POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA (DEIB)
DOCTORAL PROGRAMME IN INFORMATION TECHNOLOGY

EFFICIENT ENERGY-AWARE MODELS FOR CLOUD COMPUTING SYSTEMS AND NETWORKS

Doctoral Dissertation of:
Amine Barkat

Supervisor:
Prof. Antonio Capone

Tutor:
Prof. Paolo Giacomazzi

The Chair of the Doctoral Program:
Prof. Andrea Bonarini

Year XXIX Cycle

Abstract

DEVELOPMENTS in communication networks gave the birth to cloud computing which is revolutionizing the use of IT services in companies as electricity did in its time. Once, companies had to produce their own energy to operate before the arrival of electricity networks. Today, companies must manage their IT assets with the constraints related to their consumption and maintenance. With cloud computing, companies can use on demand IT services without worrying about the management, security and maintenance of the infrastructure.

However, the energy consumption of cloud system is not negligible. Indeed, currently the energy consumed by ICT is estimated to be more than 4% of the worldwide consumption and it is expected to double in the next few years. This consumption has effects not only on economies of governments and companies, but also on planet environment through its carbon footprint, which makes energy efficiency of cloud systems one of todays major challenges.

In this thesis we will address cloud energy issues by developing optimization models based on operational research techniques by taking into account the economic perspective of the cloud providers and users. More specifically, we consider two different types of clouds, the first is the classical cloud system in which data centers offer computing services to users. While the second is a cloud system used in mobile networks named Cloud-RAN. For each system, our goal is to wider the vision and consider more than one problem related to energy in a joint way. Obtai-

ned results show that joint optimization for both cases is more efficient in terms of energy consumption and expenses.

Summary

Gli sviluppi tecnologici nelle reti di comunicazione hanno portato alla nascita del cloud computing che sta rivoluzionando l'uso dei servizi IT in aziende come ha fatto l'elettricità a suo tempo. Una volta, le aziende dovevano produrre l'energia per operare prima dell'arrivo delle reti elettriche. Oggi, le aziende devono gestire le loro risorse IT, con i vincoli legati al loro consumo e alla manutenzione. Con il cloud computing, le aziende possono usare i servizi su richiesta, senza doversi preoccupare della gestione, la sicurezza e la manutenzione dell'infrastruttura.

Tuttavia, il consumo delle grandi infrastrutture del cloud computing è significativo. Attualmente l'energia consumata dall'ICT è oltre il 4% del consumo globale e si prevede possa raddoppiare nei prossimi anni. Questo consumo ha effetti non solo sulle economie di governi e aziende, ma anche sull'ambiente del pianeta attraverso il carbon footprint, che rende l'efficienza energetica dei sistemi cloud una delle sfide dei nostri giorni.

In questa tesi si affronta il problema del consumo di energia dei sistemi cloud attraverso lo sviluppo di modelli di ottimizzazione e metodi di ricerca operativa prendendo in considerazione la prospettiva economica degli utenti e fornitori di servizi. In particolare, consideriamo due diversi tipi di sistemi cloud: il primo è il sistema di cloud classico, in cui i data center offrono servizi di computing per gli utenti; il secondo è un sistema di cloud utilizzato in reti di telefonia mobile denominati Cloud-

RAN. Per ciascun sistema, il nostro obiettivo è quello di avere un ampia visione e considera più di un problema legato all'energia in maniera congiunta. I risultati ottenuti indicano come l'ottimizzazione congiunta per entrambi i casi sia più efficiente in termini di costi e consumi energetici.

ACKNOWLEDGMENTS

Feeling are deeper than what words can define. First and foremost, I would like to express my greatest gratitude to my advisor, Prof. Antonio Capone, for his continuous support, enlightening guidance, and for providing me with an excellent atmosphere throughout all my Ph.D. years in Antlab at Polimi.

I am very thankful to all the community of European project Erasmus Mundus GreenIT, which sustained my 3 years PhD research . A special thanks to Rebeca P. Diaz and Maria P. Pardal the GreenIT coordinators from University of Vigo, who gave me the support and facilitated my mobility to Polimi. I will never forget the help that I have received from Marco Simonini and Elena Cortiana from the international students office in Polimi, I really appreciate their support and availability in any time.

I would like to thank people from the international relations office in my home institution, University of Bejaia in Algeria, and special thanks to Karima, who was very careful and was asking about my progress frequently.

I am deeply grateful to my family for their unconditional love and support. To my parents, my first and most influential teachers, who taught me the values of life, always encouraging me to keep on with my studies. To my sister and brother who were always nearby despite the geographical distance. To my grandparents for being an inspiration in my life.

On the way to my Ph.D., I was very lucky to have crossed paths with some great people and brilliant minds who became life friends. Thanks to the international group of Ph.D. students, Soufian, Sonda, Ahmed, Kiana, Aida, and all the others, with whom I have shared the same goal, we have supported each other and we have shared a lot of memories.

A special thanks to SAKAPA Teatro group and all the amazing friends that I have met in its context, Felice, Chiara, Alessandro, Elena, David, Giacomo, Riccardo, Antonella, Valerio and all the others.

Finally, I would like to thank my brothers from different mothers, who have stood next to me through the thick and thin! Abdellah, Mohammed, Amine, Ayoub, Nasro, who were always fully available for any service that I needed even if it took days of their time.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation

In recent years, the wide adoption of Information and Communication Technologies (ICT) and the exponential growth of Internet users have significantly contributed to the increase of the world energy consumption [102, 140], and the impact of the digital economy is expected to increase even more over the next years [74, 140]. Even if ICT is actually helping other sectors of the economy to reduce their environmental impact, the energy consumption of the ICT sector itself cannot be neglected.

In Cloud Computing, data centers are well known for being particularly energy hungry. Electricity consumed by global data centers is estimated to be between 1.1% and 1.5% of total electricity use [100]. Typically, data centers are rather inefficient and consume more energy than required [65], leaving room for improvement achievable through intelligent management techniques.

A very important aspect linked to the increase of power consumption is the increase of its expenses which have a direct impact on the economy. Like carpooling, the principle of cloud computing is to rent only a part of the server instead of renting the whole of it. These fractions of servers

are called *virtual servers*, and it is the responsibility of the providers of this servers to maintain physical infrastructure, and it is up to the software publishers to set up an economical model for their applications. To approve the viability of this economical model it is necessary that the renting cost of virtual servers is the lowest possible.

The cost of a virtual server can be broken down into two parts. The first is related to the investment (construction of the Hall, purchase of the server etc.), and it is called CAPEX (Capital Expenditure). The second is related to the operation and it is called OPEX (Operational Expenditure). The cloud servers OPEX itself is composed of two parts. The first is related to the maintenance of the server (salary of administrators, basic software maintenance etc.) and the second is related to the electric power consumed by the server and the equipment necessary for its operation (air conditioning, switch networks etc.). This part of the OPEX has been growing for several years. Even though the price of servers has declined recently, the price of energy has steadily increased due to strong fluctuations of prices of oil and gas (which are of the primary energy consumed for electricity production) because of the political situation in producing and importing countries. In addition, the global financial crisis and the recession that ensued put investors in the sector of electricity in standby state and change the perspective of the energy markets (energy companies are drilling fewer wells of oil and gas, reduce expenditures at refineries, pipelines and power plants, ongoing projects have been slowed down, deferred or canceled) [118].

Differently than the economic issues, the increase in energy consumption have a significant impact on the environment. The accumulation of CO_2 in the atmosphere continues to rise since the second half of the 18th century, coinciding with the period of industrial revolution. The rate of CO_2 climbed from a normal level of 278 ppm (part per million)[1] in 1850 to 387 ppm in 2009 [33], to reach 409.44 ppm April 2016 [24]. The origin of this increase is the massive use of non-renewable carbon resources (coal, oil, natural gas, shale gas, etc.) for 150 years. Fuel coming from carbonaceous resources represent a large majority of 85% of the energy used today [33]. The remaining 15% mainly includes nuclear energy

[1] A ppm of CO_2 represents a molecule of CO_2 in a million molecules contained in air, so in a percentage of 0.0001%. 278 ppm are 0,0278% or approximately 0.03%.

and a minority of renewable energy, such as solar energy, marine energy, wind energy, or the valorisation of biomass. However, fuels from non-renewable carbon resources continue and will continue to be widely used in the coming decades, which will lead to an even greater increase in CO_2 emissions.

This greenhouse gas inevitably injected in the atmosphere as a waste of the production of energy and its use in transportation and industry, is directly involving in a slow increase in the temperature of the Earth, according to the Group of intergovernmental experts on the evolution of climate: Intergovernmental Panel on Climate Change (IPCC) [16]. This emissions are largely responsible for climatic changes observed for decades, and if nothing is done to decrease this emissions these trends are likely to continue.

The increase in planet temperature is accompanied by a dilation of the water on the planet and therefore an increase in the volume of the seas and oceans. However, stopping emitting would not change the situation overnight. Even if we cut CO_2 emissions now, it would take centuries to find a stable situation, which would not even be equal to what was before the industrial revolution. It is estimated that it would take between 100 and 300 years for the level of CO_2 in the atmosphere to stabilize and a few centuries for the temperature to be balanced. Even if that happens, the increase in sea level due to thermal expansion would continue for several centuries after that, because of the increase in temperature. This scenario is alarming; It is therefore essential to reduce CO_2 emissions and find effective solutions.

Therefore, either for economical or environmental reasons, or just practical sense, it is interesting, and even necessary, to reduce the energy consumption of the cloud servers, while considering to keep a high level of performance, because although systematically shut down half of the servers allows to reduce energy consumption, it is obviously not a viable solution. However, as we will see later, this cannot be done in any way.

1.2 Problem description

After knowing the importance of reducing energy consumption and its expenses in Cloud computing systems, the question is how to reduce servers consumption without too much impact on the quality of service and the performance of applications running on the cloud? In fact, there are many ways to reduce the electricity bills. However, by accepting a small performance degradation, it is possible to further reduce the power consumed by the infrastructure. To do so, different means are available, some are based on physical machines level, or from a wider scope on the infrastructure level, while others are purely based on applications level.

On the physical machine level, the simplest way to reduce power consumption of servers is simply to turn them off. However, this is not straightforward and simple in practice. Thus, in order to shut down a server, first make sure that it is not used at the time, but also that you are not going to need it in the immediate future, to not being left with a lack of resources, and thus not be able to meet the demands of customers. We can however move machines in several states, depending on the selected power level [51, 70]. These different states are actually depending on different sub-components that will stay on, guaranteeing a fast respond to new users' requests. In other words, optimizations on the physical machine are very important to save energy, since the majority of a server consumption comes simply from being turned on.

Knowing this, savings can be achieved on clusters level, by playing with the number of running machines, the more servers are turned off, more related infrastructure can saves energy. This has a direct impact on the cooling infrastructure of data centers. The behavior of the cooling infrastructure is intimately linked to the load on the servers. The more processing is assigned to servers, the more heat is realized, which increase the heat in the room, therefore the cooling infrastructure increases its consumption to reduce this heat. It is possible to use this fact to induce energy savings by intelligent workload management to avoid hot spots.

On the other hand, energy consumption can be reduced at the design stage of the Cloud infrastructure. While choosing data centers location and their network architecture, many energy aspects may be introduced

at the design process, including: the energy price in candidate data center locations, the availably of renewable energy source, air temperature in outside installation sites, etc. Recently, big Cloud services provider such as Google, Facebook and Microsoft, started inaugurating new data centers in the north countries (Sweden, Finland, etc), where the average annual temperature is lower than zero in some areas. Therefore, cold air contribute to cooling the data center. In addition, seawater is used as coolant liquid.

Finally, it is possible to act on energy at the application level. When a cloud manager implements its infrastructure, it is made to meet requests for execution of applications. In doing so, the service provider must, according to the demands of the customers, links its infrastructure, and the tasks to run. In other words, it must do these task allocation on different servers in the infrastructure taking into account the metrics that are important, whether the cost, energy, or performance. This allowance is a purely application decision-making process that will aim to decide where to perform tasks, and also when to perform them and how. A distinction is to be made between these three aspects. Indeed, decision on the location of the execution of a task allow to not overload some machines, thus, avoid hot spots, or allocate more tasks on machines that are more energy efficient. While deciding when to run a task, it is not necessarily to respect the arrival order, but rather to try to optimize the overall execution time of these tasks. Reorganizing the order of tasks may results in a short total execution time, therefore less energy consumption. Finally, the decision on how to execute a task, for example using different server frequencies (DVFS) which allows energy savings [128].

1.3 Thesis Objectives

Most of existing work on energy efficiency in the Cloud focused on managing different components of the system separately. In this thesis we address the problem from a larger scope, in other words, we will tackle the problem of energy efficiency taking into account different components of the cloud systems and try to jointly optimize their functioning considering multi-objectives.

More specifically, we will address multiple problems of energy re-

lated to two different types of Clouds. The first type considered, is a Cloud system providing Platform as a Service (PaaS) to its users, where load balancing between different servers can be used to save energy, however, it may be very expensive in terms of network traffic, as it may lead to congestions and excess of energy consumption in routers if too much workload is exchanged between servers. Therefore, jointly managing servers and their interconnecting network is a challenging task.

The second type of considered Clouds, is called Cloud Radio Access Network (C-RAN), where cloud computing principle is applied in the Radio Access Network of mobile telecommunication systems. While this kind of systems is considered to be energy efficient compared to the traditional RAN, the challenge is to be able to convey two different costs (CAPEX and OPEX) in a single framework, where the aim is jointly find an optimal design for the infrastructure and to minimize its energy consumption.

The purpose of this thesis is, in first part, to present an energy-aware joint management framework for Cloud Computing systems aiming at limiting energy consumption of both data centers and their network while maximizing the use of renewable green energy resources with a strategy based on energy storage technologies. Among other things, the impact of having such a framework when compared with approaches where cloud components are managed separately. This will allow to reduce energy consumption and its related costs by tuning on and off servers, and also the allocation of tasks to the most effective location of data center in terms of energy.

In the second part, our goal is to fill the gap and to develop efficient optimization framework for Cloud Radio Access Networks. The idea to introduce energy management at planning and design stage of the network infrastructure. Decisions regarding Radio planning are usually based on economical and performance constraints, by including energy, we minimize its expenses and we improve the effectiveness of power management.

1.4 Thesis Contributions

In considering the objectives identified in the previous section, we have defined two major contributions:

- ***Joint Framework for management of Cloud Data Centers and Network:***

  We proposed a holistic framework for geo-distributed data centers and their interconnection network. The model is based on virtual machine migration and formulated using mixed integer linear programing (MILP). It can be solved using state-of-the art solvers such as CPLEX in reasonable time. The proposed approach covers various aspects of Cloud computing systems. Alongside, it jointly manages the use of energies coming from renewable and non-renewable resources by using energy storage technologies.

- ***Joint Planning and Energy Management of Cloud Radio Access Networks:***

  We developed a mixed integer linear programming model to jointly plan and manage Cloud Radio Access Networks. In this solution we try to combine different constraints of both problems, and to constitute a trade-off between different objectives of planning and management stages. In other words, we proposed a model that balances the overall cost of the C-RAN between installation costs (CAPEX), and operative costs (OPEX).

1.5 Plan of the Manuscript

After presenting in this chapter the motivation of or work, stated the problem and research objectives, and noted the main contributions of this thesis, the four remaining chapters are structured in a way that reflects the progress of work in this doctorate at the level of contributions. In Chapter 2, we present the context of Cloud Computing and its basic concepts, including different definitions, the concept of visualization and Service Level Agreement (SLA). In Chapter 3, we compare the most famous softwares for building and managing cloud system infrastructure, which we consider an important. In Chapter 4, we present the first contribution, starting by a state of the art of some related works, then we

describe the proposed model and different experiments made for its validation, and we discuss the obtained results. In Chapter 5, we introduce the second contribution and its related works, the obtained results and their discussion. Finally, we dedicate the last chapter for a general conclusion that summaries the work done and the obtained results.

CHAPTER 2

Cloud Computing: Context and Concepts

In this chapter, we present the context in which fits the topic of this thesis. We present some definitions, basic concepts and technologies related to our work. First, we start by trying to find origins of Cloud Computing as well as its definition, and we discuss some of its characteristics and models. Finally, we describe Virtualization technology which present a key aspect in Cloud Computing.

2.1 History

The concept dates back to 1960, when John McCarthy held that the computation may someday be organized as a public utility service [73]. He said at the MIT Centennial in 1961:

> *"If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility ... The computer utility could become the basis of a new and important industry."*

It is difficult to tell the date of birth of Cloud computing, since it is an accumulation of technologies over many years: mainframes in 1970,

client-server in 1980, the web in 1990, Service-oriented architecture SOA in 2000. But we can consider the deployment of Amazon to its demo version of Elastic Computing Cloud (EC2) in 2006 as a birth date for nowadays Cloud Computing.

The notion of Cloud refers to the classic allegory of the Internet, often represented as sky's Cloud. The metaphor of the Cloud is actually a network of computer resources (i.e. hardware and software) virtualized and mutualized, remotely accessible on demand and self-served via a network through Internet technologies [34, 48, 141, 153].

2.2 Definitions

The growth of services across the Web as well as the needs for dealing with increasingly complex data have led to the success of the paradigm of Cloud Computing where providers expose a collection of computing resources (computing, storage, networking,...) through the Internet, allowing customers to use them remotely. These computer resources are charged on the mode of *Pay As You Go*, which means that the user pays for the provided resources only for the its using period, and this is the same mode of billing for the other utilities (electricity, water, gas,...).

This model could be based on outsourcing to a third party. Where a service provider build its services on an infrastructure offered by another service provider. So it can limit the costs of acquisition and maintenance of hardwares and softwares, and also it outsources a part of the risks associated with the infrastructure to another party. Netflix is an example, its users watch videos that are distributed by taking advantage of Amazon's infrastructure. Although Amazon and Netflix are two actors in Cloud Computing, they provide services of different kinds.

While Cloud computing and grid computing are very close paradigms, they differ on some points. First, usually grid computing is an aggregation of resources from several organizations in order to provide a bigger common infrastructure, while the idea behind Cloud computing model tends to be based more on one provider that uses its infrastructure to offer services for different users, except in the case of cloud federation, however, it is not an easy to distinguish between this two paradigms, especially because they are based on similar technologies. In our opin-

ion, the main big difference, is in the pricing models adopted by both paradigms. As mentioned before, Cloud computing is based on *Pay As You Go* model, in which users pay exactly what do they use, while grid computing is based on a renting model, in which users allocate a set of resources (computing, storage,...) for a period of time, where they pay for the renting even if they don't use this allocated resources, which is not the case in Cloud computing.

There are attempts to define the term Cloud Computing (CC) and there is little consensus on a single, universal definition. This multitude of definitions reflects the diversity and technological wealth of Cloud Computing technology. In the following, we cite some of the most relevant definitions. In [141], the authors propose a definition of Cloud computing in which they try to aggregate many elements to satisfy the sensibility of Cloud Computing:

> *"Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs."*

Another interesting definition is proposed by Buyya *et al.* 2009 [48]:

> *"A Cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers."*

However, the definition proposed by the National Institute of Standards and Technology (NIST) in [108], has become almost the reference and commonly accepted by the public.

