems. *IEEE Wireless Communications*, 20(3):24–32, June 2013. ISSN 1536-1284. doi: 10.1109/MWC.2013.6549280.

[45] I. Goiri, R. Beauchea, K. Le, T. D. Nguyen, M. E. Haque, J. Guitart, J. Torres, and R. Bianchini. Greenslot: Scheduling energy consumption in green datacenters. In *2011 Int. Conf. for High Performance Computing, Networking, Storage and Analysis*, pages 1–11, Nov 2011. doi: 10.1145/2063384.2063411.

[46] I. Goiri, W. Katsak, K. Le, T. D. Nguyen, and R. Bianchini. Designing and managing data centers powered by renewable energy. *IEEE Micro*, 34(3):8–16, May-June 2014. ISSN 0272-1732. doi: 10.1109/MM.2014.6. URL doi.ieeecomputersociety.org/10.1109/MM.2014.6.

[47] Íñigo Goiri, Kien Le, Thu D. Nguyen, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. Greenhadoop: Leveraging green energy in data-processing frameworks. In *Proceedings of the 7th ACM European Conf. on Computer Systems*, EuroSys '12, pages 57–70, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1223-3. doi: 10.1145/2168836.2168843.

[48] Íñigo Goiri, William Katsak, Kien Le, Thu D Nguyen, and Ricardo Bianchini. Parasol and greenswitch: Managing datacenters powered by renewable energy. *ACM SIGARCH Computer Architecture News*, 41(1): 51–64, 2013.

[49] Inigo Goiri, Md E. Haque, Kien Le, Ryan Beauchea, Thu D. Nguyen, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. Matching renewable energy supply and demand in green datacenters. *Ad Hoc Networks*, 25: 520 – 534, 2015. doi: https://doi.org/10.1016/j.adhoc.2014.11.012.

[50] Google Cloud. Google cloud, 2019. https://cloud.google.com/.

[51] H. Goudarzi and M. Pedram. Force-directed geographical load balancing and scheduling for batch jobs in distributed datacenters. In *2013 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 1–8, Sep. 2013. doi: 10.1109/CLUSTER.2013.6702637.

[52] Léo Grange, Georges Da Costa, and Patricia Stolf. Green IT scheduling for data center powered with renewable energy. *Future Generation Computer Systems*, 86:99–120, September 2018. URL https://doi.org/10.1016/j.future.2018.03.049.

[53] GreenDataNet. Greendatanet research project, 2015. http://www.greendatanet-project.eu/.

[54] Greenpeace. Clicking Clean: Who is winning the race to build a green internet? Technical report, Greenpeace, January 2017.

[55] C. Gu, C. Liu, J. Zhang, H. Huang, and X. Jia. Green scheduling for cloud data centers using renewable resources. In *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 354–359, April 2015. doi: 10.1109/INFCOMW.2015.7179410.

[56] C. Gu, H. Huang, and X. Jia. Green scheduling for cloud data centers using esds to store renewable energy. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–7, May 2016. doi: 10.1109/ICC.2016.7511449.

[57] Diana Gudu, Marcus Hardt, and Achim Streit. Combinatorial auction algorithm selection for cloud resource allocation using machine learning. In Marco Aldinucci, Luca Padovani, and Massimo Torquati, editors, *Euro-Par 2018: Parallel Processing*, pages 378–391, Cham, 2018. Springer International Publishing.

[58] Tom Guerout, Yacine Gaoua, Christian Artigues, Georges Da Costa, Pierre Lopez, and Thierry Monteil. Mixed integer linear programming for quality of service optimization in clouds. *Future Generation Computer Systems*, 71:1 – 17, 2017. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2016.12.034. URL http://www.sciencedirect.com/science/article/pii/S0167739X1630869X.