> *"cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."*

In the same document, NIST define five basic characteristics of Cloud Computing, which we will present in the following section.

2.3 Characteristics

Generally, a service, a solution or a runtime environment should satisfy a list of characteristics to be considered as Cloud Computing. Among these features, there are those that are recognized as fundamental [108].

- ***On-demand self-service :*** A user can unilaterally allocate needed computing resources (servers, network, storage, execution environment, application), automatically and without need for human interaction.
- ***Broad network access:*** Cloud Computing resources are available through the network through a standard mechanisms that encourage their use from heterogeneous client devices (e.g., computers laptops, phones, tablets).
- ***Resource pooling :*** Computing resources are pooled to serve several customers using a multi-tenant model. These resources, physical or virtual, are allocated and released dynamically according to the request of the consumer. Generally, the user has no control or knowledge on the exact location of the resources allocated. In some cases, it can choose the geographic location at a high level (e.g. by country, continent, or data center).
- ***Rapid elasticity :*** Resources are allocated and released in an elastic way, ideally in an automatically way, to quickly adapt to the request whether it is increasing or decreasing. For the consumer, the allocation resources appear as unlimited and can be allocated at any time.

- ***Measured service :*** All resources can be monitored and controlled in order to measure their consumption with an appropriate level of abstraction depending on the type of service (e.g., storage, computation time, and bandwidth).

2.4 Service Models

In order to classify different clouds according to the type of service they offer, we can find three main models based on the level of management offered, starting from the lowest level which is the physical level (hardware) to the highest level (Applications).

- ***Infrastructure as a Service (IaaS):*** This lowest layer includes services that provide computing resources in the form of Virtual machines (VMs) to users. These virtual machines are hosted on physical servers located in data centers belonging to the IaaS provider. The use of visualization technique allows to consolidate the use of physical servers usage, by allowing more than one virtual machine to be on the same physical server, which improve the exploitation of the available resources. In addition, visualization techniques can prevent the interference between virtual machines owned by different users. However, we will provide more details about visualization technique in the following sections.

  IaaS providers are responsible for provision and hosting images of virtual machines. It is up to them as well the management of additional storage resources and functionalities related to the communication inter-VMs like virtual networks. They guarantee as well the quality of service (QoS). Amazon EC2 [6] is an example of IaaS.

- ***Platform as a Service (PaaS):***

  PaaS providers provide software platforms to the developers of service, which are delivered with dedicated programming interfaces (API) that can host and run Cloud Computing applications. By using the services of the PaaS, the work of application designers is simplified, because much of the complexity related to the infrastructure is hidden behind the API offered by the PaaS provider. In this way, the process of developing a software service is simplified,

a designer who uses a PaaS platform has only to provide the source code to the PaaS provider, who in turn cares to deploy it and make it accessible to end-users.

Although the PaaS model is able to reduce the complexity of developing Cloud applications, the source code has to respect the technological constraints imposed by the provider of PaaS, such as the support of a limited number of programming languages, the obligation to use some libraries or some databases servers. However, these constraints make easier the integration of the source code in the technology used in the infrastructure. An example of the PaaS services are: App Engine [4] offered by Google and Amazon Web services [5] offered by Amazon.

- ***Software as a Service (SaaS):***

  It is the highest layer of Cloud Services. It makes the service accessible to the end-users using thin client (Web Browser, Mobile application ...). This services are accessible from distance, which means that the biggest part of the code is executed on the infrastructure of the SaaS service provider or a third party.

  The benefits of this service model are many, such as the fact that the complexity of the deployment of a service as well as its maintenance are hidden to the end user. Hardware requirements are also lower, as the most of the service execution is on the infrastructure of the service Provider. The end-user has a warranty of quality of service (Service Level Agreement) without having to invest in specialized and high-performance hardware. Any breach to this agreement make the provider pay for the user. As an example, we can find many famous services that became a part of our daily life, for example Gmail mailing service [15] by Google and Facebook social network [9].

2.5 Deployment Models

When we talk about the deployment model, we mean by that the manner the Cloud infrastructure is organized and managed. According to NIST, there are four main deployment models [108]:

- ***Private Cloud:***

  The private Cloud is operated only by one single organization. Its infrastructure could be hosted by the organization itself or by a third party. In other words, a private Cloud offers services for a limited set of users, using a firewall to guaranty better control and management.

- ***Public Cloud:***

  Unlike the Private Cloud, Public clouds are made available to general public. It is a model where cloud services are provided to everyone and it can be free to subscribe for their use.

- ***Community Cloud:***

  The community cloud is a model that is shared between several organization in order to meet specific needs (for example: collaborative mission, security, political). The community cloud is designed to meet the requirements of a community or other types of individual companies.

- ***Hybrid Cloud:***

  As the name indicates, it is a combination of two or more types of Clouds. For example, an organization can create its own private Cloud, then it exploits some public cloud services.

2.6 Virtualization

After knowing different models for cloud computing, in this section we address a very important concept in Cloud computing which is the Virtualization. [39, 135]

It is a technique that allows to stimulate the behavior of a physical machine through a software. Virtual machines emerged for the first time in 1960, as a program that can stimulate a full computer which hosts an operating system and which have the same features as a physical computer (CPU, storage, network ...). A virtual machine is a computer program hosted on a physical machine (on the hardware, as it is possible to run a virtual machine in another virtual machine).

One of the main advantages of virtual machines is that they provide an abstraction of the hardware, which allows the execution of services on heterogeneous hardware. In addition, they provide isolation between processes running within multiple VMs, which is a basis for security properties between services belonging to different customers of the same infrastructure provider. Finally, the fact that multiple virtual machines can be performed within the same physical machine opens the way to consolidation mechanisms to increase the usage rate of physical machines.

Virtual machines are the most popular virtualization technique within the IaaS systems, they require the installation of a complete operating system by each virtual machine. An alternative light technique is called *Containers* which use became popular recently. Containers take advantage of mechanisms of isolation between processes (using chroot) of modern operating systems to give the impression that services are running on different operating systems, whereas they are in fact run in different environments, but on the same system belonging to the host physical server. Thanks to the success of the project Docker [12], containers have become much more known, and are starting to be taken into account by IaaS managers which were until then used by PaaS providers.

CHAPTER 3

Open Source Solutions for Building IaaS Clouds

Open source cloud platforms were born as a response to the necessity of Infrastructure as a Service (IaaS) solutions to provide privacy and control over the virtualized environments. Therefore the open source cloud platforms were primly used to build private clouds. Eventually, these open source solutions can be used to set up public clouds, private clouds or a mix of them, i.e. hybrid clouds. With the emergence of different open source cloud solutions, the decision to choose the most suitable one becomes a confusing task, given the specific characteristics of each platform [2]. Moreover, since hybrid clouds are the most widely used nowadays, surveying open source middlewares that simplify cluster management is an important matter. In this chapter we present the general features of OpenStack, CloudStack, OpenNebula, Eucalyptus and Nimbus and compare their general features and important properties.

3.1 OpenStack

OpenStack is a cloud software that offers capability to control large pools of compute, storage and networking resources. It also empowers

users providing on-demand resources [2]. Starting from 2010, OpenStack was developed by Rackspace Hosting and NASA aimed to provide open source cloud solution to build public or private clouds. The mission of OpenStack is to enable any organization to create and offer cloud computing services running on standard hardwares. Provisioned as open source solution, OpenStack is built keeping these core principles in mind: (1) Open source: all code will be released under the Apache 2.0 license allowing the community to use it freely; (2) Open design: every 6 months the development community will hold a design summit to gather requirements and write specifications for the upcoming releases; (3) Open development: maintains a publicly available source code repository through the entire development process; (4) Open community: produces a healthy, vibrant development and user community through an open and transparent process.

3.1.1 General Architecture

As in any cloud platform, the infrastructure underneath OpenStack is standard hardware, which can contain any pieces of physical devices such as servers, disks or network devices. In order to provide cloud services, OpenStack develops virtualization layers giving the abstract view of physical infrastructure to end users. The OpenStack architecture consists of three main components: Compute (Nova), Network (Quantum) and Storage (Swift). Beside these three pillars, OpenStack has been developing many other services, each of those designed to work together to provide a complete IaaS solution. The integration of these services is facilitated through public application programming interfaces (APIs) offered by each service [17].

In the following, the detailed description of each component is provided.

A) ***Compute (Nova):***

Compute is the heart of OpenStack (codename is Nova and it is written in Python), which is the computing fabric controller responsible for managing large networks of virtual machines (VMs), and eventually to properly schedule VMs among available physical machines (PMs) [17]. Compute is a distributed application

that consists of six components: Nova-api, Message Queue, Nova-Compute, Nova-Network, Nova-Volume and Nova-SchedulerNova supports the complete life-cycles of an instance in the cloud, starting from the request to initialize a VM until its termination. It follows this architecture in which we took definitions of its component from [11]:

- Nova-api: accepts and responds to end user compute API calls. Besides providing its own OpenStack Compute API, Nova-api is compatible with Amazon EC2 API, offering the potential to integrate with Amazon cloud services. It has another special Admin API reserved for privileged users to perform administrative actions. The orchestration activities such as running an instance, or enforcing the policies such as quota checks are initiated by this component.
- Nova-compute: is primarily a worker daemon that creates and terminates VM instances via hypervisor APIs. In order to do so, it accepts actions from the queue and performs system commands to fulfill them, while updating the database state accordingly. OpenStack supports several standard hypervisors (listed in Section 3.6) while keeping the openness that allows to interface other hypervisors through its standard library.
- Nova-volume: manages the creation, attaching and detaching of persistent volumes to compute instances. There are two types of block devices supported for a VM instance: (1) Ephemeral Storage: is associated to a single unique instance. Its life-cycle exists together with the instance life-cycle, which means when the instance is terminated, data on this storage will also be deleted; (2) Volume Storage: is persistent and independent from any particular instance. This storage can be used as external disk device where the data stored on it still remain even when the instance is terminated.
- Nova-network: is a worker daemon that handles network-related tasks. It accepts and performs networking tasks from the queue to manipulate the network such as setting up bridging interfaces or changing iptable rules.

- Nova-schedule: handles the scheduling of VMs among PMs. It takes a virtual machine instance request from the queue and determines the physical host it should place the instance on. While the scheduling algorithms can be defined by users, Nova-schedule supports by default three algorithms: (1) Simple: attempts to find least loaded host, (2) Chance: chooses random available host from service table, (3) Zone: picks random host from within an available zone. By allowing users to define their own scheduling algorithms, this component is important for building fault tolerant and load-balanced system.
- Queue: provides a central hub for passing messages between daemons. This is usually implemented withÂ RabbitMQ message broker, but it supports any AMQP message queue.
- Database: stores most of the build-time and run-time state of a cloud infrastructure. For example, it provides information of the instances that are available for use or in use, networks availability or storage information. Theoretically, OpenStack Nova can support any SQL-based database but the most widely used databases currently are sqlite3, MySQL and PostgreSQL.

Given this architecture, all its components follow a shared-nothing and messaging-based policy. Shared-nothing means that each component or each group of components can be installed on any server, in a distributed manner; while the messaging-based policy ensures the communication among all components such as volume, network and scheduler is performed via Queue Server.

B) ***Network (Quantum):***

Network is the key of cloud computing for several reasons: (1) Offered resources and services must be accessible; (2) Address binding between different services is essential to support multi-tier applications; (3) Automatic network configuration capability is important, especially in scenarios where auto-scaling installations evolves. The OpenStack Networking component gives operators the ability to leverage different network technologies to power their cloud networking through a rich set of APIs, multiple networking models

(e.g. flat or private network) and flexible plug-in architecture. Especially, the plug-in architecture - with the plug-in agent - enables, not only capability of using various network technologies, but also the ability to handle user workloads. It means, at network level, that developers can implement their own load balancing algorithms and plug it in the platform to achieve better workload control.

The Network architecture consists of four distinct physical data center networks:

- Management network: used for internal communication between OpenStack components. The IP addresses on this network should be reachable only within the data center.
- Data network: used for VM data communication within the cloud deployment. The IP addressing requirements of this network depend on the OpenStack Networking plug-in in use.
- External network: used to provide VMs with Internet access in the deployment scenarios. The IP addresses on this network should be reachable by anyone on the Internet.
- API network: exposes all OpenStack APIs, including the OpenStack Networking API, to tenants. The IP addresses on this network should be reachable by anyone on the Internet.

C) ***Storage:***

The Storage component, one of three main pillars of OpenStack architecture, is used to manage storage resources. OpenStack has support for both Object Storage and Block Storage, with many deployment options for each, depending on the use case.

Object Storage (codename Swift) is a scalable object storage system. It provides a fully distributed, API-accessible storage platform that can be integrated directly into applications or used for backup, archiving, and data retention [17]. In Object Storage, data are written to multiple hardware devices, with the OpenStack software responsible for ensuring data replication and integrity across clusters. Object storage clusters are scaled horizontally while adding new nodes. If a node fails, OpenStack replicates its content from other active nodes. Because OpenStack uses software logic to ensure data

replication and distribution, inexpensive commodity hard drives and servers can be used instead of expensive equipments. Therefore, Object Storage is ideal for cost effective, scale-out storage [17].

Block Storage (codename Cinder), is the storage system that allows block devices to be exposed and connected to compute instances for expanded storage, better performance and integration with enterprise storage platforms, such as NetApp, Nexenta and SolidFire [17]. By managing the storage resources in blocks, Block Storage is appropriate for performance sensitive scenarios such as database storage, expandable file systems, or providing a server with access to raw block level storage.

D) ***User interface - Dashboard:*** The OpenStack dashboard provides to administrators and users a graphical interface to control their compute, storage and networking resources. Through the dashboard, administrators can also manage users and set limits on resources access for each user.

E) ***Shared Services:***

OpenStack Shared services are a set of several services that span across three pillars of compute, storage and networking, making it easy to perform cloud management operations. These services include the identity, image, telemetry, orchestration and database services [17]:

Identity Service (code-named Keystone): is the security service to protect resources access and usage. This service provides a central directory management, mapping users to OpenStack accessible services. It acts as a common authentication system across the cloud operating system. It supports multiple forms of authentication including standard username and password credentials, token-based systems and AWS-style logins. *Image Service (code-named Glance)*: is the repository for virtual disk and server images used by the VMs. In OpenStack, user can copy or snapshot a server image and immediately store it away. Stored images can be used as a template to get new servers up and running quickly and consistently. *Telemetry Service*: aggregates resources usage and per-

formance data of the services deployed in OpenStack cloud. This powerful capability provides visibility into the usage of the cloud infrastructure and allows cloud operators to view metrics globally or individually. *Orchestration Service*: is a template-driven engine that allows application developers to describe and automate the deployment of the cloud infrastructure as well as detailed post-deployment activities of infrastructure, services and applications. *Database Service*: allows users to quickly and easily utilize the features of a relational database. Cloud users and database administrators can provision and manage multiple database instances as needed.

3.1.2 Properties

Provisioned as IaaS, OpenStack is built following an open philosophy: avoid technology lock-ins by not requiring specific technologies and providing user freedom to choose the best slot that matches its needs [17]. In this section, we will analyze some important properties of OpenStack.

- Live migration: OpenStack supports two types of live migration: (i) Shared storage based live migration, and (ii) Block live migration. The former supports live migration scenarios where the source and destination hypervisors have access to the shared storage, while the latter does not require shared storage.

- Load balancing: OpenStack supports load balancing at different scales. First of all, the supporting feature of live migration has enabled system administrators to distribute application workloads among physical servers by means of adjusting VM placement. Moreover, it is possible to control application workloads at VM level, service provided by OpenStack Network layer, controlled by Network component. This component, with a flexible plug-in architecture allows the development of run-time custom algorithms to distribute workloads among VMs. Indeed, OpenStack has an on-going project called Load Balancing as a Service (LBaaS) that is aimed to provide load balancing service to end users. This service has monitoring feature to determine whether the VMs are available to handle user requests and take routing decisions accordingly. Several rout-

ing policies are supported such as round robin (i.e. rotates requests evenly between multiple instances), source IP (i.e. requests from a unique source IP address are consistently directed to the same instance) and least connections (i.e. allocate requests to the instance with the least number of active connections).

- Fault tolerance: Within the flexible architecture of OpenStack, fault tolerance can be handled at different levels. These levels depend on the way the IaaS system is configured and deployed. At the VMs level, in order to prevent failures, users can develop scheduling algorithms (besides the three already supported algorithms by OpenStack) for placing the VMs that best fits to his use cases. Some scheduling algorithms have been designed at the present time, such as: group scheduling (i.e. VMs that provide the same functionalities are grouped and placed to separate PMs) and rescheduling (i.e. rescheduling of VMs from failed host to surviving hosts using live-migration). At storage or database level, fault tolerance is achieved by using replication and synchronization to ensure that a failure occurred at one device will not break the whole system.

- Availability: In OpenStack, high availability can be achieved through different setups depending on types of services, i.e. stateless or stateful services. Stateless services can provide answer to a request without requiring further information of other services or historical data. OpenStack stateless services include nova-api, nova-scheduler, etc. For these services, high availability is achieved by providing redundant instances and load balance them. In the opposite, stateful services are ones that requires other information to answer a request, which makes them difficult to obtain high availability. These services, e.g. database or storage, can be highly available by using replication but at the same time the system has to maintain the synchronization between the main version and replicated versions in order to keep the system consistent [1].

- Security: OpenStack has a separated service (Identity service) which provides a central authentication management across the cloud operating system and users. The possibility to set up VPNs and firewalls is also available.

- Compatibility: OpenStack is highly compatible with Amazon EC2 and Amazon S3 and thus client applications written for Amazon Web Services can be used with OpenStack with minimal porting effort [17]. In terms of hypervisors, OpenStack supports multiple hypervisors, e.g. Xen, KVM, HyperV, VMWare, etc. Other hypervisors with existing standard drivers can also be interfaced with OpenStack through standard library, e.g. libvirt library.

3.2 CloudStack

CloudStack [8] is an open source software platform, written in Java, designed for development and management of cloud Infrastructure as a Service. It aggregates computing resources for building private, public or hybrid clouds. CloudStack is a turnkey technology that brings together the "Stack" of features requested by companies and users, like data centers orchestration, management and administration of users and NaaS (Network as a Service).

The start of CloudStack was with Cloud.com in 2008. In May 2010, it was open source under GNU General Public License. Citrix bought CloudStack in July 2011, then in April 2012, Citrix donated CloudStack to Apache Software Foundation (ASF) where it was relicensed under Apache 2.0 and accepted as an incubation project. Since March 2013, CloudStack became a Top Level Project of Apache. Many companies are basing on CloudStack for building and managing their cloud infrastructures. Among these companies, there are: Nokia, Orange, Apple, Disney and many others.

3.2.1 CloudStack Architecture

In this section, we break down the logical architecture of CloudStack. In CloudStack, physical resources are organized and managed in a hierarchical structure. The lowest level contains computational devices such as host and primary storage. Hosts are attached together and access shared storage to form a Cluster. The next level consists of clusters which are combined by a layer 2 switch to form a Pod. Go up to higher level, Pods are grouped together with Secondary storage by layer 3 switch to form a Zone. At the highest level, zones are grouped to create a Region. All

these resources are managed by a Management Server. In the following, we describe in details each component that forms the whole architecture.

A) ***Host*** A host represents a physical computational machine that contains local storage. The physical hosts are virtualized by hypervisors. CloudStack supports many hypervisors for VMs management such as Xen, KVM, vSphere, Hyper-V, VMWare, etc. as well as bare metal provisioning. All hosts within a cluster must be homogeneous in terms of the hypervisor, with the possibility of having heterogeneous hypervisors in different clusters.

B) ***Cluster*** Within a cluster, hosts are tied together into the same computational pool with the primary storage and have the same IP subnet. The primary storage can be any kind of storage supported by the hypervisor. One cluster can have more than one primary storage device.

C) ***Pod*** A Pod is a collection of different clusters linked with a layer 2 switch. Hosts in the same Pod are in the same subnet. Pod is not visible to the end user.

D) ***Zone*** The benefit of using Zone is for isolation and redundancy. Often, it corresponds to a data center; although if a data center is large enough, it can have multiple zones. A zone contains Pods that are attached to the secondary storage using a layer 3 switch. Zones are visible to the end user and they can be private or public. Public zones are visible to all users in the cloud while private zones are visible only to users from a particular domain.

E) ***Region*** A region is the largest organizational unit in CloudStack. A region contains multiple zones distributed in geographic locations close to each other.

F) ***Management Server*** A Management Server is used to manage all resources in cloud infrastructure through APIs or UI. One management server can support around 10K hosts and can be deployed on a physical server or a VM. In case we have more than one management server, user interaction to either of them will return the same result. This ensures high availability of CloudStack. A database is

required for management servers to be persistent. In order to prevent single point of failure, we can have one primary database and several database replica which always stay synchronized with the primary copy.

G) ***Storage***

In addition to the host local storage, CloudStack manages two main types of storage: primary storage and secondary storage.

- Primary storage: is a storage associated with a cluster or a zone. In the same cluster, we can deploy multiple primary storages. This kind of storage is basically used to run VMs and stores application data. Since this storage interacts directly with applications deployed in VMs, it can be expensive in terms of I/O operations, this is the reason why it is placed physically near to the hosts.
- Secondary storage: is used to store ISO images, templates, snapshots, etc. It supports two different types, NFS and Object Storage.
  - ISO image: is used when user wants to create a VM.
  - Template: is the base operating system image that the user can choose when creating new instance. It may also include additional configuration information such as installed applications.
  - Snapshot: is used as backup for data recovery service. CloudStack supports two types of snapshot: individual snapshot and recurring snapshot. The former is one-time full snapshot, while the latter is either one-time full snapshot or incremental snapshot.

H) ***Networking*** CloudStack supports the use of different physical networking devices (e.g. NetScaler, F5 BIG-IP, Juniper SRX, etc). In CloudStack, users have the ability to choose between two types of networks scenarios: basic and advanced. The basic scenario is for an AWS-style networking. It provides a single network where guest isolation is done through the layer 3 switch. The advanced scenario is more flexible in defining guest networks [8]. For example, the

administrator can create multiple networks for use by the guests. CloudStack provides many networking services. Among them we cite:

- Isolation: CloudStack assures the isolation of networks, by allowing the access to the isolated network only by virtual machines of a single account.
- Load Balancing: to balance the traffic in the cloud, the user can create a rule to control and distribute the traffic and apply it to a group of VMs. Within the defined rule, user can choose a load balancing algorithm among the supported ones.
- VPN: for accessing to the VM using CloudStack account, users can create and configure VPNs. Each network has its own virtual router, so VPNs are not shared across different networks. Using VPN tunnels, hosts in different zones are allowed to access each other.
- Firewall: hosts in the same zone can access to each other without passing through the firewall. Users can use external firewalls.

3.2.2 Properties

Cloudstack has many properties that motivate companies to use it to manage their infrastructure. The main properties of CloudStack are [8]:

- Live migration: A live migration of running VMs between hosts is allowed in CloudStack through the Dashboard. Depending on the VM's hypervisor, migration conditions can be different. For example, live migration using KVM hypervisor will not support the use of local disk storage, and source and destination hosts have to be in the same cluster; while Xen and VMWare support local disk storage and allow to migrate between different clusters [7].
- Load balancing: a Load balancer is an optional component of CloudStack that allows to distribute the traffic among different

management servers [126]. In addition to creating rules and using load balancing algorithms, CloudStack offers the possibility to integrate with external load balancers such as Citrix NetScaler [126].

- Fault tolerance: in CloudStack, fault tolerance is achieved at different scales. In order to prevent failures of management server, the server can be deployed in multi-node configuration. Should one management node fail, other nodes can be used without affecting cloud functioning. Failures at database level are handled by using one or more replication of the database linked to the management server. For host's fail-over, CloudStack recovers the VM instances by taking the images from secondary storage and using application data in primary storage.
- Availability: CloudStack ensures high availability of the system by using multiple management server nodes which may be deployed with load balancers.
- Security: in addition to isolation using different accounts, VPNs and firewalls, CloudStack offers the isolation of traffic using the strategy of security groups which are sets of VMs that filter the traffic on the basis of configuration rules. CloudStack provides a default security group with predefined rules, however they can be modified if necessary.
- Compatibility: CloudStack is built based on a pluggable architecture, one cloud can support different hypervisor implementations including: Hyper-V, KVM, LXC, vSphere, Xenserver, Xen Project and also bare metal provisioning. Moreover, CloudStack is compatible with Amazon API and enables the integration of these two platforms.
- Scalability: CloudStack has the ability to manage thousands of servers distributed in different data centers and different locations thanks to the management server capability (one management server node can manage a big pool of physical resources), and the possibility of using multiple management servers for reducing VMs downtime.

- API extensibility: The CloudStack APIs are very powerful and allows developers to create new command line tools and UIs, and to plug them into CloudStack architecture. If the developer wants to use new hypervisor, new storage system or new networking service, he just needs to write a new plug-in in Java and integrate it.

3.3 Eucalyptus

Eucalyptus is a popular open source IaaS product, provided by Eucalyptus Systems [13]. It is used to implement, manage, and maintain private and hybrid clouds (but cannot build public clouds) with a main key design feature which is Amazon Web Services (AWS) API compatibility. Many programming language can be used to code Eucalyptus including: Java, C, Groovy, Shell, Perl, Python [13].

Originally it was designed at the University of California, Santa Barbara, as set of services that could emulate AWS on a different site apart from Amazon servers, with the aim of linking together AWS, supercomputer centers at National Science Foundation and several university sites [111]. It became a for profit organization in 2009. In 2012, Eucalyptus started a partnership with AWS that allowed to develop more AWS-compatible environments, and to create hybrid clouds by facilitating movements of instances -created from stored Operating System Images- between Eucalyptus private cloud and Amazon Elastic Compute Cloud (EC2). In September 2014 HP acquired Eucalyptus and lunch it under the Helion Eucalyptus name.

Beside its high AWS compatibility that allow running an application on AWS and Eucalyptus without any modifications, Eucalyptus is characterized with its high availability configuration, easy installation and simple user interface. Its main clients include: AppDynamics, Nokia, NASA, Puma and others [13].

3.3.1 Design and Architecture

Eucalyptus have a highly modular, hierarchical and distributed architecture [117]. Users familiar with AWS do not find any difficulties with Eucalyptus because it replicates the same interaction tools and interfaces

used in AWS, such as: euca2ools - the command-line tool - or the Eucalyptus User Console - a GUI based tool. Eucalyptus architecture is characterized by five main components: Cloud Controller, Cluster Controller, Storage Controller, Node Controller and Scalable Object Storage, and one optional component: VMware Broker. This components are grouped in three different logical levels: Cloud Level, Cluster Level and Node Level [13]. In the following sections we describe each logical level and the associated components.

A) ***Cloud Level*** Contain tow components: The Cloud Controller and the Scalable Object Storage.

The *Cloud Controller* (CLC) is a Java program that provides the interface to the cloud, and each cloud contain only one. CLC contain query interfaces and a EC2-compatible SOAP. It handles users requests and provides high level authentication, quota management, accounting and reporting. CLC does also meta-schedule and manages different cloud resources after collecting information provided by the Cluster Controller.

The *Scalable Object Storage* (SOS) is a Eucalyptus service that allows the use of external (open source or commercial) storage solutions. SOS is equivalent to AWS Simple Storage Service (S3). In case of small deployments Eucalyptus has a basic storage implementation called Walrus that have two main functionalities: (i) storage of system files that could be accessible from different nodes (it may contains: VM images, volumes, snapshots, Linux kernel images, Root filesystem), and (ii) be used as Storage as a Service to store users data and applications.

B) ***Cluster Level*** Each cluster is formed by a group of nodes linked with a LAN network. This level contain two main components: Cluster Controller and Storage Controller, and one optional: VMware Broker. Clusters are under subnets with a specific range of IP addresses, what compromises the flexibility and is a disadvantage of Eucalyptus.

The *Cluster Controller* (CC) is a C program that collects information, manages the execution of the virtual instances and virtual network and verifies the respect of Service Level Agreement. Multiple

Cluster Controller could exist within one cloud. It works as an intermediate between the Cloud Controller, the Storage Controller and the Node Controller. CC is equivalent to AWS availability zone.

The *Storage Controller* (SC) is a Java program developed to have the same features as AWS Elastic Block Store (EBS). It manages the snapshots and volumes of a specific Cluster, it also controls the block-access network storage, and communicates with different storage systems (NFS, iSCSI, SAN ...), with the Node Controller and the Cluster Controller.

The *VMware Broker* (VB) is an optional component which is available only in Eucalyptus version with VMware support. It provides an AWS compatible interface for VMware environments that allows the deployment of VMs on VMware infrastructure elements. VB manage directly the communication between the CC and the VMware hypervisors (ESX/ESXi), or it is possible that it passes through VMware vCenter.

C) ***Node Level*** In this level we find only one component which is Node Controller, a C program that hosts VMs and their associated services, and manages the endpoint of the virtual network. NC interacts with the hypervisor and the hosted operating system to control VMs life-cycled, their creation and termination. It collects and sends information about VMs and their associated physical resources to the Cluster Controller to make high level decisions, like when to proceed with the load balancing.

3.3.2 Properties

- Compatibility: it is the main feature of Eucalyptus. AWS API are built on top of Eucalyptus tools which simplifies intercommunication between both [101]. AWS API supported by Eucalyptus are: Elastic Compute Cloud (EC2), Elastic Block Storage (EBS), Amazon Machine Image (AMI), Simple Storage Service (S3), Identity and Access Management (IAM), Auto Scaling, Elastic Load Balancing, CloudWatch, CloudFormation.

- Live migration: one of the weak points of Eucalyptus is the absence

of a live migration feature. Nevertheless many external approaches was proposed to add this feature like the one described in [97].

- Load balancing and Fault tolerance: Eucalyptus does not contain an implicit load balancing mechanism for the low level of VMs nor for the high level of user requests, but it could be achieved using Elastic Load Balancing (ELB) of AWS. Elastic Load Balancing is a service that provides a great application fault tolerance by distributing the incoming service requests and users traffic among different Eucalyptus instances. It automatically detects overloaded instances, and redirects the traffic to more available ones. Load balancers are the core of Elastic Load Balancing, they are special Eucalyptus instances created from a specific Eucalyptus VMs images. If there is a problem in one of the instances or if it is removed due an internal error, the system stop rerouting traffic to that instance until it is restored or a new one is created. Load balancing could be managed within the same cluster or among different clusters depending on the need. Elastic Load Balancing contains a health check system that routinely send check requests to instances. It uses latency, RequestCount and HTTP response code counts to verify if the instance is responding in time to users requests.

- Scalability: as we mentioned before, Eucalyptus is by nature distributed, what make it highly scalable. There is also an auto scaling feature that allow to add and remove instances and VMs depending on traffic increase, the available resources and with respect to the Service Level Agreement. Using this feature developers can scale resource up or down depending on the need. There are three main component in Auto Scaling which are:

  - Auto Scaling group: is the main component of Auto Scaling, it defines for each user the minimum and the maximum number of scaling instances, and the related parameters. In case the user did not select any specifications, it chooses the default parameters, which are equal to the minimum number of instances to use.
  - Launch configuration: it contains the information needed to

make the scaling, including instance type, VM image ID, security groups, and many others.

  - Scaling plan (policy): it defines the manner of doing the auto scaling. It could be manually or automatically, responding to CloudWatch alarms.

- Cloud Watch: is a Eucalyptus service that collect raw data from different cloud resources (instances, elastic block store volumes, auto scaling instances and load balancers) and generate and record performance metrics. This allows users to make operational business decisions based on the historical records. Cloud Watch also configures alarms based on data from user filled metrics.

- Availability: Eucalyptus contains high availability as a feature since version 3. If an individual node or even a rack fails, Eucalyptus put other nodes into use immediately or moves to another rack in case of rack failure. This is achieved through a service that is running concurrently on physical machines which is "hot spare". The information failure is quickly diffused internally, without any signs to the users, and the failure is recovered with respect to SLA (service level agreement). In [136] an evaluation tool was proposed for testing availability in IaaS platforms. This tool was tested on Eucalyptus by faults injections in a Eucalyptus cloud testbed. This paper shows that software repairs were more often than hardware repairs.

- Security: Eucalyptus manages the access to the cloud using policies related to users, groups and accounts. A group is a set of users within an account, which have the authorization to access to a specified pool of resources. A user can belong to different groups. Security groups are also used by defining firewalls that should be applied to the set of VMs within a group. The security policies could be managed by users using the command line Euca2ools.

3.4 OpenNebula

OpenNebula [139] is an open source cloud platform developed by Universidad Complutense university of Madrid "UCM" (Under the Apache 2.0 License) that delivers a simple, but feature rich, solution to build

enterprise clouds and virtualized data centers. OpenNebula provides a complete toolkit to centrally manage heterogeneous virtual infrastructure, which is compatible with conventional hypervisors: VMware, Xen, KVM. It operates as a scheduler of storage layers, network, supervision and security. It is an appropriate solution for the conversion of a virtual infrastructure to IaaS platform. The centralized orchestration of hybrid environments is the heart of the tool. OpenNebula also utilizes Cloud Computing Interface (OCCI) and support to Amazon Elastic Cloud Compute (EC2) in order to expand the resources connected to it in order to form a hybrid cloud [139].

OpenNebula project started in 2005 by Ignacio M. Llorente and Ruben S. Montero, and has delivered its first version in 2008 and remains active since. Many releases have achieved today significant functional changes on the support of the storage nodes, high availability environments and ergonomics of the administrative interfaces. OpenNebula has also a very wide user base that includes leading companies in banking, technology, telecommunications, and research and supercomputing centers. At present, it has more than 4000 downloads per month and many research institutes and enterprises use it to build their own cloud.

3.4.1 Architecture Component and role

A key feature of OpenNebula's architecture, which we describe in the following, is its highly modular design and flexibility, which facilitates integration with any virtualization platform and third-party component in the cloud ecosystem. The main components of OpenNebula architecture are: the Driver, the Core and the Tools.

A) ***Driver:***

Is responsible for directly communicating with underlying operating system and by the encapsulation of the underlying infrastructure as an abstract service (e.g. virtualization hypervisor, transfer mechanisms or information services). It is designed to plug-in different virtualization, storage and monitoring technologies and cloud services into the core. These pluggable drivers are responsible for the creation, startup and shutdown of virtual machines (VMs), allocating storage for VMs and monitoring the operational status of

physical machines and VMs. The roles of each service are: transfer Driver manages the VMs disk images on different kind of storage systems (a shared one: Network File System (NSF) or Internet Small Computer System Interface (iSCSI), or a non-shared one such as a simple copy over Secure Shell (SSH). VM Driver is considered a set of hypervisor-specific drivers used to manage VMs instances on different hosts. Information Driver is also considered a set of hypervisor-specific drivers used to monitor and retrieve the current status hosts and VMs instances through SSH.

B) ***Core:***

Reflects a centralized layer that control and monitor VM full life cycles, virtual networks (VN), storage and hosts. These components are implemented in this layer by invoking a suitable driver. The feature of such component is: 1) VM manager that allocate resources required by VMs to operate, functionality to implement VMs deployment policies. 2) VN manager has capabilities to interconnect VMs, and to charge of generating MAC and IP address for a VM. 3) Host manager manage a VM's storage and allocate such a VM's disk. 4) SQL Pool is a database (SQLite or MySQL) that stores configuration data and current status of hosts and VMs instances in a database. 5) Request manager (XML-RPC) which can be used to access the application programming interface directly.

C) ***Tools:***

Contains tools distributed with OpenNebula. First, it includes Scheduler that manages the functionality provided by the core layer. Scheduler component makes VM placement decisions. These VMs are deployed on host nodes following specific user requirements and resource-aware policies, such as packing, striping, or load-aware. Secondly, Command line interface-CLI and Libvirt API [10], an open interface for VM management for communicating with users that can manage VM through these interfaces. Additionally, third party tools that can be easily created using the XML-RPC interface or the OpenNebula Client API. External users are capable of sharing these functionalities through a cloud interface which is provided by the Tools layer.

3.4.2 Properties

- Live migration: is one of the advantages of OpenNebula. It supported with shared storage, but it could demand a high-performance SAN (Storage Area Network) [3]. OpenNebula uses libvirt's migration capabilities. More precisely, it uses the TCP protocol offered by libvirt.

- Load balancing: is provided across NGINX [151] which is a modern, open-source, high-performance web server. It is capable of handling a huge number of concurrent connections easily. It represents a centralized manager to balance the workload. In order to distribute efficiently the I/O of the VMs across different disks, LUNs or several storage backends, OpenNebula is able to define multiple system datastores per cluster. Scheduling algorithms (used by load balancers) take into account disk requirements of a particular VM, so OpenNebula is able to pick the best execution host based on capacity and storage metrics [3].

- Fault tolerance: the basic idea of fault tolerant is to add redundant resources to system. Though physical resources can provide fault tolerant very well, it has high cost at the same time. With the development of virtualization we can use virtual resources to provide fault tolerant service [3]. OpenNebula provides VM migration for those not running VMs. However, OpenNebula can only detect problems on VM level. If a service or an application corrupts unexpectedly, simply rebooting the VM on anther node may break the continuity of the service and result in loss. This service is maintained by database backend (registers VM information) to store host and VM information.

- High Availability: OpenNebula delivers the availability required by most applications running in VMs. OpenNebula ensures high availability of the system by using multiple persistent databases backend in a pools cluster of hosts that share datastores and VNs. It provides information in order to prepare for failures in the VMs or physical nodes, and recover from them. These failures are categorized depending on whether they come from the physical infrastructure

(Host failures) or from the virtualized infrastructure (VM crashes). In both scenarios, OpenNebula provides a cost-effective failover solution to minimize downtime from server and OS failures, and supports high availability configurations [55].