[59] Tom Guérout, Samir Medjiah, Georges Da Costa, and Thierry Monteil. Quality of service modeling for green scheduling in clouds. *Sustainable Computing: Informatics and Systems*, 4(4): 225 – 240, 2014. ISSN 2210-5379. doi: http://dx.doi.org/10.1016/j.suscom.2014.08.006. URL http://www.sciencedirect.com/science/article/pii/S221053791400047X. Special Issue on Energy Aware Resource Management and Scheduling (EARMS).

[60] Brian Hayes. Cloud computing. *Commun. ACM*, 51(7):9–11, July 2008. ISSN 0001-0782. doi: 10.1145/1364782.1364786. URL http://doi.acm.org/10.1145/1364782.1364786.

[61] Nikolas Roman Herbst, Nikolaus Huber, Samuel Kounev, and Erich Amrehn. Self-adaptive workload classification and forecasting for proactive resource provisioning. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, ICPE '13, pages 187–198, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1636-1. doi: 10.1145/2479871.2479899. URL http://doi.acm.org/10.1145/2479871.2479899.

[62] IBM. Cplex optimizer, 2019. https://www.ibm.com/analytics/cplex-optimizer.

[63] IBM Cloud. Ibm cloud, 2019. https://www.ibm.com/cloud.

[64] Santiago Iturriaga and Sergio Nesmachnow. Scheduling energy efficient data centers using renewable energy. *Electronics*, 5(4):71, 2016.

[65] D. Jain, S. Agrawal, S. Sengupta, P. De, B. Mitra, and S. Chakraborty. Prediction of quality degradation for mobile video streaming apps: A case study using youtube. In *2016 8th International Conference on*

Communication Systems and Networks (COMSNETS), pages 1–2, Jan 2016. doi: 10.1109/COMSNETS.2016.7440005.

[66] Z. Jia, L. Wang, J. Zhan, L. Zhang, and C. Luo. Characterizing data analysis workloads in data centers. In *2013 IEEE International Symposium on Workload Characterization (IISWC)*, pages 66–76, Sep. 2013. doi: 10.1109/IISWC.2013.6704671.

[67] A. Kassab, J. M. Nicod, L. Philippe, and V. Rehn-Sonigo. Scheduling independent tasks in parallel under power constraints. In *2017 46th International Conference on Parallel Processing (ICPP)*, pages 543–552, Aug 2017. doi: 10.1109/ICPP.2017.63.

[68] Ayham Kassab, Jean-Marc Nicod, Laurent Philippe, and Veronika Rehn-Sonigo. Assessing the use of genetic algorithms to schedule independent tasks under power constraints. In *2018 International Conference on High Performance Computing & Simulation (HPCS)*, pages 252–259. IEEE, 2018.

[69] Z. Khan and S.L. Kiani. A cloud-based architecture for citizen services in smart cities. In *Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on*, pages 315–320, Nov 2012. doi: 10.1109/UCC.2012.43.

[70] Sonja Klingert, Florian Niedermeier, Corentin Dupont, Giovanni Giuliani, Thomas Schulze, and Hermann de Meer. Renewable energy-aware data centre operations for smart cities the dc4cities approach. In *Smart Cities and Green ICT Systems (SMARTGREENS), 2015 International Conference on*, pages 1–9. IEEE, 2015.

[71] Fanxin Kong and Xue Liu. A survey on green-energy-aware power management for datacenters. *ACM Comput. Surv.*, 47(2):30:1–30:38, November 2014. ISSN 0360-0300. doi: 10.1145/2642708. URL http://doi.acm.org/10.1145/2642708.

[72] Jonathan Koomey. Growth in data center electricity use 2005 to 2010. *A report by Analytical Press, completed at the request of The New York Times*, page 9, 2011.

[73] Jonathan Koomey, Kenneth Brill, Pitt Turner, John Stanley, and Bruce Taylor. A simple model for determining true total cost of ownership for data centers. *Uptime Institute White Paper, Version*, 2:2007, 2007.