- Security: OpenNebula take many measures to ensure the security. The infrastructure administrator manage a secure and efficient Users and Groups Subsystem for pluggable authentication and authorization based on passwords, SSH and RSA key pairs, X.509 certificates or LDAP. OpenNebula uses Firewall to configure the VMs, in order to shutdown TCP and UDP ports, filter some unwanted packets and define a policy for ICMP connections. OpenNebula also uses ACL (Access Control List) which is a collections of permit and deny conditions (ACL rules), allowing different role management with fine grain permission granting over any resource managed by OpenNebula, support for isolation at different levels. OpenNebula add a new mechanism from versions 4.4 as special authentication mechanisms for SunStone (OpenNebula GUI) and the Cloud Services (EC2 and OCCI).

- Compatibility: OpenNebula can be deployed to existing infrastructure and integration with various cloud services and multi-platform. OpenNebula currently includes an EC2 driver, which can submit requests to Amazon EC2 and Eucalyptus, as well as an ElasticHosts driver. OpenNebula supports different access interfaces including REST-based interfaces (e.g., EC2-Query API), OGF OCCI service interfaces, the OpenNebula Cloud API (OCA), and APIs for native drivers, for example, for connecting to AWS.

- Scalability: OpenNebula has tested in the management of medium scale infrastructures with hundreds of servers and VMs [55]. By a cloud federation, OpenNebula provides scalability, isolation, and multiple-site support to interface with external clouds. This allows complementing the local infrastructure with computing capacity from public clouds to meet peak demands. Thus, a single access point and centralized management system can be used to control multiple deployment of OpenNebula. In OpenNebula highly scalability can be achieved through database back-end with support for

MySQL and SQLite, and virtualization drivers can be adjusted to achieve maximum scalability.

- Flexibility and Extensibility: OpenNebula provide easy extension and integration to fit into any existing data center, and flexible architecture, interfaces and components, allowing its integration with any product or tool. OpenNebula offers different means to easily extend and adjust behavior of the cloud management instance to the requirements of the environment and use cases, e.g. new drivers can be easily written in any language for the main subsystems to easily leverage existing IT infrastructure and system management product [3].

3.5 Nimbus

Nimbus is an open source solution (licensed under the terms of the Apache License) for cloud computing to scientific applications. Released in 2005, the solution migrated to GitHub in 2009 and is maintained by an international researchers committee. Although the focus on the scientific community, Nimbus approaches three goals targeting three different communities: (1) Enable resource owners to provide their resources as an infrastructure cloud; (2) Enable cloud users to access infrastructure cloud resources more easily, and (3) Enable scientists and developers to extend and experiment with both sets of capabilities. These goals are related with the architecture of Nimbus. Its main architecture can be divided in two components, each one responsible for attending one of the goals. The first goal is realized by the Nimbus Infrastructure and the second by the Nimbus Platform. The third goal is realized by the strong support of open source development practices via modular, extensible code and engagement with open source developers [20].

3.5.1 General Architecture

This section presents further details on the components Nimbus Platform and Nimbus Infrastructure.

A) ***Nimbus Platform:***

This is an integrated set of open source tools that allow users to easily leverage Infrastructure as a Service (IaaS) cloud computing systems. This includes application instantiation, configuration, monitoring, and repair [20]. The nimbus platform is divided in three modules: the context broker, the elastic scaling tools and the deployment coordination.

B) ***Context Broker:***

Is a service that allows clients to coordinate large virtual cluster launches automatically and repeatably. To deploy the virtual clusters the context broker requires that each VMs run a lightweight script at boot time called the Context Agent. This Context Agent depends only on Python and on the ubiquitous curl program, which securely contacts the context broker using a secret key. To guarantee the security the context broker uses contextualization, *e.g.* the key is created on the fly and seeded inside the instance. This agent gets information concerning the cluster from the context broker and then causes last minute changes inside the image to adapt to the environment [20].

C) ***Elastic scaling tools:***

On Nimbus are called EPU. The EPU system is used with IaaS systems to control highly available services. On these kind of services any failures are compensated with replacements. EPU is also useful with services that can be configured to be elastic; it responds to monitoring signals with adjustments of the amount of instances that composes a service. If the service cannot handle instances being added and dropped on the fly (many services have static node-number configurations), EPU can still provide the automatic launch, monitoring, and failure replacement capabilities [20].

D) ***Nimbus Infrastructure:***

This is a set of tools that provides the IaaS at the nimbus cloud computing solution. The Nimbus infrastructure is divided in Workspace service and Cumulus.

E) ***Nimbus workspace service***

Is a standalone site VM manager that different remote protocol frontends can invoke [20]. The workspace service is web services based and provides security with the GSI authentication and authorization. Currently, Nimbus supports two front-ends: Amazon EC2 and WSRF. The structure of the workspace service is composed by three modules: Workspace resource manager, workspace pilot and workspace control.

- Workspace control: is responsible for controlling VM instances, managing and reconstructing images, integrating a VM to the network and assigning IP and MAC addresses. The workspace control tools operate with the Xen hypervisor and can also operate with KVM. Implemented in Python in order to be portable and easy to install [20, 71].
- Workspace resource management: is an open source solution to manage different VMs, but can be replaced by other technologies such as OpenNebula [20, 71].
- Workspace pilot: is responsible for providing virtualization with few changes in cluster operation. This component handles signals and integrates administration tools [20, 71].

F) ***Nimbus Cumulus:***

Is an open source implementation of the Amazon S3 REST API. It provides an implementation of a quota-based storage cloud. In order to boot an image on a given Nimbus cloud, that image must first be put into that same clouds Cumulus repository, although advanced use cases can bypass this restriction. In practice, it is used as the Nimbus repository solution but can also be installed standalone. Cumulus is designed for scalability and allows providers to configure multiple storage cloud implementations [20].

3.5.2 Properties

- Live Migration Nimbus has no in built support to live migration. It is possible to migrate virtual machines only at hypervisor interface level [124]. The Nimbus team recently participated in a research that addresses network contention between the migration traffic and

the Virtual Machine application traffic for the live migration of co-located Virtual Machines, and the live migration support is on the map of next improvements of the solution [66].

- Load Balancing At VM level, Nimbus features a system of Nagios plugins that can give information on the status and availability of the Nimbus head node and worker nodes, including changes of the virtual machines running on the worker node [119]. Specifically from the cloud provider's point of view Nimbus does the back-filling of partially used physical nodes, allowing also preemptable virtual machines. On the platform level, the cloudInit.d tool provides management to virtual machines deployed and allows compensation of stressed workloads based on policies and sensor information. Nimbus also provides tool to handle capacity allocation and capacity overflow, for example, the ability to give different users different lease limits as a means of scheduling comes standard with Nimbus. In addition, the idea of allowing EC2 or another cloud the ability to pick up excess demand is heavily researched with Nimbus [129].

- Fault Tolerance Nimbus presents fault tolerance only at storage level, through the integration of Nimbus Storage Service with Globus GridFTP [30]. GridFTP - an extension of the standard File Transfer Protocol (FTP) for use with Grid computing - provides a fault tolerant implementation of FTP, to handle network unavailability and server problems. Moreover, transfers can be automatically restarted if a problem occurs [143].

- Availability On the platform point of view, the EPU provides high accessible services to the user. On the infrastructure point of view, Nimbus provides high-available services through the hosted service Phantom. The Nimbus Phantom leverages on-demand resources provided by infrastructure clouds and allows users to scale VMs that are running on the many clouds of FutureSystem as well as Amazon EC2. The Phantom can be extended through decision engines, components that determine the behavior of the service. Phantom is freely available on the FutureSystem infrastructure and is provided as a highly available service itself.

- Security Nimbus provide GSI authentication and authorization - through PKI credentials, grid proxies, VOMS, Shibboleth (via GridShib) and custom PDPs. It also guarantees secure access to VMs via EC2 key generation. It also allows images and image data validation.

- Compatibility Nimbus can be integrated with various cloud services and multi-platform mainly through web services. Nimbus Infrastructure is EC2/S3-compatible, with SOAP and Query frontends. It is possible to upload VM images to Cumulus with the Python library Boto, or with s3cmd. It is also possible to use EC2 spot instances.

3.6 Cloud Solutions Comparisons

This section provides a comparison between the analyzed cloud solutions, aiming at three different levels: (i) General comparison aimed at providing high level comparison in terms of model, policy and architecture, (ii) functional comparison, whose goal is to compare supported functionalities, and (iii) property comparison, which considers cloud properties implemented in these platforms.

3.6.1 General Comparison

The general comparison is provided in Table 3.1, considering general aspects: licensing, cloud model compatibility, business model, architecture, etc.

As it can be seen, the five frameworks are generally equal with respect to business model, licensing policy and cloud models. Each of them has been adopted by large organizations. Nevertheless, a big difference is spotted from the architecture viewpoint. While OpenStack is fragmented into modules, CloudStack has a monolithic central controller, OpenNebula have three main components, Eucalyptus has five parts controllers with AWS, and Nimbus have Lightweight based components. This difference is explained by the open philosophy of Open Stack and OpenNebula which try to avoid technology lock-ins and provides high degree of flexibility, extension and availability. Nimbus is also relatively easy to install and deploy comparing to others solutions. The

Table 3.1: *General comparison*

Property	OpenStack	CloudStack	OpenNebula	Eucalyptus	Nimbus
Open Source License	Apache 2.0	Apache 2.0	Apache 2.0	Linux Open-Source	Linux Open-Source
Commercial model	Free	Free	Free	Free, GPLv3 (only), with proprietary relicensing	Free
Compatibility with	Private, public and hybrid clouds	Private, public and hybrid clouds	Private, public and hybrid clouds	Private and hybrid clouds	Private, public and hybrid clouds
Easy installation	Difficult (many choices, not enough automation)	Medium (Few parts to install)	Easy (process based package installers)	Difficult (different configuration possibilities)	Easy (no root account required)
Architecture	Fragmented into many pieces	Monolithic controller	Modular (third-party component)	Five part controller and AWS	Lightweight components (IaaS service and VMM node)
Large organizations adopters	Yahoo, IBM, VMWare, Rackspace, Redhat, Intel, HP, etc.	Nokia, Orange, Apple, Citrix, Huawei, TomTom, Tata, etc.	CERN, CloudWeavers, IBM, Hexafrid	UEC, NASA, Sony, HP, Cloudera, Puma, USDA, FDA	Brookhaven National Labs, Cumulus project

highly decentralized design and different configurations possibilities of Eucalyptus making it difficult to configure and install. Like Eucalyptus, OpenStack are also characterized by increasing complexity of installation and configuration.

3.6.2 Functional Comparison

The functional comparison looks at the offered functionalities or technical aspects of the five presented solutions, as described in Table 3.2. Most popular hypervisors such as: Xen and KVM are supported by all this platforms, but for example VMware is not available on Nimbus. We mention here that OpenStack and CloudStack have the largest number of supported hypervisors, even though there are ways to interface with non-supported ones. Administration feature is the interface available to interact with these platforms. All solutions present Web interfaces (Web

UI) and command line interfaces (CLI). Also user management is provided in all this five platforms.

Table 3.2: *Functional comparison between the open source Cloud solutions*

Functionality	OpenStack	CloudStack	Eucalyptus	OpenNebula	Nimbus
Supported hypervisors	Xen, KVM, HyperV, VMWare, LXC, vSphere	Xen, KVM, HyperV, VMWare, LXC, vSphere	Xen, KVM, VMware	Xen, KVM, VMware, vCenter	Xen, KVM
Administration	Web UI, CLI	Web UI, CLI	Web UI, CLI	Web UI, CLI	Web UI, CLI
User management	yes	yes	yes	yes	yes

Other important functional aspects of open source cloud solutions are related to the activeness of its community. A solution that is always seeing new releases is constantly evolving. An active community, with well documented wiki, good bug reporting and fixing system and active users support are fundamental features for the success of an open source solution.

About the releases, after its insertion on Apache Incubator, CloudStack made its first major release (4.0.0-incubating) on November of 2012. Basically, it has regular releases in which new feature are introduced. Other releases are for stability of the new features and security. Meanwhile, OpenStack does, since 2012, two big releases per year. In 2012, there were Essex and Folsom, in 2013 Grizzly and Havana and in 2014 Icehouse, Juno in 2015, and Kilo in 2016. In OpenStack, the number of minor releases varies from version to version. For Eucalyptus the last major release is Version Eucalyptus 4.4.x (Released March 7, 2017) and the one before was Version Eucalyptus 4.3.x (Released August 9, 2016). For OpenNebula the last major release is 5.1.80 published in September 16, 2016. But 2 minor releases were announced since that time, the last one is 5.2.1 in January 9, 2017. For Nimbus since its first release in January 9, 2009 only few releases were done. From this we can conclude that OpenStack and Eucalyptus are showing more updates per year.

About the activeness of the community, OpenStack remains the largest and most active open source cloud computing project [19]. Nevertheless,

CloudStack and Eucalyptus are growing and have an important participation in the market. While OpenNebula and Nimbus are less active. In this point, OpenStack has a bigger and more active community, followed by CloudStack, then Eucalyptus, then OpenNebula and finally Nimbus.

3.6.3 Properties Comparison

Properties comparison gives deeper insights into the two platforms, considering some important properties that an IaaS has to provide. The comparison is given in Table 3.3.

Table 3.3: *Properties comparison between the open source cloud solutions.*

Property	OpenStack	CloudStack	Nimbus	Eucalyptus	OpenNebula
Live migration	Yes	Yes	No	No	Yes
Load balancing	Yes	Yes	Yes	Yes	Yes
Fault tolerance	VM scheduling, replication	VM scheduling, replication	Through Globus GridFTP	Through AWS Elastic Load Balancing	VM scheduling, replication
High Availability	Yes	Yes	Yes	Yes	Yes
Security	VPNs, firewall, user authentication, others	VPNs, firewall, user management, others	user authentication	group and users policies	user authentication
Compatibility	Amazon EC2, Amazon S3	Amazon EC2, Amazon S3	Amazon EC2, Amazon S3, WSRF	Amazon EC2, Amazon S3	All Amazon Interfaces

Live Migration may be approached in two ways: shared storage based live migration, and block live migration. OpenNebula and OpenStack offer both possibilities. CloudStack also offers both possibilities, but the conditions change according with the user hypervisor. Eucalyptus and Nimbus offers no integrated live migration system. This way, if the live migration is sensitive to the application, OpenNebula and OpenStack are more adequate solutions.

Load balancing can be considered on the VM level or on the host level. Load balancing at host level is implemented in OpenStack through live migration, which is the same in CloudStack; Nimbus does the backfilling of partially used physical nodes, allowing also preemptable virtual machines. All the solutions approach VM level load balancing through

the establishment of a plug-in architecture. OpenNebula can be coupled with NGINX, Eucalyptus with the ELB of AWS and Nimbus with Nagios plugins. Similarly OpenStack and Cloudstack detail flexible plug-in architecture on network component. In resume, at host level, live migration is used to provide load balancing. At VM level, the cloud solutions trust on plugins to provide load balancing. Also, the automatic leasing of resources seems to be a tendency. For example, allowing EC2 or another cloud the ability to pick up excess demand is heavily researched with Nimbus [129].

Fault tolerance mechanisms exist on VM or on storage/database levels. At VM level, fault tolerance is approached under the policies to schedule VM placement or services replication. OpenNebula comes with a match making scheduler - that implements the Rank Scheduling Policy - and the quote management system, that ensure that any user gets a adequate quantity of resources. OpenStack has in-built scheduling algorithms (group scheduling and rescheduling) and newer ones can be implemented by the user. Nimbus has not already implemented fault tolerance system, but it can provide through Globus GridFTP. At storage or database level, fault tolerance is achieved by using replication and synchronization to ensure that a failure occurred at one device will not break the whole system. Eucalyptus has no in-built mechanism for fault tolerance, but can provide it at storage level, through the AWS Elastic Load Balancing.

High availability is approached in all platforms by means of using redundant service instances and load balancing to distribute workloads among those instances. In the case of the replication of service instances, some synchronization technique has to be considered.

Security is provided on different levels by each cloud solution. Centralized in-built user authentication is provided on Nimbus, OpenStack and OpenNebula. The establishment of security policies for users, groups and accounts is possible on CloudStack, Eucalyptus and OpenNebula. On OpenStack it is also possible to extend the security of the cloud through the addition of plugins.

Compatibility refers to the capability of the cloud solution to integrate with other tools and cloud solution. In this topic, the Amazon Web Services are a common place on the solutions, thanks to its dominance on

commercial public clouds. All open source analyzed solutions, in different levels, present some integration with AWS services, being Amazon EC2 and S3 the most popular.

3.6.4 Summary

In general lines, Eucalyptus offers a flexible solution for users that want privacy in specific modules while keep managing their clouds with AWS. OpenNebula is for someone interested in the internal technical details of the cloud, but also seems to be a good solution for someone that wants to build up a cloud quickly using just a few machines. OpenStack and CloudStack have the largest developing community, but have opposite values, since OpenStack has as modularized architecture, while CloudStack has a monolithic centralized one. Nimbus is focused on the scientific community that requires high customization level.

3.7 Conclusions

We have presented the up-to-date architecture of five prominent open source cloud platforms, looking into details of the provided functionalities and their properties. The analyzed platforms are under continuous development. Therefore their documentation and technical reviews are often updated and have to be regularly checked. We argue that there is no best solution to any general case, but there are tools more adapted to specific audiences. The comparison was carried out from the user perspective, considering the properties that a user needs to know when choosing a IaaS cloud solution.

CHAPTER 4

Joint Framework for management of Cloud Data Centers and Network

4.1 Introduction

In Cloud Computing, data centers are well known for being particularly energy hungry. Electricity consumed by global data centers is estimated to be between 1.1% and 1.5% of total electricity use [100]. Typically, data centers are rather inefficient and consume more energy than required [65], leaving room for improvement achievable through intelligent management techniques.

By breaking down the energy consumption of data centers into their components as shown in Figure 4.1 [120], we can observe that about 52% of energy is consumed by computing equipments and the remaining 48% are for power equipments and cooling systems.

One of the reasons for energy inefficiency is the underutilization of servers whose consumption is not proportional to computing load. As the statistics show, average server utilization in data centers is around 30% [106], due to capacity over provision based on worst-case scenario in order to ensure high levels of reliability [50].

To address this problem and improve Cloud systems power efficiency,

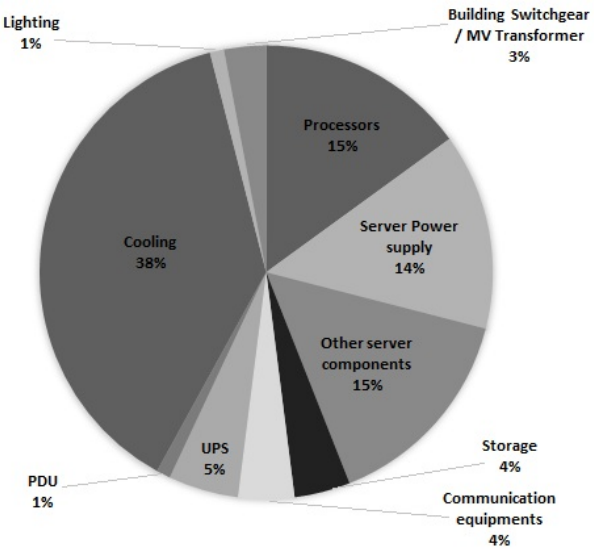

Figure 4.1: *Data Center energy consumption breakdown*

VM (Virtual Machine) migration has been proposed and has shown great potential. Migrating a VM consists of changing its physical host without service interruption. This can be done for different purposes, such fault resilience or for system maintenance. It can be used in power management strategies to move services running on a big set of underutilized servers to a smaller set of optimally loaded servers, so that the others can be switched off for power saving.