[74] Vincent Krakowski, Edi Assoumou, Vincent Mazauric, and Nadia Maïzi. Reprint of feasible path toward 40 to 100% renewable energy shares for power supply in france by 2050: A prospective analysis. *Applied Energy*, 184:1529 – 1550, 2016. ISSN 0306-2619. doi: https://doi.org/10.1016/j.apenergy.2016.11.003. URL http://www.sciencedirect.com/science/article/pii/S0306261916315872.

[75] K. Kurowski, A. Oleksiak, W. Piątek, T. Piontek, A. Przybyszewski, and J. Węglarz. DCworms – A tool for simulation of energy efficiency in distributed computing infrastructures. *Simulation Modelling Practice and Theory*, 39:135–151, December 2013. ISSN 1569-190X. doi: 10.1016/j.simpat.2013.08.007.

[76] K. Kurowski, A. Oleksiak, W. Piątek, T. Piontek, A. Przybyszewski, and J. Węglarz. Dcworms – a tool for simulation of energy efficiency in distributed computing infrastructures. *Simulation Modelling Practice and Theory*, 39:135 – 151, 2013. ISSN 1569-190X. doi: http://dx.doi.org/10.1016/j.simpat.2013.08.007. URL http://www.sciencedirect.com/science/article/pii/S1569190X1300124X. S.I.Energy efficiency in grids and clouds.

[77] K. Le, R. Bianchini, T. D. Nguyen, O. Bilgir, and M. Martonosi. Capping the brown energy consumption of internet services at low cost. In *International Conference on Green Computing*, pages 3–14, Aug 2010. doi: 10.1109/GREENCOMP.2010.5598305.

[78] Kien Le, Ozlem Bilgir, Ricardo Bianchini, Margaret Martonosi, and Thu D. Nguyen. Managing the cost, energy consumption, and carbon footprint of internet services. *SIGMETRICS Perform. Eval. Rev.*, 38 (1):357–358, June 2010. ISSN 0163-5999. doi: 10.1145/1811099.1811085. URL http://doi.acm.org/10.1145/1811099.1811085.

[79] Hongtao Lei, Tao Zhang, Yajie Liu, Yabing Zha, and Xiaomin Zhu. SGEESS: Smart green energy-efficient scheduling strategy with dynamic electricity price for data center. *Journal of Systems and Software*, 108:23 – 38, 2015. doi: https://doi.org/10.1016/j.jss.2015.06.026.

[80] Hongtao Lei, Rui Wang, Tao Zhang, Yajie Liu, and Yabing Zha. A multi-objective co-evolutionary algorithm for energy-efficient scheduling on a green data center. *Computers & Op. Research*, 75:103–117, 2016.

[81] X. Li, Y. Li, T. Liu, J. Qiu, and F. Wang. The method and tool of cost analysis for cloud computing. In *2009 IEEE International Conference on Cloud Computing*, pages 93–100, Sep. 2009. doi: 10.1109/CLOUD.2009.84.

[82] Y. Li, A. C. Orgerie, and J. M. Menaud. Balancing the use of batteries and opportunistic scheduling policies for maximizing renewable energy consumption in a cloud data center. In *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 408–415, March 2017. doi: 10.1109/PDP.2017.24.

[83] Y. Li, A. C. Orgerie, I. Rodero, M. Parashar, and J. M. Menaud. Leveraging renewable energy in edge clouds for data stream analysis in iot. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 186–195, May 2017. doi: 10.1109/CCGRID.2017.92.

[84] M. Lin, Z. Liu, A. Wierman, and L. L. H. Andrew. Online algorithms for geographical load balancing. In *2012 Int. Green Computing Conference (IGCC)*, pages 1–10, June 2012. doi: 10.1109/IGCC.2012.6322266.

[85] Zhenhua Liu, Minghong Lin, Adam Wierman, Steven H. Low, and Lachlan L.H. Andrew. Geographical load balancing with renewables. *SIGMETRICS Perform. Eval. Rev.*, 39(3):62–66, December 2011. ISSN 0163-5999. doi: 10.1145/2160803.2160862. URL http://doi.acm.org/10.1145/2160803.2160862.

[86] Zhenhua Liu, Iris Liu, Steven Low, and Adam Wierman. Pricing data center demand response. *SIGMETRICS Perform. Eval. Rev.*, 42(1): 111–123, June 2014. ISSN 0163-5999. doi: 10.1145/2637364.2592004. URL http://doi.acm.org/10.1145/2637364.2592004.

[87] Jose Luis Lucas-Simarro, Rafael Moreno-Vozmediano, Ruben S. Montero, and Ignacio M. Llorente. Scheduling strategies for optimal service deployment across multiple clouds. *Future Generation Computer Systems*, 29 (6):1431 – 1441, 2013. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2012.01.007. URL http://www.sciencedirect.com/science/article/pii/S0167739X12000192. Including Special sections: High Performance Computing in the Cloud and Resource Discovery Mechanisms for P2P Systems.

[88] Liang Luo, Wenjun Wu, W.T. Tsai, Dichen Di, and Fei Zhang. Simulation of power consumption of cloud data centers. *Simulation Modelling Practice and Theory*, 39(0):152 – 171, 2013. ISSN 1569-190X. doi: http://dx.doi.org/10.1016/j.simpat.2013.08.004. URL http://www.sciencedirect.com/science/article/pii/S1569190X13001214. S.I.Energy efficiency in grids and clouds.

[89] Toni Mastelic, Ariel Oleksiak, Holger Claussen, Ivona Brandic, Jean-Marc Pierson, and Athanasios V. Vasilakos. Cloud computing: Survey on energy efficiency. *ACM Comput. Surv.*, 47(2):33:1–33:36, December 2014. ISSN 0360-0300. doi: 10.1145/2656204. URL http://doi.acm.org/10.1145/2656204.

[90] Peter M. Mell and Timothy Grance. Sp 800-145. the nist definition of cloud computing. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2011.

[91] Microsoft. Microsoft azure: Sla for virtual machines, 2018. URL https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_8/.

[92] Microsoft Azure. Microsoft azure, 2019. https://azure.microsoft.com/.

[93] Trevor Mudge. Power: A first-class architectural design constraint. *Computer*, 34:52–58, 2001. ISSN 0018-9162. doi: doi.ieeecomputersociety.org/10.1109/2.917539.

[94] D.B. Nelson, M.H. Nehrir, and C. Wang. Unit sizing and cost analysis of stand-alone hybrid wind/pv/fuel cell power generation systems. *Renewable Energy*, 31(10):1641 – 1656, 2006. ISSN 0960-1481. doi: https://doi.org/10.1016/j.renene.2005.08.031. URL http://www.sciencedirect.com/science/article/pii/S0960148105002600.

[95] Sergio Nesmachnow, Cristian Perfumo, and Íñigo Goiri. Holistic multiobjective planning of datacenters powered by renewable energy. *Cluster Computing*, 18(4):1379–1397, Dec 2015. ISSN 1573-7543. doi: 10.1007/s10586-015-0485-1. URL https://doi.org/10.1007/s10586-015-0485-1.

[96] Cisco Visual Networking. Cisco global cloud index: Forecast and methodology, 2016–2021. *White paper. Cisco Public, San Jose*, 2016.

[97] Anne-Cecile Orgerie, Marcos Dias de Assuncao, and Laurent Lefevre. A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Comput. Surv.*, 46(4):47:1–47:31, March 2014. ISSN 0360-0300. doi: 10.1145/2532637. URL http://doi.acm.org/10.1145/2532637.

[98] D. O'Sullivan. Data centre redundancy levels. Technical report, EATON, March 2016.

[99] Debdeep Paul, Wen-De Zhong, and Sanjay K. Bose. Energy efficiency aware load distribution and electricity cost volatility control for cloud service providers. *Journal of Network and Computer Applications*, 59:185 – 197, 2016. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2015.08.012. URL http://www.sciencedirect.com/science/article/pii/S1084804515001988.