VM migration has shown to be very effective if combined with management strategies. It is able to significantly save energy through switching off servers more than that consumed by the migration process itself [88, 137].

Another important component that consumes energy in Cloud services is the communication network. Networks are also typically provisioned for worst-case scenarios such as traffic burst and busy-hours load. Actually, the network has typically a pretty large capacity margin even with respect to peak load for service quality and robustness reasons, and then it usually wastes a lot of energy [37]. In the internal data center network, the main energy consumers are Ethernet switches that are hierarchically interconnected. In the external network based on IP technology, the core network routers dominate energy consumption [86]. The relative contribution to energy consumption of core router components is shown in Figure 4.2 [86]. IP networks typically operate at less than

50% utilization, while still consuming almost 100% of maximum power due to an almost flat energy profile (consumption versus load) [37]. For managing network devices in order to consume less energy, two main approaches are used: turning off the nodes or scaling down their performance [116, 130].

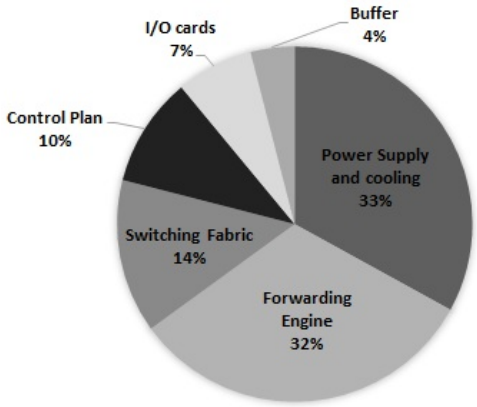

Figure 4.2: *Breakdown of power consumed by a core router*

Most of previous work on energy efficiency in Cloud systems focused on managing computing and networking components separately. However, optimizing energy consumption of data centers and their network independently may be significantly inefficient, in particular when dynamic resource management schemes like VM migration are considered. Considering energy consumption of data center servers only may cause traffic congestion and degrade the quality of Cloud services offered to end users, as well as decrease the energy efficiency of the network.

On the network side, energy saving techniques are based on estimations of the traffic matrices over time, and if data centers are not considered, large traffic variations due to decisions taken by dynamic resource managers can cause energy waste. Integrating techniques for managing energy consumption of computing and networking components in a new generation of Cloud systems can potentially provide non negligible efficiency gains.

A key aspect that makes some level of integration in services offered by data centers and networks particularly important is the geographic distribution of Cloud systems. Distributing data centers over different locations brings Cloud services closer to end users, and offers the opportunity to better exploit the variation of energy prices in different locations

and time zones, as well as the efficient use of the green energy that is locally generated.

Even if there are only a few existing works in literature that have investigated the impact of joint optimization solutions for energy saving in Cloud systems [94, 95, 156], the effectiveness of such solutions from energy cost point of view, and their contribution to reducing environmental impact through the use of green energy remain open issues that have motivated our work (see Section 4.2 for more details).

In this chapter, we present a holistic approach for jointly managing Cloud data centers and their networks. In the considered scenario, the Cloud system provides Platform as a Service (PaaS) to a variety of users, and data centers are distributed geographically in different locations and interconnected by a network. We propose an optimization model based on Mixed Integer Linear Programing (MILP), which has the goal of minimizing the Cloud energy cost and exploit the availability of green energy sources in different places where data centers are located. The proposed approach covers many aspects of Cloud Computing including live migration of VMs, energy storage management, and green energy exploitation. In this model, we consider both energy consumption of data-center servers and their interconnection network, and optimize the use of energy coming from the electrical grid, as well as the energy locally generated using renewable resources exploiting also energy storage

4.2 State of the Art

As mentioned in the introduction, most of the existing works tackle the problem of energy efficiency in Cloud systems separating the management of data center servers and network nodes.

For the data centers, there is a large body of work on energy management of computing resources. We can categorize existing approaches into two classes: server consolidation with power state management and workload scheduling.

Server consolidation consists of efficiently using the available computing resources with the view to reduce the total number of active servers and thus saving energy by turning off unused ones. Entropy is a resource manager proposed in [84] that is based on constraint programming, and

it is able to consolidate applications running on a number of underutilized servers to a smaller number of highly utilized servers using live migration of VMs. The adopted scheme does not take into account heterogeneity in application requirements and servers, which is rather common in multiple cloud provider environments. A similar approach able to cope with heterogeneous environments is named pMapper [142], an application placement controller based on continuous optimization. For more complex environments, with a combination of SLAs (Service Level Agreements), different power models and energy policies, a VM consolidation engine named Plug4Green has been proposed in [69].

The workload placement in modern data centers with a large number of servers significantly affects their operating temperature in addition to energy consumption. A smart placement using workload scheduling techniques may reduce cooling requirements and save even more energy. A good example of schemes based on this scenario is EnaCloud [103]. EnaCloud is an energy-aware heuristic-based approach that chooses the most appropriate scheme for dynamic application placement based on their arrivals, departures or resizing events. The approach in [75] proposes an integer linear programming (ILP) model that combines job allocation and VM migration.

None of the above solutions considers network requirements or the geographic distribution of data centers.

To manage energy within a heterogeneous environment and support a combination of SLAs, different power models and energy policies. Plug4Green, were proposed in [69], it is a VM consolidation engine based on constraint programming. It focuses on flexibility and adaptability to new scenarios. It manages energy by migrating VMs and managing servers on/off states. It can be used as a VM manager that saves energy, but it neither considers network energy consumption nor the use of green energy resources.

In [132], a model that tries to minimize the number of migrated VMs during consolidation and hotspot mitigation is proposed. This model is based on a load prediction algorithm to make decisions regarding migrating VMs and their destinations, and whether the migration is necessary or not. The authors used Utilization VM Selection Policy for VMs selection, and CloudSim simulator [49] for testing. In this algorithm, the

nodes having minimum utilization are considered as being under loaded, which may not be true always and it can cause unnecessary migrations.

An architectural framework for Green Clouds is proposed in [42]. This is mainly a resource allocation approach based on heuristics, which uses a CPU power model for monitoring the energy consumption of the cloud, and computes for each VM to be moved to the appropriate server. Energy minimization is achieved by performing live VM migration and switching idle nodes to the sleep mode. This work is similar to the First Fit Decreasing algorithm, which was used in several previous works [44, 142, 147]. It does not use proactive VM reconfiguration actions, and it neglects the time for powering on/off physical machines.

In [103], the authors proposed an energy-aware heuristic-based approach called EnaCloud. Their main contribution is an algorithm for load aggregation and application scheduling that aims at minimizing energy consumption. It chooses the most appropriate scheme for dynamic application placement based on their arrival, departure or resizing events. Moreover, an over-provision approach for resource resizing is proposed to deal with the varying demands of applications. The architectures of this system relies on a centralized global controller, therefore it faces scalability problems when used for a large-scale data centers.

As for previous work on Cloud networks, available contributions are mainly focused on designing and operating the communication infrastructure in order to achieve fault tolerance, scalability, high utilization and cost efficiency [80, 81]. For energy efficiency, beside the hardware improvements, many contributions related to protocols and network architecture aim at achieving a better trade-off between performance and energy consumption.

In [93], G-Route (Green Route), a service routing protocol for achieving energy-efficiency and collaboration among cloud providers, is proposed. It is a routing scheme that creates autonomous energy-efficient paths between different providers before running a specific service. It has been implemented and tested on Amazon EC2 cloud infrastructure, and shown quite significant energy and cost savings per service request. A drawback of this approach is that it needs a trusted third party to control the energy profiling process. Other energy-aware routing solutions can be found in [26, 31, 36, 43, 45, 57, 113, 123].

Switching off network nodes and rerouting traffic on other paths has a significant impact on saving energy. In [58] the authors proposed an integer linear programing model and some heuristic algorithms that minimize energy by finding the set of routers and links that must remain powered on for a given traffic level while switching off the others. This model is based on the knowledge of the traffic profile exchanged between source/destination nodes, and the maximum link utilization. Unlike most similar works where the objective is to minimize cost or to maximize performance, the authors minimize the total power consumption of the network. Other works that consider switching off network components and sleep mode for saving energy can be found in [25, 47, 72, 90, 107, 131, 133, 145].

PCube [87] is an elastic data center scheme that conserves energy by adjusting the network topology and varying bandwidth availability based on traffic demand. It is designed to be able to dynamically adjust network structure depending on different traffic volumes, and to turn off a set of switches to save energy. Similar solutions are Bcube [79] and ElasticTree [83].

The above solutions focus on network only and do not consider the energy optimization of servers in data centers together with network nodes.

In [27] Addis et al. proposed an approach for network power management based on traffic engineering. Energy consumption is minimized by partial shutdown of different router elements (cards and chassis) depending on the traffic load. The problem was formulated using mixed integer linear programing, in which both fixed and variable routing schemes were considered.

Relatively few work consider joint management of data centers and network. In [95], the authors proposed an optimization approach to jointly minimize the energy consumption in data center hosts and network. The basic idea is to consider both VM placement and traffic routing for energy saving. To avoid the complexity of the problem, a unified representation method is proposed and the optimization model of VM placement is made similar to a routing problem; then the placement and routing problems are solved as a single one. A similar approach is PowerNetS [156], a power optimization strategy based on workload and

traffic correlation analysis. The problem is formulated using constrained programming with the goal of consolidating VMs which are not positively correlated with the same physical machines. At the same time, the model takes into account the network by consolidating VMs that are linked through traffic flows onto the same server or servers close to each other. In [67] the authors proposed an energy saving scheme for VM placement considering both physical servers and network resources. The problem is modeled as a combination of bin packing and quadratic assignment problems with multi-objective optimization and solved with a greedy algorithm that combines hierarchical clustering with best fit scheme. The above approaches focus on the network internal to the data center and do not consider geographically distributed Cloud systems, as well as the availability of green resources and energy storage.

The approach in [78] jointly minimizes the cost in big data processing based on three factors: task assignment, data placement, and data routing. The cost minimization problem is formulated as a mixed-integer nonlinear programming (MINLP) model, which is then linearized to make it tractable. This work considers geographically distributed data centers but it does not explicitly model the external network for energy consumption, it does not exploit the difference of energy prices in various locations, and it does not take into account green energy usage.

In this chapter, we study the energy cost minimization problem of Cloud systems by managing data centers and Network as a whole. Differently from existing works, where the focus is only on one aspect of cloud computing like VM placement/migration, in this chapter we aggregate multiple Cloud Computing aspects into one approach, and propose a holistic energy aware solution for managing Cloud data centers and their interconnection network.

4.3 Global Green Cloud management framework

4.3.1 Model description

We consider a PaaS (Platform as a Service) scenario, where the provider operates on a virtualized infrastructure composed by multiple data centers distributed over different geographical locations. Each data center is equipped with thousands of physical servers. This scenario is rather

common nowadays even if the number of locations and servers vary with the size of the provider. For example, Google data centers are distributed among various locations in the world: 19 in the US, 12 in Europe, one in Russia, one in South America, and three in Asia. While Amazon Web Services Cloud operates 32 Zones within 12 geographic Regions around the world.

Let $\mathcal{I}$ be the set of available data centers. We assume they are fully connected by a backbone network, where in each path between two data centers i and j, the number of routers and the available bandwidth capacity are known.

We assume that the Cloud Provider is able to host different user applications by offering a set $\mathcal{L}$ of heterogeneous types of VMs. Each type of VMs executes a specific service application that is capable to serve a set $\mathcal{K}$ of user request classes as shown in the system model [Figure 4.3].

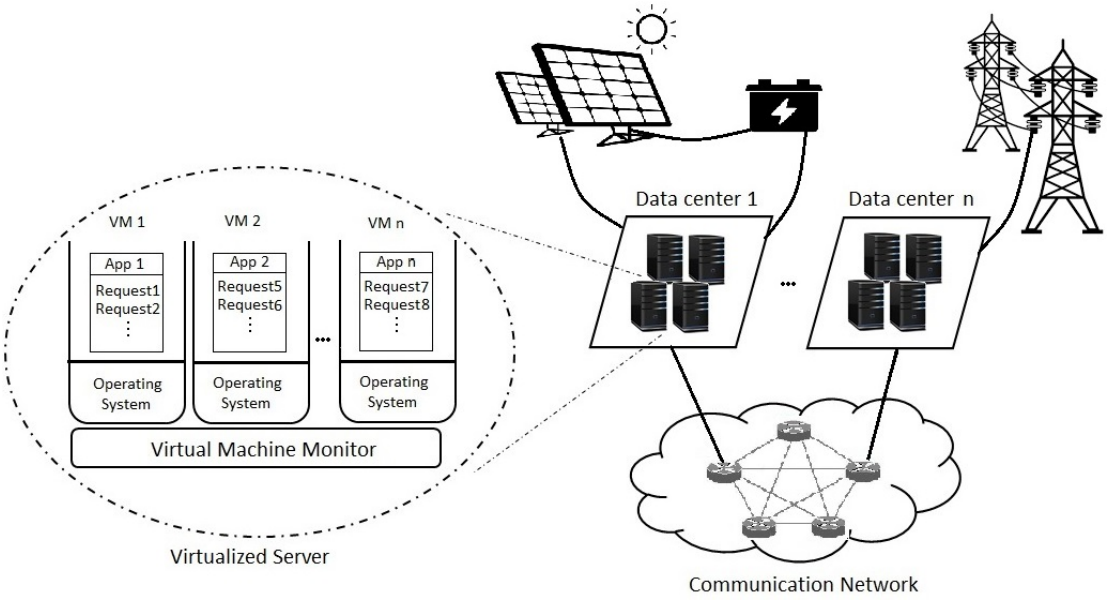

Figure 4.3: *Cloud System Model*

We consider a one-day horizon, divided into 24 time periods, and we solve the problem in advance for each day. We consider predictions of the application workload based on historical traffic information [38, 138, 146], thus, an estimation of the incoming traffic for each application is provided. We denote by λ_{ik}^{t} the arrival rate of requests of class $k \in \mathcal{K}$ to data center i at time $t \in \mathcal{T}$.

Based on the traffic profile, our goal is to minimize the total energy

cost in the cloud system by allocating VMs to servers and if necessary, migrating them between data centers, considering the fact that migration itself costs energy on both source and destination. Depending on the location of a data center and its time zone (day/night), the price of energy varies. We exploit different energy regions by migrating VMs to data centers where the price of energy is cheaper. We also consider the availability of energy from renewable resources for reducing environmental impact of the cloud system. Therefore, migration of VMs to data centers with more available green energy is an opportunity that can be exploited to optimize costs. Basically, using VM migration we actually reduce the load of expensive and polluting data centers, while we exploit cheap and green energy when available.

We consider that VMs are live migrated between DC using post-copy live migration scheme [85], in which the VM is suspended immediately upon beginning of the migration process. First CPU state is transferred to the destination DC, while the memory is still at the source DC. Then the destination DC requests fault pages from the source DC, while the latter is transferring the memory state to the destination DC. Post-copy live migration decreases the migration traffic therefore reduces the total migration time, since the VM page is transferred only once over the network unlike in pre-copy live migration [60]. Indeed, in this case memory pages are copied from the source DC to the destination DC without interrupting the execution of the virtual machine, which implies a succession of iterations of memory transfer before stopping the VM execution on the source DC and starting it again at the destination DC.

By migrating VMs from one data center to another, we consider migration energy cost of the destination data center, the source data center, and the network. For the network, its energy is assumed to be proportional to the amount of exchanged traffic, which includes bandwidth consumed by migrated VMs including their memory, disk images of VMs , and users traffic.

Moreover, we jointly manage the use of green and brown energy. We assume that all the energy coming from the electrical grid is brown energy, while we consider the energy generated in data centers as green and available for free if it is consumed locally, to privilege the use of on-site generated green energy. However, it is straightforward to modify

the formulations to include also an amount of green energy coming from the electrical grid (with a cost depending on location).

We assume that data centers are able to generate on-site an amount of green energy using renewable sources, such as solar energy, wind energy or geothermal energy. The use of this smart electricity sourcing strategies on-site is increasing, e.g., Facebook's solar-powered datacenter [109], and Green House Data wind-powered datacenter [110]. Since our work focuses only on system management, we do not include capital expenditures for renewable sources. The costs of green energy generation are significantly declining over the last few years. Depending on the technology, installation prices vary, e.g., Parabolic trough plants used to generate Concentrating Solar Power (CSP) have capital costs as low as 4600$/kW in USA market, while wind power technologies tend to be more competitive, between 1800$/kW and 2200$/kW [21].

Matching exactly the energy consumption with green energy generation is difficult and can potentially generate inefficiencies when produced energy cannot be consumed immediately. Therefore, we relax this problem by considering the use of energy storage technologies. In our scenario, data centers are equipped with rechargeable battery systems that are able to store the locally generated green energy. Balancing the use of green energy produced, between immediate usage and storage in batteries for later consumption allows for green energy to be available when the price of brown energy is high, as well as to solve the problem of discontinuous availability of renewable resources.

For each time period, the model defines how to allocate the load in each data center. In other words, how many VMs are kept active or off. The same thing for the network, where we define for each time period which links are to be turned on and which should be off depending on the number of routers in each link and their capacity. Note that, even if we associate energy consumption to links, the real energy consumers in the network are line cards connected to the links in the routers on both sides.

4.3.2 Model Formulation

In this section, we first introduce decision variables. We then formulate the cost minimization objective function and the problem constraints.

4.3.2.1 Decision Variables

The goal of our optimization model is twofold: 1) finding how many VMs to be migrated among DCs, and the source and destination of the migrations, 2) managing the usage of the available green energy sources. These decisions allow to move the load and the energy consumption among DCs during the day in order to exploit the availability of green energy and the differences in brown energy prices of various DC locations.

We formulate the problem using several sets of decision variables. The first pair of main decision variables refer to VMs migration and requests forward. The integer variable v_{ijl}^t represents the number of VMs of type l to be migrated from data center (DC) i to data center j during time period t. This variable depends on the number of received requests by each data center. We use a continuous positive variable x_{ijkl}^t to represent the arrival rate of class k requests received by VM of type l in data center i and then served in data center j after live migration.

The second pair of main decision variables are related to the green energy management. In particular, variables sge_i^t and dge_i^t indicate the sources of the green energy supplied to the DC i during the time period t. sge_i^t is the amount of energy coming from DC batteries, while dge_i^t is the green energy produced at the DC i directly supplied to the DC for its operations.

Together with these main decision variables, there are several other secondary variables that, depending on the values of the main ones, are used to model the behavior of the system. We can group them in the three domains, they refer to: VM and Migration, Networking, Battery and Green Energy Management.

4.3.2.2 VM and Migration

Integer variable $\overline{w}_{il}^t$ denotes the number of active VMs type l that are originally running on data center i, while we use the integer variable $\underline{w}_{il}^t$ to refer to the number of VMs executing on a data center i after all live migrations took place. The variables won_{il}^t and $woff_{il}^t$ depict the number of VMs to be turned on and off respectively at time period t. The energy consumption associated to the migration of type-l VMs, including both

current VMs at DC i migrated to other DCs and new VMs migrated to DC i from other DCs, is captured by the variable mig_{il}^{t}. Figure 4.4 describes a small illustrative scenario of the migration process, in which a DC i receive a number v_{ijl}^{t-1} of migrated VMs from a DC j during the time slot $t-1$, then, during time t it migrates a number v_{ijl}^{t} to a DC j, we mention the space between time bands is just to show the number of turned off VMs $woff_{il}^{t}$.

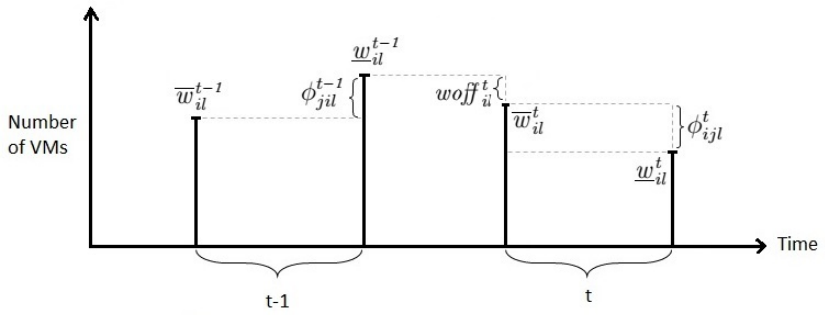

Figure 4.4: *VM variables*

4.3.2.2.1 Networking As for the network, we assume that the energy consumption of a link is proportional to its load, expressed in terms of the ratio of used bandwidth over available bandwidth. We rely on variable b_{ij}^{t} to express the bandwidth used at the link (i, j) connecting DC i with DC j during the time period t, which is determined by the traffic volume exchanged by the two DCs due to all migration processes among them. In addition, we let unused links to be switched off. We capture this behavior using binary variables z_{ij}^{t}, equal to 1 if the link (i, j) is active, and 0 otherwise. Similarly to data centers, zon_{ij}^{t} and $zoff_{ij}^{t}$ indicate whether each link has to be turned on or off at the beginning of time period t according to its status during the previous time period $(t-1)$. Figure 4.5 illustrates networking variables during a small scenario for transmissions in the link between a DC i and a DC j.

4.3.2.2.2 Battery and Green Energy Management Concerning green energy and batteries, the variable c_{i}^{t} represents the amount of energy charged in a battery i at time t from renewable energy sources installed at DC i. It is related to the main decision variables sge_{i}^{t} and dge_{i}^{t} as described by the

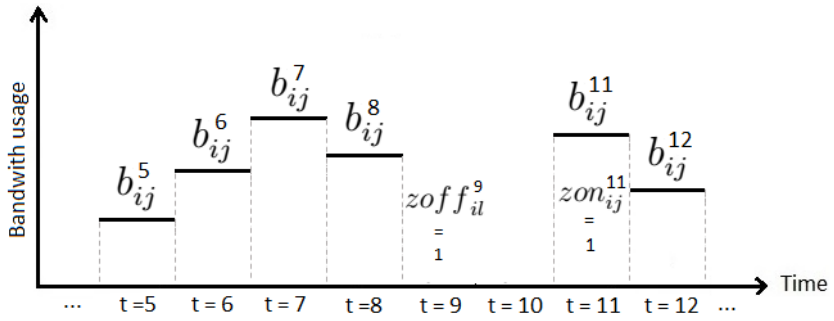

Figure 4.5: *Networking variables*

Figure 4.7, where basically the generated green energy not immediately provided in dge_i^t is used to recharge the batteries.

In addition to the above-mentioned variables, we have a set of variables to models the energy charging and discharging phases at DC i. We suppose that the energy charged at a time t cannot be used in the same time period. To define the energy level of a battery we use two different variables, one refers to the level of energy at the beginning of a time period t denoted by $\overline{s}_i^t$, and the other one $\underline{s}_i^t$ refers to the level of energy at the end of the time period t. Figure 4.6 shows how variable sge_i^t, which indicates the amount of the batteries' energy consumed during time period t to run the DC, is connected to other energy-related variables. Finally, variables sge_i^t and dge_i^t define the total amount of green energy provided to the DC during the time period t, expressed by the variable g_i^t

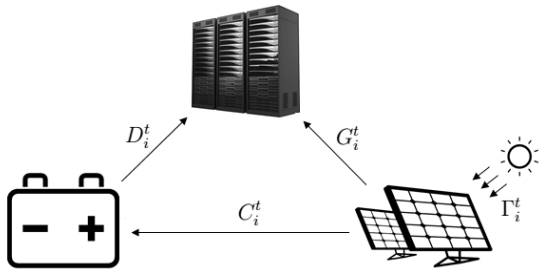

Figure 4.6: *Energy Generation, Storage and Consumption*

All defined variables are summarized in Table 4.1.

4.3.2.3 Objective function

The objective of our model is to minimize the energy cost, which consists of two components: DC energy consumption and networking en-

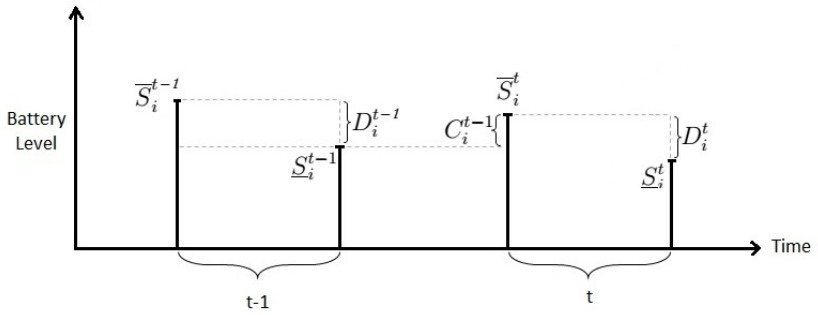

Figure 4.7: *Batteries Functioning*

Main Decision variables	
v_{ijl}^t	number of migrated VM type l from DCi to DCj at time t
x_{ijkl}^t	number of requests class k received by i and treated in j
sge_i^t	energy supplied to DCi from batteries at time t
dge_i^t	green energy directly supplied to DCi at time t
VMs related variables	
$\overline{w}_{il}^t$	number of VMs type l at DCi before live migration
$\underline{w}_{il}^t$	number of VMs type l at DCi after live migration
mig_{il}^t	migrations energy consumption of VMs class l in DC i during t
won_{il}^t	number of VMs type l to be turned on
$woff_{il}^t$	number of VMs of type l to be turned off
Network variables	
b_{ij}^t	Bandwidth used in link (i,j) at time t
z_{ij}^t	link (i,j) status at time t (binary)
zon_{ij}^t	Whether the link (i,j) has to be turned on (binary)
$zoff_{ij}^t$	Whether the link (i,j) has to be turned on (binary)
Batteries and Green Energy variables	
c_i^t	Energy charger in battery i at time t
$\overline{s}_i^t$	energy level in battery i at the beginning of time period t
$\underline{s}_i^t$	energy level in battery i at the end of time period t
g_i^t	all green energy used by DC i during t

Table 4.1: *Decision Variables*

ergy consumption. Therefore, we design the following objective function:

$$\min \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{I}}\left\{M_i^t\left[\rho_i\sum_{l\in\mathcal{L}}\left(\alpha_{il}\underline{w}_{il}^t+\eta_{il}won_{il}^t+\theta_{il}woff_{il}^t+mig_{il}^t\right)-g_i^t\right]\right\}$$
$$+\sum_{t\in\mathcal{T}}\sum_{i,j\in\mathcal{I}}E_{ij}^tR_{ij}\left[(\gamma_{ij}-\delta_{ij})\frac{b_{ij}^t}{Q_{ij}}+\delta_{ij}z_{ij}^t+\tau_{ij}zon_{ij}^t+\xi_{ij}zoff_{il}^t\right] \tag{4.1}$$

The first term accounts for the cost of data centers consumption. It considers the costs of all data centers over all time periods, where for each data center we multiply the specific site cost of brown energy, M_i^t, and PUE (Power Usage efficiency), ρ_i, for the total energy consumed by the servers. The consumed energy consists of:

- the total consumption of running VMs, where α_{il} is energy needed for running a type l VM in DC i (e.g., Wh)
- the energy needed for turning on and off the servers, where η_{il} and θ_{il} are, respectively the energy needed for turning on or off a type l VM in DC i (e.g., Wh)
- the energy consumed by DCs to migrate VMs, captured by variable mig_{il}^t

In addition, the consumed energy is discounted by the amount of green energy provided by renewable energy sources installed at DCs represented by g_i^t. As we consider the green energy produced locally, the operating cost of green plants is set to zero in the model.

The second term of the objective function accounts for the network consumption. It is computed as a sum of each path cost, which in turn consists of two components:

- the energy required to operate each router on link (i, j) during the time period t, considering both active and idle state energy consumption, respectively indicated by γ_{ij} and δ_{ij}
- the energy required to turning on and off a router at the beginning of the time period, respectively, τ_{ij} and ξ_{ij}

The model assumes that all the routers of the link (i, j) are identical, therefore, the energy consumption of each router multiplied by the number of routers along the link, R_{ij}. This assumption can be easily modified by considering individualized energy consumption values, which have been omitted here for sake of ease in presentation. Finally, in order to compute the energy cost of this second term, the total energy consumption is multiplied by the average energy price along the link during time period t, E^t_{ij}.

4.3.2.4 Constraints

In this section we present different groups of constraints used to model the Vm migration, the operations of data centers, the network, and battery and green energy management features.

4.3.2.4.1 VM Migration First, we must ensure that all the requests received from cloud users are processed by the data centers. The requests of different classes have to be processed by suitable type of VMs. For this purpose we use the following constraints:

$$\sum_{l \in \mathcal{L}_k} \sum_{j \in \mathcal{I}} x^t_{ijkl} = \lambda^t_{ik} \quad \forall i \in \mathcal{I}, \forall k \in \mathcal{K}, \forall t \in \mathcal{T} \tag{4.2}$$

In particular, constraint (4.2) ensures that all the incoming traffic is processed in the Cloud, by any of the DCs with an appropriate VM. Note that we consider the set $\mathcal{L}_k$, which is the set of VM classes that can process class-k requests.

In addition, a migration plan requires to define the number of VMs to migrate, as well as their source and destination DCs. For this purpose we use the following constraints:

$$\sum_{j \in \mathcal{I}} v^t_{ijl} \geq \sum_{k \in \mathcal{K}} \big[\big(\sum_{j \in \mathcal{I}} x^t_{ijkl} \big) - x^t_{iikl} \big] / \mu_l \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{4.3}$$

$$\sum_{j \in \mathcal{I}} v^t_{jil} \geq \sum_{k \in \mathcal{K}} \big[\big(\sum_{j \in \mathcal{I}} x^t_{jikl} \big) - x^t_{iikl} \big] / \mu_l \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{4.4}$$

$$v^t_{iil} = 0 \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{4.5}$$

Equations (4.3) and (4.4) compute the number of VM that were sent and received, respectively, by each DC. They are proportional to the rate of request, respectively, redirected to and received from other DCs. Note that the term x^t_{iikl} represents the requests arrived to DC i and locally served. The number of migrated VMs must be sufficient to serve the rate of forwarded requests, this is captured by the parameter μ_l expressing the maximum service rate for a type l VM. The last constraint (4.5) guarantees that a DC does not migrate VMs to itself.

The following constraint (4.6) determines the number of active VMs in a data center i after making the necessary migrations. Starting from the number of VMs that was created in a data center ($\overline{w}^t_{il}$), we subtract the number of VMs that migrated and add the ones that arrived from other locations. In most cases, a data center makes one of these operations: sending or receiving VMs but not both at the same time, and that depends on its capacity and the energy constraints.

$$\underline{w}^t_{il} = \overline{w}^t_{il} - \sum_{j \in \mathcal{I}} v^t_{ijl} + \sum_{j \in \mathcal{I}} v^t_{jil} \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{4.6}$$

Finally, constraint (4.7) calculates the energy consumed by migration operations.

$$mig^t_{il} = EH_l \sum_{j \in \mathcal{I}} v^t_{jil} + ES_l \sum_{j \in \mathcal{I}} v^t_{ijl} \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{4.7}$$

Parameters ES_l and EH_l represent the energy consumption for migrating a VM type l consumed, respectively, at source and destination DC.

4.3.2.4.2 DC behavior Together with the migration plan, we need to model the DC behavior. We assume each DC has a maximum number of requests that can process per time period. This number depends on the number of VMs that a data center is able to handle simultaneously. Hence, we use the following constraints to ensure that the capacity requirements of data centers are not exceeded:

$$\underline{w}^t_{il} \leq P_{il} \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{4.8}$$

$$\overline{w}_{il}^{t} \geq \sum_{j \in \mathcal{I}} \sum_{k \in \mathcal{K}} \frac{x_{ijkl}^{t}}{\mu_l} \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{4.9}$$

$$\underline{w}_{il}^{t} \geq \sum_{j \in \mathcal{I}} \sum_{k \in \mathcal{K}} \frac{x_{jikl}^{t}}{\mu_l} \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{4.10}$$

Where, constraint (4.8) ensures that the number of running VMs after all migration operations took place does not exceed the capacity of the system resources. In other words, the overall utilization of resources dedicated to run class-l VMs is below a planned threshold in each DC i, P_{il}. Constraint (4.9) defines the number of VMs of type l originated in DC i, while constraint (4.10) defines the number of VMs of type l running on DC i after making all the necessary live migrations. Both numbers depend on outgoing and ingoing rates of migrated requests.

In order to ensure time continuity in the number of running VMs at each DC, we need the following constraints.

$$won_{il}^{t} \geq \overline{w}_{il}^{t} - \underline{w}_{il}^{t-1} \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{4.11}$$

$$woff_{il}^{t} \geq \underline{w}_{il}^{t-1} - \overline{w}_{il}^{t} \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{4.12}$$

Constraints (4.11) and (4.12) determine the number of VMs to be turned on and off at the beginning of time period t, according to their number at the end of time period $t-1$.

4.3.2.4.3 Networking For the network, the following constraints are defined to ensure that we do not exceed bandwidth capacity and to guarantee the proper operation of network links:

$$b_{ij}^{t} = \sum_{l \in \mathcal{L}} \phi_{ijl}^{t} \big(VMsize_l + DI_l \big) + \sum_{k \in \mathcal{K}} \big(B_k \sum_{l \in \mathcal{L}} x_{ijkl}^{t} \big) \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{I}, \forall t \in \mathcal{T} \tag{4.13}$$

$$b_{ij}^{t} + b_{ji}^{t} \leq Q_{ij} z_{ij} \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{I}, \forall t \in \mathcal{T} \tag{4.14}$$

$$z_{ij}^{t} = z_{ji}^{t} \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{I}, \forall t \in \mathcal{T} \tag{4.15}$$

Constraint (4.13) computes the portion of bandwidth used for transferring data between different DCs. In our scenario, all the exchanged data is related to migration operations. Basically, we consider the size of different types of migrated VMs, in terms of memory state and the content of CPU registers, then, we associate to each type of VMs an amount of users traffic in terms of requests. Parameters $VMsize_l$ and DI_l indicates the bandwidth consumed to migrated a type-l VM and the size of disk images of type-l VMs, while B_k is the bandwidth required for a type-k request. Constraint (4.14) guarantees that the VM exchanges do not exceed the link capacity, Q_{ij}, which is forced to 0 when the link is switched off ($z_{ij} = 0$). Finally, Constraint (4.15) ensures that if a link is active in one direction, it is also active in the other one.

Similarly to the case of the number of VMs at DCs, we need to ensure the time continuity of the link status. Constraints (4.16) and (4.17) define which links have to be turned on or off at each time period transition.

$$zon_{ij}^t \geq z_{ij}^t - z_{ij}^{t-1} \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{4.16}$$

$$zoff_{ij}^t \geq z_{ij}^{t-1} - z_{ij}^t \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{4.17}$$

4.3.2.4.4 Green Energy and Batteries management Green energy management and storage play an important role in our model. The constraints below guarantee the appropriate behavior of the available renewable resources and batteries.

$$g_i^t \leq \rho_i \sum_{l \in \mathcal{L}} \left(\alpha_{il} \underline{w}_{il}^t + \eta_{il} won_{il}^t + \theta_{il} woff_{il}^t + mig_{il}^t \right) \quad \forall i \in \mathcal{I},\ \forall t \in \mathcal{T} \tag{4.18}$$

$$g_i^t = sge_i^t \beta_i + dge_i^t \quad \forall i \in \mathcal{I}, \forall t \in \mathcal{T} \tag{4.19}$$

$$c_i^t + dge_i^t \leq \Gamma_i^t \quad \forall i \in \mathcal{I}, \forall t \in \mathcal{T} \tag{4.20}$$

$$\sum_{t \in \mathcal{T}} g_i^t \leq \sum_{t \in \mathcal{T}} \Gamma_i^t \quad \forall i \in \mathcal{I} \tag{4.21}$$

In particular, constraint (4.18) ensures that the consumed green energy during a time period is less than the required amount required to run the corresponding DC. Constraint (4.19) states that the green energy at time period t can be provided directly from the renewable source (dge_i^t) or from batteries (sge_i^t), taking into account the energy loss rate during a time period due to its storage, denoted by β_i. Constraint (4.20) guarantees that the amount of energy charged in batteries and the one directly supplied are less than the total amount generated in one time period, denoted by Γ_i^t, while constraint (4.21) ensures that the total green energy consumed during a day in a data center ($\sum_{t_\mathcal{T}} g_i^t$) does not exceed the total amount generated ($\sum_{t_\mathcal{T}} \Gamma_i^t$). This makes the daily repetition of the plan sustainable.

$$\overline{s}_i^t = c_i^{t-1}\psi_i + \underline{s}_i^{t-1}\zeta_i \quad \forall i \in \mathcal{I}, \forall t \in \mathcal{T} \setminus \{1\} \tag{4.22}$$

$$sge_i^t = \overline{s}_i^t - \underline{s}_i^t \quad \forall i \in \mathcal{I},\ \forall t \in \mathcal{T} \tag{4.23}$$

$$sge_i^t \leq \overline{s}_i^t \quad \forall i \in \mathcal{I},\ \forall t \in \mathcal{T} \tag{4.24}$$

Constraints (4.22), (4.23), and (4.24) are related to batteries. Constraint (4.22) states that the energy level in a battery at the beginning of the period t is given by the energy remaining at the end of time period $t-1$ (considering the energy discharging efficiency of the battery ζ_i) and the energy charged during $(t-1)$ (considering the energy charging efficiency of the battery ψ_i). Constraint (4.23) forces the amount of discharged energy, sge_i^t, to be equal to the difference between the level of energy at the beginning and at the end of time period t. Constraint (4.24) makes sure that the energy discharged from a battery is less than the available energy in that battery at the beginning of the time period.