[100] Debdeep Paul, Wen-De Zhong, and Sanjay K Bose. Demand response in data centers through energy-efficient scheduling and simple incentivization. *IEEE Systems Journal*, 11(2):613–624, 2017.

[101] J. Pierson, G. Baudic, S. Caux, B. Celik, G. Da Costa, L. Grange, M. Haddad, J. Lecuivre, J. Nicod, L. Philippe, V. Rehn-Sonigo, R. Roche, G. Rostirolla, A. Sayah, P. Stolf, M. Thi, and C. Varnier. Datazero: Datacenter with zero emission and robust management using renewable energy. *IEEE Access*, pages 1–1, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2930368.

[102] Jean-Marc Pierson, Patricia Stolf, Hongyang Sun, and Henri Casanova. MILP formulations for spatio-temporal thermal-aware scheduling in Cloud and HPC datacenters. *Cluster Computing : The journal of Networks, Software Tools and Applications*, pages 1–19, avril 2019. URL https://doi.org/10.1007/s10586-019-02931-3.

[103] C. Preist and P. Shabajee. Energy use in the media cloud: Behaviour change, or technofix? In *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pages 581–586, Nov 2010. doi: 10.1109/CloudCom.2010.40.

[104] D. Puthal, B.P.S. Sahoo, S. Mishra, and S. Swain. Cloud computing features, issues, and challenges: A big picture. In *Computational Intelligence and Networks (CINE), 2015 International Conference on*, pages 116–123. IEEE, Jan 2015. doi: 10.1109/CINE.2015.31.

[105] Charles Reiss, John Wilkes, and Joseph L. Hellerstein. Google cluster-usage traces: format + schema. Technical report, Google Inc., Mountain View, CA, USA, November 2011. Revised 2014-11-17 for version 2.1. Posted at https://github.com/google/cluster-data.

[106] C. Ren, D. Wang, B. Urgaonkar, and A. Sivasubramaniam. Carbon-aware energy capacity planning for datacenters. In *2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 391–400, Aug 2012.

[107] REN21. Renewables 2017 global status report. Technical report, REN21, January 2017.

[108] RenewIT. Renewit project, 2016. http://www.renewit-project.eu/.

[109] B. Saovapakhiran and M. Devetsikiotis. Enhancing computing power by exploiting underutilized resources in the community cloud. In *2011 IEEE International Conference on Communications (ICC)*, pages 1–6, June 2011. doi: 10.1109/icc.2011.5962544.

[110] M. Satyanarayanan. Pervasive computing: vision and challenges. *Personal Communications, IEEE*, 8(4):10–17, Aug 2001. ISSN 1070-9916. doi: 10.1109/98.943998.

[111] Karen Scarfone, Cyrus Tibbs, Matthew Sexton, et al. Guide to securing wimax wireless communications. *NIST Special Publication*, 800:127, 2010.

[112] Navin Sharma, Sean Barker, David Irwin, and Prashant Shenoy. Blink: Managing server clusters on intermittent power. *SIGARCH Comput. Archit. News*, 39(1):185–198, March 2011. ISSN 0163-5964. doi: 10.1145/1961295.1950389.

[113] Arman Shehabi, Sarah Smith, Dale Sartor, Richard Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, Nathaniel Horner, Inês Azevedo, and William Lintner. United states data center energy usage report. Technical report, Federal Energy Management, 2016.

[114] S. Shen, V. v. Beek, and A. Iosup. Statistical characterization of business-critical workloads hosted in cloud datacenters. In *2015 15th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing*, pages 465–474, May 2015. doi: 10.1109/CCGrid.2015.60.

[115] M. Srinivas and L. M. Patnaik. Genetic algorithms: a survey. *Computer*, 27(6):17–26, June 1994. doi: 10.1109/2.294849.

[116] Hongyang Sun, Patricia Stolf, and Jean-Marc Pierson. Spatio-temporal thermal-aware scheduling for homogeneous high-performance computing datacenters. *Future Generation Computer Systems*, 71:157–170, Feb 2017. URL http://doi.org/10.1016/j.future.2017.02.005-http://oatao.univ-toulouse.fr/18918/.