The following final constraints are related to physical limitations of the batteries. Constraints (4.25), (4.26) and (4.27) ensure that the model does not exceed the energetic capacity and charging and discharging rate limits of a battery.

$$\overline{s}_i^t \leq Smax_i \quad \forall i \in \mathcal{I},\ \forall t \in \mathcal{T} \tag{4.25}$$

$$c_i^t \leq Cmax_i \quad \forall i \in \mathcal{I},\ \forall t \in \mathcal{T} \tag{4.26}$$

$$sge_i^t \leq Dmax_i \quad \forall i \in \mathcal{I},\ \forall t \in \mathcal{T} \tag{4.27}$$

Parameters $Smax_i$, $Cmax_i$, and $Dmax_i$ refer, respectively, to the maximum energy storage capacity, energy charging in one hour, and energy discharging in one hour of the considered batteries. We also assume the energy charged into a battery during time t cannot be used until the next time period, therefore, we add the constraints (4.28), (4.29) and (4.30) to initialize batteries' status in $t = 1$.

$$sge_i^1 = 0 \quad \forall i \in \mathcal{I} \tag{4.28}$$

$$\overline{s}_i^1 = 0 \quad \forall i \in \mathcal{I} \tag{4.29}$$

$$\underline{s}_i^1 = 0 \quad \forall i \in \mathcal{I} \tag{4.30}$$

Table 4.2 summarizes all model parameters.

4.4 Model Evaluation

Our model has been evaluated using a state-of-the-art MILP solver and considering various instances and workload configurations. In this section, we present results obtained on a set of scenarios with realistic values for parameters.

4.4.1 Parameter Setting

We considered 15 data centers distributed geographically all over the world. For their locations, we have taken inspiration from Cloud Computing infrastructure of Google [77]. We used four geographical macro areas: West USA, East USA, Europe and Asia. In each area, data centers have the same or close time zone. A detailed view of used data centers location is provided in Table 4.3.

For each data center, we associate a PUE value in order to include the power facilities that support the IT equipment load, such as cooling systems. According to [76] the global average PUE of the largest data

$\mathcal{I}$	Set of Data Centers
$\mathcal{T}$	Set of Time bands
$\mathcal{L}$	Set of VMs types
$\mathcal{K}$	Set of request classes
Data centers and VMs parameters	
λ_{ik}^t	incoming class k requests rate at DC i in time t (Req/hour)
B_k	bandwidth required for a type k request
P_{il}	the maximum number of VMs type l in DC i
$VMsize_l$	bandwidth consumed to migrated a type-l VM type
DI_l	size of disk images of VMs type l
μ_l	maximum service rate for a type l VM
EH_l	destination DC energy consumption for migrating a VM type l
ES_l	source DC energy consumption for migrating a VM type l
α_{il}	energy needed for running a type l VM in DCi)
η_{il}	energy needed for turning on a type l VM in DCi
θ_{il}	energy needed for switching off a type l VM in DCi
ρ_i	PUE (Power Usage efficiency) for DC i
M_i^t	price of energy in DC i at time t
Network parameters	
R_{ij}	number of routers in link (i, j)
Q_{ij}	Maximum bandwidth in link (i, j)
γ_{ij}	energy needed for running a router in (i, j)
δ_{ij}	energy needed for keeping idle a router in (i, j)
τ_{ij}	energy needed for turning on a router in link (i, j)
ξ_{ij}	energy consumption for switching off a router in (i, j)
E_{ij}^t	price of energy in link (i, j)
Batteries and Green Energy parameters	
Γ_i^t	Green Power that could be generated at DC i in time t (kWh)
ψ_i	energy charging efficiency in DC i battery
β_i	energy loss rate per time in DC i battery
ζ_i	energy discharging efficiency in battery i
$Smax_i$	maximum energy storage capacity in battery i
$Cmax_i$	maximum energy charging in one hour for battery i
$Dmax_i$	maximum energy discharging in one hour for battery i

Table 4.2: *Model Parameters*

DC	City	Country	Macro-area	Time zone
DC1	Mountain View - CA	USA	West America	UTC-08
DC2	Pleasanton - CA	USA	West America	UTC-08
DC3	San Jose - CA	USA	West America	UTC-08
DC4	Atlanta - Georgia	USA	East America	UTC-05
DC5	Reston - Virginia	USA	East America	UTC-05
DC6	Berlin	Germany	Europe	UTC+01
DC7	Groningen	Netherlands	Europe	UTC+01
DC8	Mons	Belgium	Europe	UTC+01
DC9	Paris	France	Europe	UTC+01
DC10	Dublin	Ireland	Europe	UTC+00
DC11	Milan	Italy	Europe	UTC+01
DC12	Moscow	Russia	Asia	UTC+03
DC13	Tokyo	Japan	Asia	UTC+09
DC14	Hong Kong	China	Asia	UTC+08
DC15	Beijing	China	Asia	UTC+08

Table 4.3: *Data Centers location*

centers is around 1.7, while the average PUE for all Google data centers is 1.12. In our tests, we vary PUE values between 1.1 and 2.

For data centers capacity, we generated a random number of physical servers for each DC, within the range [5000:16000], and we assume 1:1 ratio for the physical to virtual resources assignment (i.e., 1 physical core is assigned to 1 virtual core of equal capacity).

Regarding technical characteristics of servers in DCs, we consider an HP ProLiant DL370 G6, with a Intel Xeon W5580 processor (8 cores at 3200 Mhz) and 96GB of total memory. Even if we considered three different classes of VMs (see below), we modeled only a single server type, in order to simplify energy consumption analysis. For this reason, all VMs require the same amount of energy to run at peak load or when idle, while they differ in the class of requests that can be processed and their total number per time period. The values of energy consumed for migrating a VM are taken from an experiment that is designed to estimate servers (Host/Destination) consumption due to live migration [137].

Another important parameter is the energy cost. Energy prices varies during a 24 hours period, we consider that peak hours do not occur simultaneously at different time-zones. The values used are obtained based on an analysis of various energy markets in the world including GME (Gestore dei Mercati Energetici) in Italy, New England Market and PJM in

California USA, SEMO in Ireland and many others. Therefore, the costs of energy per MWh varies between 10 and 65 Euro/MWh.

We assume that data centers are fully connected, thus, we consider that capacity of different links varies between 0.5 Gbps and 1 Gbps. Moreover, a typical link connecting data centers is built up both by physical lines (such as optical fiber) and network components (such as routers and switches). Therefore, we estimate the energy cost of each link as the cost of energy consumed by its routers, proportionally to the bandwidth in use. For the number of routers in each link, a traceroute application was used to determine the number of hopes between two nodes. We have also considered a single reference router which is a Juniper E320, with a maximum power consumption of 3.84 kW [23]. The other values of parameters related to routers are listed in Table 4.4.

We built workloads based on a trace of requests registered at a website of a big University. This trace was collected hourly during one year, from sessions registered on 100 servers. To generate the workload, we consider the total number of Internet users for each country where a data center is located [91], then, based on the number of Google search done all over the world and percentages of Internet users of the same country, we estimate the requests rate for each data center.

For VMs types, we consider 3 classes of VMs, where each class is able to serve 5 types of requests. For VMs size and related parameters, we took inspiration from Amazon EC2 Instance Types [32]. Disk images size varies depending on the type of VMs. In the considered scenario, we assume that the type of VM are not storage intensive therefore we consider that the size of disk images is between 0.5 and 20 GB.

In order to estimate the total amount of green energy produced by each data center during a single day, we multiply the average energy produced by a green plant per square meter with the average data center size that we vary between $450\ \mathrm{m}^2$ and $10000\ \mathrm{m}^2$ [40]. Moreover, we consider that data centers are equipped with Li-ion (lithium-ion) batteries with overall capacity of 1486 Ah. This kind of batteries have a C-rate around 73 Ah per module, so with a voltage of 14.8 V, a module can charge 1.08 kWh during one hour, which is almost the full capacity of a module. Energy charging and discharging efficiency are considered equal to 88% [63]. Table 4.4 summarizes parameter settings used to test

the model.

B_k [200, 450] kb	α_{il} [60, 90] Wh	η_{il} [2, 3] Wh
θ_{il} [0.28, 14] Wh	ρ_i [1.1,2]	γ_{ij} 3.84 kWh
δ_{ij} 0.768 kWh	τ_{ij} 0.128 kWh	ξ_{ij} 0.128 kWh
EH_l [203, 908] Ws	ES_l [203, 908] Ws	PUE [1.1,2]
M_i^t [10, 65] Euro/MWh	$VMsize_l$ [800, 2700] Mb	DI_l [0.5, 20] Gb
$Smax_i$ 1486 Ah	$Cmax_i^t$ 1.08 kWh/module	ψ_i and β_i = 88%

Table 4.4: *Parameters settings*

4.4.2 Numerical Results

We used the commercial solver IBM ILOG CPLEX 12.1 as a MILP optimization solver [148]. The model has been run on an 8-core 2.4GHz Intel Xeon server with 96 GB RAM. To evaluate the energy saving of joint optimization and compare it to traditional strategies that separate data centers and network energy management, we considered the proposed model denoted by *Global Green* and two different scenarios. The *Servers Only* scenario, where we consider a modified version of the model that includes the optimization of servers only (decision variables) while it just accounts for energy consumption of the network without optimizing it. A second scenario, called *Separated*, consists of two separate tests: for data centers, the energy consumption and its cost are calculated solving the *Servers Only* scenario, while the corresponding values for the network are calculated from another modified version of the model that optimizes the network only and takes as input traffic values from the *Servers Only* solution.

Figure 4.8 shows the energy costs for the different scenarios for different values of traffic (number of requests per day). We also report the energy savings (in percentage) of the *Global Green* scenario with respect the other two. The joint optimization approach (*Global Green*) can save large amount of energy cost up to 70% compared to the cost without network power management. The reason behind this is the non-negligible energy saved in the network turning on and off routers according to traffic dynamics. The savings can be up to 34% compared to the separate case, and this is due obviously to the separate solution of the two problems that leads to suboptimal solutions, and to the use of fixed traffic values

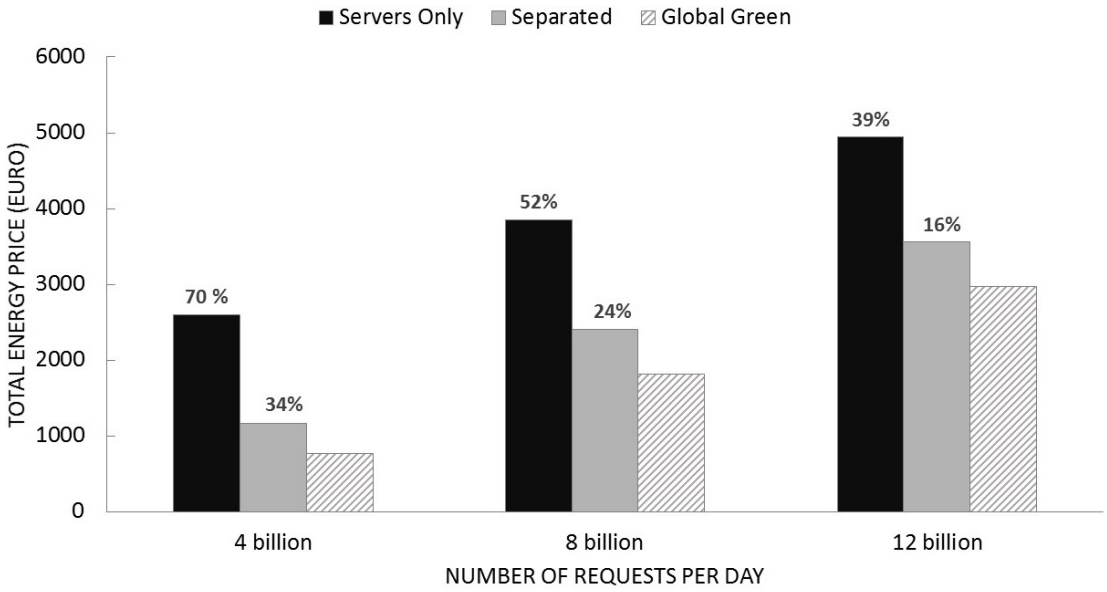

Figure 4.8: *Joint Optimization Cost Comparison*

for the optimization of the network. As expected, the savings tend to decrease as traffic increases since we are forced to keep more system nodes (servers and routers) active to accommodate a larger number of requests and we have less room (smaller space of admission solutions) for optimizing energy consumption.

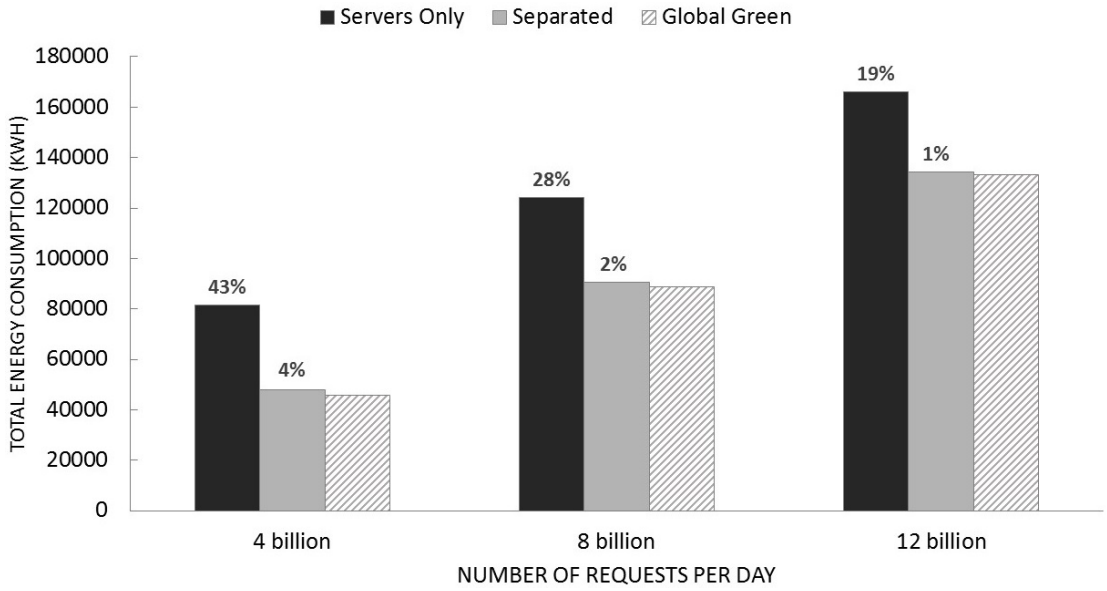

Figure 4.9: *Joint Optimization Energy Consumption Comparison*

In Figure 4.9 we plot the energy consumption of the same three scenarios. Obviously, we observe a significant saving of about 43% of the

joint model compared to the *Servers Only* case, because of the network energy consumption. On the other side, the energy consumption of the separate and joint models is comparable. The reason is that in the objective function we considered the energy costs rather than the energy consumption. Since the local generated green energy is available for free, the system tends to exploit it at best using also storage to better match production periods with consumption.

In order to better investigate the behavior of the proposed joint optimization model, we performed a series of other tests aimed at understanding the contribution of different system features like the geographical distribution of data centers with different energy prices and green energy availability. To this purpose we have considered three additional scenarios: the *Brown Base* scenario, where we neither exploit the geographical distribution of data centers nor the use of green resources, the *Green Base* scenario, where we use green resources locally without transferring load between data centers, and the *Global Brown* scenario, where we exploit traffic distribution among data centers but without using green energy generation. Note that in scenarios where green energy production is considered, we also use storage to optimize its use over time.

Before analyzing the results of the model, it is worth considering their computational complexity. Figure 4.10 shows the average execution time required by CPLEX to solve the models using different workload configurations. We observe that in all cases it is possible to solve the problem within a few minutes even for a very large number of user requests. For small instances, i.e. 2 Billion requests, the solver takes around 1 minute for the Global Green scenario, and half a minute for the Global brown scenario. While for the two base cases (Brown and Green), solution time is always less than half a second, even if the problems become unfeasible for instances with more than 16 billion requests, due to capacity limits of data centers servers. In the worst case, with a very high traffic compared to capacity (40 Billion requests per day), the solution of the joint optimization problem took 4.52 minutes, which is very good for a 24 hours time horizon of traffic planning.

Figure 4.11 shows the results obtained for energy costs using the scenarios mentioned above for different traffic levels. On top of each bar, we

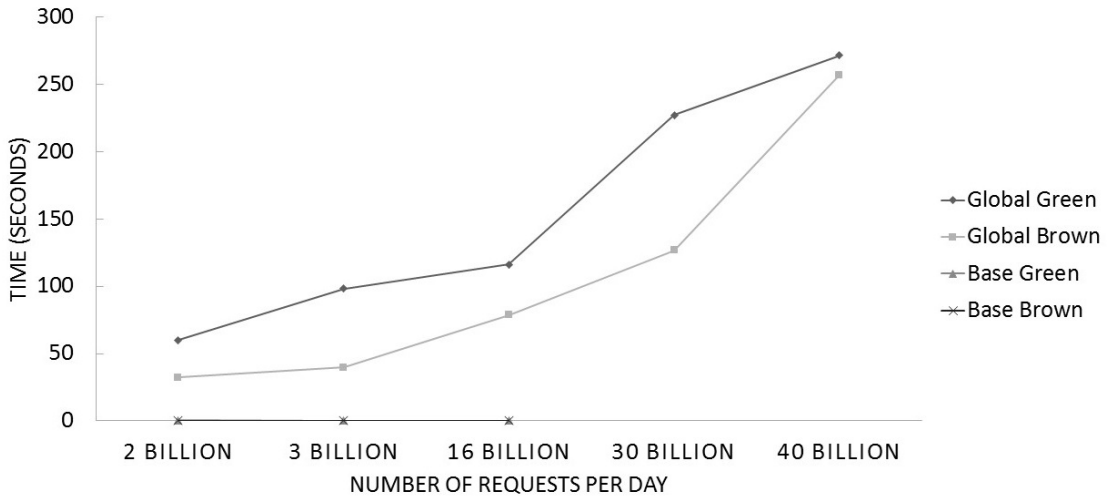

Figure 4.10: *Solving Time*

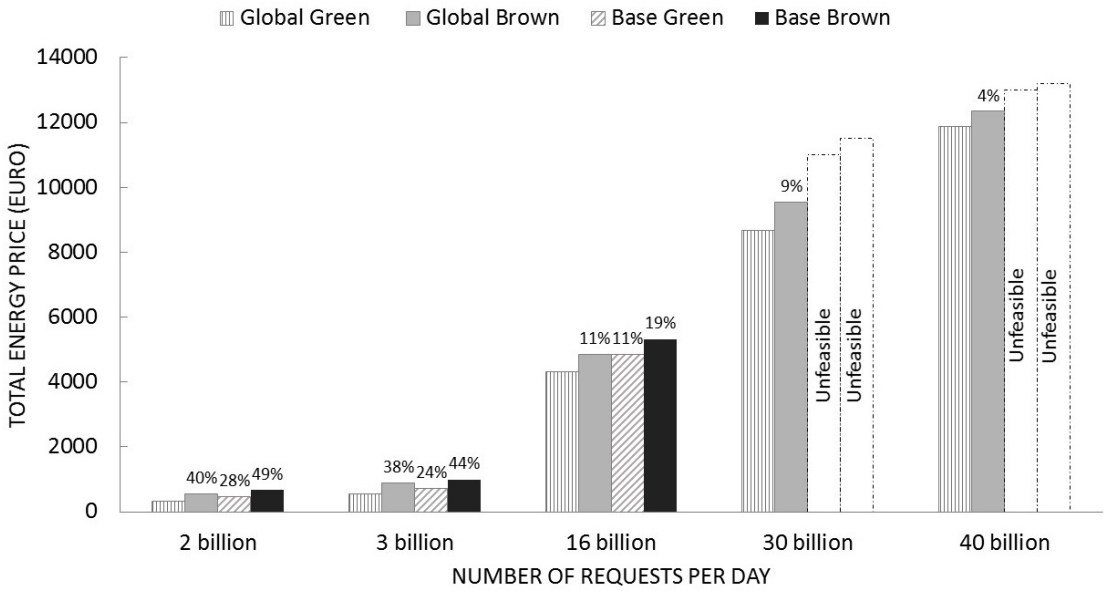

Figure 4.11: *Overall Cost Comparison*

indicate the percentages of savings of the Global Green model compared the one indicated by the bar. It can be easily noted how significant cost reduction is achieved through collaboration between data centers using VMs migration. Moreover, with the cooperative and jointly optimized schemes the available capacity of the Cloud system is higher than with the non-cooperative schemes, as for high load levels the Base Brown and Green models are not feasible. We notice also that using the cooperative model without green energy (*Global Brown*) still provides non-negligible savings compared with the non cooperative approach (*Base Brown*).