[117] Andrei Tchernykh, Uwe Schwiegelsohn, Vassil Alexandrov, and El ghazali Talbi. Towards understanding uncertainty in cloud computing resource provisioning. *Procedia Computer Science*, 51:1772

– 1781, 2015. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2015.05.387. URL http://www.sciencedirect.com/science/article/pii/S1877050915011953. International Conference On Computational Science, ICCS 2015.

[118] My Ton and Brian Fortenbury. High performance buildings: Data centers uninterruptible power supplies (ups). *Lawrence Berkeley National Laboratory, Final PIER-CEC Report*, 2005.

[119] Adel Nadjaran Toosi, Chenhao Qu, Marcos Dias de Assuncao, and Rajkumar Buyya. Renewable-aware geographical load balancing of web applications for sustainable data centers. *Journal of Network and Computer Applications*, 83:155 – 168, 2017. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2017.01.036. URL http://www.sciencedirect.com/science/article/pii/S1084804517300590.

[120] Johan Tordsson, Ruben S. Montero, Rafael Moreno-Vozmediano, and Ignacio M. Llorente. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. *Future Generation Computer Systems*, 28(2):358 – 367, 2012. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2011.07.003. URL http://www.sciencedirect.com/science/article/pii/S0167739X11001373.

[121] V. G. Tran, V. Debusschere, and S. Bacha. Hourly server workload forecasting up to 168 hours ahead using seasonal arima model. In *2012 IEEE International Conference on Industrial Technology*, pages 1127–1131, March 2012. doi: 10.1109/ICIT.2012.6210091.

[122] William N Venables and Brian D Ripley. *Modern applied statistics with S-PLUS*. Springer Science & Business Media, 2013.

[123] V. Villebonnet, G. Da Costa, L. Lefevre, J. M. Pierson, and P. Stolf. Energy aware dynamic provisioning for heterogeneous data centers. In *2016 28th Int. Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 206–213, Oct 2016. doi: 10.1109/SBAC-PAD.2016.34.

[124] L. Wang, G. von Laszewski, J. Dayal, and F. Wang. Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with dvfs. In *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 368–377, May 2010. doi: 10.1109/CCGRID.2010.19.

[125] Xiaoying Wang, Zhihui Du, Yinong Chen, and Mengqin Yang. A green-aware virtual machine migration strategy for sustainable datacenter powered by renewable energy. *Simulation Modelling Practice and Theory*, 58:3–14, 2015.

[126] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, January 1991. URL http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html.

[127] S. Yamamoto, S. Matsumoto, and M. Nakamura. Using cloud technologies for large-scale house data in smart city. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 141–148, Dec 2012. doi: 10.1109/CloudCom.2012.6427546.

[128] F. Yang and A. A. Chien. Large-scale and extreme-scale computing with stranded green power: Opportunities and costs. *IEEE Transactions on Parallel and Distributed Systems*, 29(5):1103–1116, May 2018. ISSN 1045-9219. doi: 10.1109/TPDS.2017.2782677.

[129] Liang Zhang, Tao Han, and Nirwan Ansari. Renewable energy-aware inter-datacenter virtual machine migration over elastic optical networks. In *Cloud Computing Technology and Science (CloudCom), 2015 IEEE 7th International Conference on*, pages 440–443. IEEE, 2015.

[130] Yuanxin Zhu, Bridget Carragher, Robert M. Glaeser, D. Fellmann, Chandrajit Bartz Bajaj, Marshall Bern, Fabrice Mouche, Felix de Haas, Richard J. Hall, David J. Kriegman, Steven J. Ludtke, Satya P. Mallick, Pawel A. Penczek, Alan M. Roseman, Fred J. Sigworth, Niels Volkmann, and Clinton S. Potter. Automatic particle selection: results of a comparative study. *Journal of structural biology*, 145 1-2:3–14, 2004.