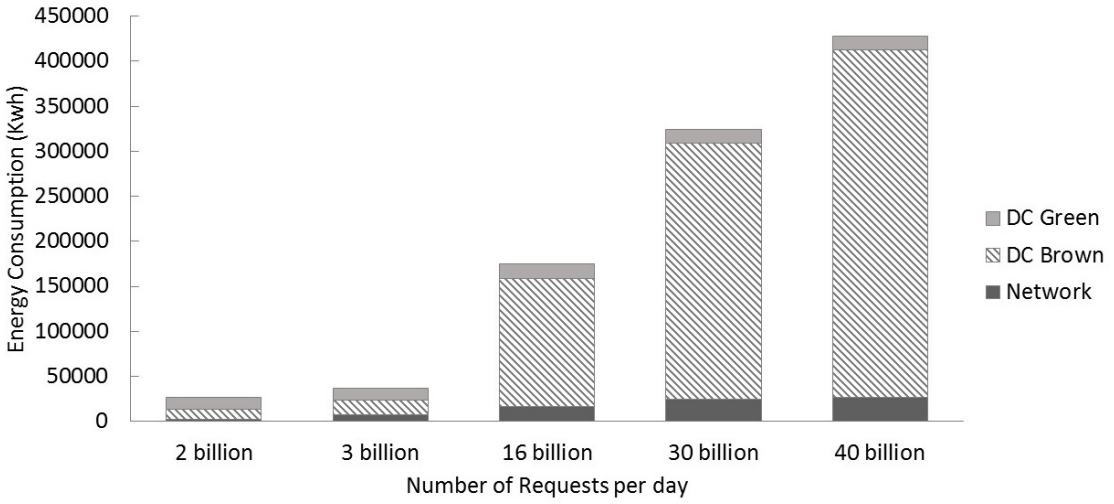

Figure 4.12: *Energy Consumption Split*

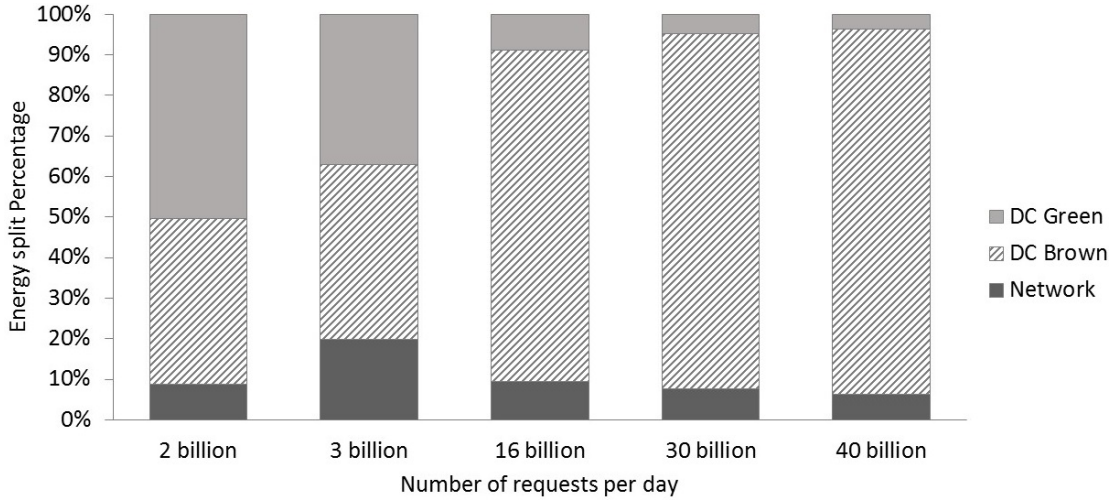

Figure 4.13: *Energy Split Percentage*

To better understand how the *Global Green* model uses energy in the system, we can analyze the split of energy use in Figures 4.12 and 4.13. The amount of green energy is limited by the capacity of generators considered in our instances, and it is a significant portion of the energy used only at low traffic levels, while it becomes rather small at high load. Taking a closer look to optimal solutions we notice that the system in different time periods tends to saturate the capacity of sites where energy is cheaper and green energy available for migrating VMs, and uses the other sites for the load exceeding capacity until the savings are significant compared to migration costs. As load increases, the capacity of cheap and green sites tends to be saturated by local demand and the cost

savings decrease since migration is used mainly for load balancing.

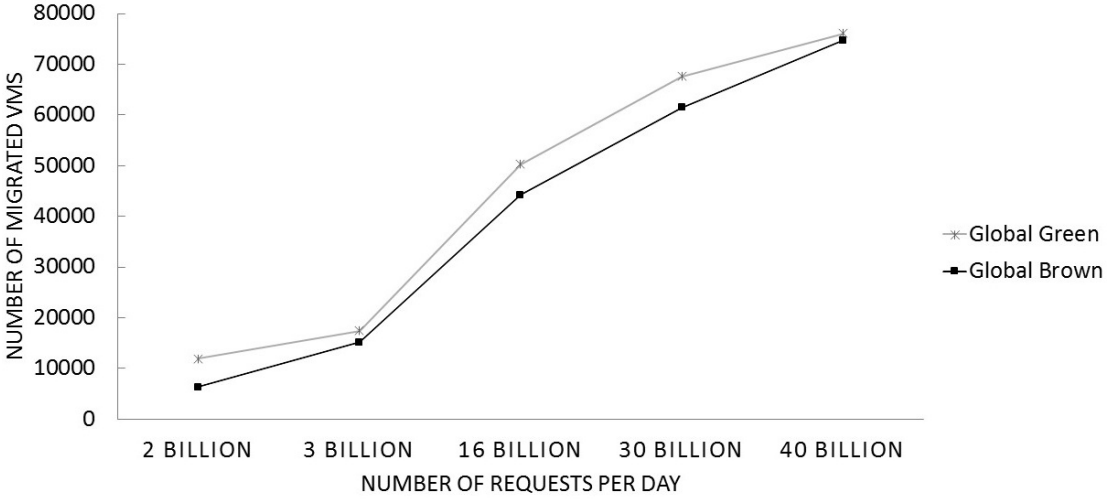

Figure 4.14: *Number of Virtual Machines Migrated*

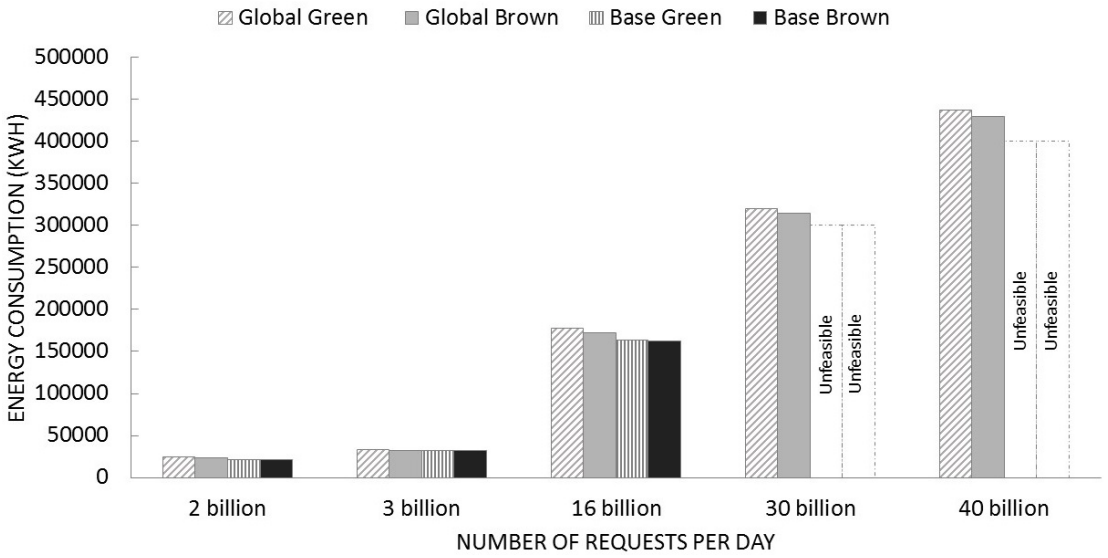

Figure 4.15: *Total Energy Consumption*

Figure 4.14, shows the number of migrated VMs in both *Global Green* and *Global Brown* scenarios. As it is expected, it is proportional to the number of received requests. Even though VM migration process itself costs energy, the overall cost saved is more significant. We can notice also that with the presence of green sources of energy more VMs are migrated to exploit it, unlike the brown case where we only benefits from the difference of prices between various locations.

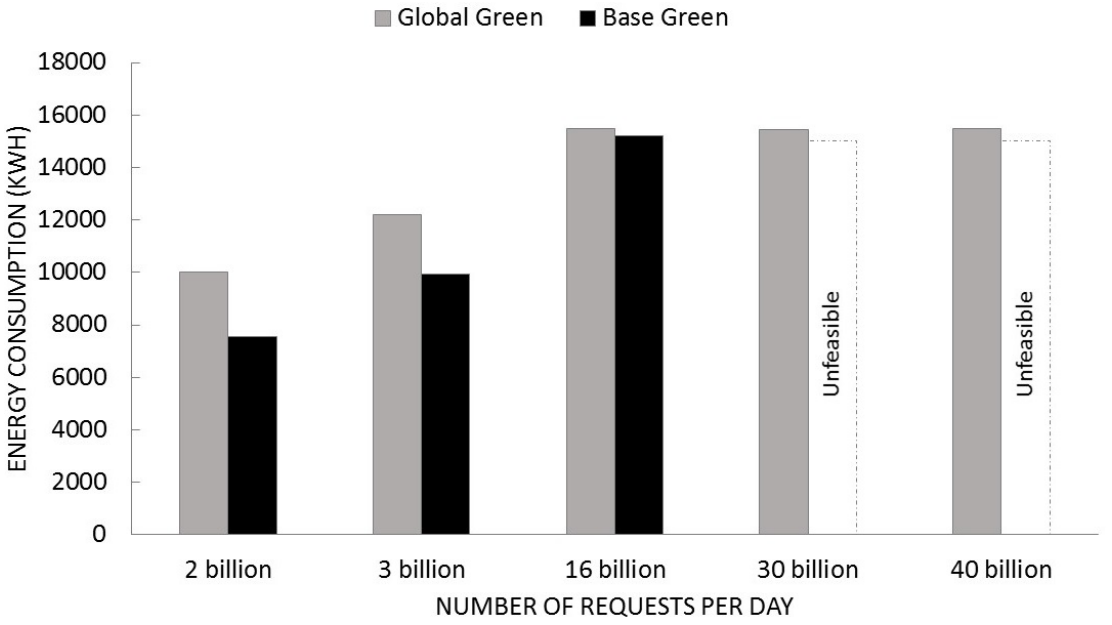

Figure 4.16: *Green Energy Usage*

While our model can achieve significant savings of the total power cost, it may cause consumption of larger amounts of energy as shows Figure 4.15. The reason is that we use beside data center servers, network devices for VMs load balancing and this consume more power. On the other hand, our model performs better in exploiting green energy by migrating VMs as reported in Figure 4.16. Therefore, even with the additional amount of energy that we consume, the proposed model is greener because it uses less brown sources of power by replacing them with green resources.

4.5 Green Energy Storage using batteries

In order to study the effect of batteries on energy consumption and expenses in Data centers, we simplify the model, in order to focus on the energetic side. In other words, instead of migrating VMs, we simply redirect user requests between Data centers. In this model, Data centers collaborate to optimize the redirection of users requests depending on energy prices in different locations, and on the capacity of Data Center servers. The objective in this modified model is to minimize the total energy operating costs by balancing the green energy produced in the Data centers between a direct supply for an instant usage, and the storage in

batteries in order to use it when green power resources are drought.

For this purpose, different battery types can be used, including: lead-acid batteries, Li-ion (lithium-ion) batteries, Sodium-nickel (zebra) batteries, Sodium-Sulfur (Nas) batteries, and many others [150]. The choice of the battery type differs depending on the performance needed and on purchasing power of the company.

Lead-acid batteries were used in Data Centers as a UPS (Uninterruptible Power Supply) where they are required to accomplish only few cycles during the year (2 to 3 complete charge/discharge cycles). Their price is very cheap compared to other types of batteries (less than 300 $/kwh) [63] which nominates them to be used for green energy storage, but the problem is their short life. A Lead-acid battery can achieve 800 to 1000 charge/discharge cycles, which limits their use in applications where charging and discharging is frequent.

Li-ion, Zebra and Nas batteries, are more suitable for green energy storage, their lifespan is longer than Lead-acid batteries. The life duration of this type of batteries ranges from 2000 up to 5000 complete charge/discharge cycles. Their disadvantage is their high price, i.e. Li-ion battery nowadays is around 900-1300$/kWh [63] but it is expected to become cheaper in the next few years.

Charging and discharging rates determine the efficiency of the battery, where the less energy loss during charging and discharging operations, the more efficiency of batteries. In case of Li-ion batteries, they have a high efficiency which is from 85% up to 95%, while for example Lead-acid batteries efficiency is from 70% to 90% [150].

Another important factor to consider in defining batteries efficiency is energy loss rate per time or self-discharge. Batteries lose a percentage of their charge due to internal chemical reactions that occur over time. The self-discharge of the same type of batteries may differ under different temperatures and storage climates. Multiple battery technologies have different self-discharge rates, for example, Lead-acid batteries may lose 0.1% to 0.3% during one day, while Li-ion batteries lose around 1% to 2% during a month [150]. The term C-rate in Batteries is used to indicate charging or discharging rate, which is equal to the capacity of a battery in one hour of time.

4.5.1 Modified formulation

To simplify the objective function, we introduce this two new variables. V_i^t present all the energy consumed by a Virtual Machine i at time t, while O_{ij}^t is the energy consumed by networking operations on a link (i, j). As we replace both variables $\overline{w}_{il}^t$ (number of VMs before migration) and $\underline{w}_{il}^t$ (number of VMs after migration), by only one variable w_{il}^t which indicate the number of type l VMs running in DC i in time t. Therefore, the new objective function is formulated as follows:

$$\text{Min} \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{I}} \left\{ M_i^t \left[\rho_i V_i^t - g_i^t \right] \right\} + \sum_{t\in\mathcal{T}} \sum_{i,j\in\mathcal{I}} E_{ij}^t R_{ij} O_{ij}^t \tag{4.31}$$

Similarly to the previous objective function, we try to minimize the total energy cost in 4.31. This function is divided in two parts, firstly, we calculate the energy expanses of VMs taking in consideration the renewable power usage, and secondly, we calculate the network energy expanses.

The cost of buying energy from the electrical grid is multiplied by the total energy consumed by VMs minus the green power used in each time band. Then, the subtracted green energy is the sum of the energy coming from batteries with respect to discharging efficiency, plus the energy coming directly from the power generator.

Concerning the network, similarly to the previous model, the price is multiplied by the sum of energy of all links in all time bands, which depends on the number of routers in each link and the cost of energy in that link. We assume that network use only brown power.

Regarding the constraints, 4.32, 4.33 present the new decision variables.

$$V_i^t = \sum_{l\in\mathcal{L}} (\alpha_{il} w_{il}^t + \eta_{il} won_{il}^t + \theta_{il} woff_{il}^t) \quad \forall i \in \mathcal{I}, \forall t \in \mathcal{T} \tag{4.32}$$

$$\begin{aligned} O_{ij}^t = (\gamma_{ij} - \delta_{ij}) \frac{b_{ij}^t}{Q_{ij}} + \delta_{ij} z_{ij}^t + \tau_{ij} zon_{ij}^t + \xi_{ij} zoff_{il}^t \\ \forall i \in \mathcal{I}, \forall j \in \mathcal{I}, \forall t \in \mathcal{T} \end{aligned} \tag{4.33}$$

Concerning Data centers specifications the next constraints are used:

$$\sum_{l \in \mathcal{L}} \sum_{j \in \mathcal{I}} x^t_{ijkl} = \lambda^t_{ik} \quad \forall i \in \mathcal{I}, \forall k \in \mathcal{K}, \forall t \in \mathcal{T} \tag{4.34}$$

$$x^t_{ijkl} \leq \lambda^t_{ik} m_{kl} \quad \forall i, j \in \mathcal{I}, \forall l \in \mathcal{L}, \forall k \in \mathcal{K}, \forall t \in \mathcal{T} \tag{4.35}$$

$$w^t_{il} = \sum_{j \in \mathcal{I}} \sum_{k \in \mathcal{K}} \frac{x^t_{ijkl}}{\mu_l} \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{4.36}$$

$$w^t_{il} \leq P_{il} \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{4.37}$$

Constraint (4.34) ensures that all user requests are served by the Cloud. Furthermore we use (4.35) to make sure that each type of requests is served by the required type of VMs. Constraints (4.36) and (4.37) defines the number of active VMs at time t and makes sure this number do not exceed the capacity of the DC.

$$won^t_{il} \geq w^t_{il} - w^{t-1}_{il} \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{4.38}$$

$$woff^t_{il} \geq w^{t-1}_{il} - w^t_{il} \quad \forall i \in \mathcal{I}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{4.39}$$

Constraints (4.38) and (4.39) are used to determine the number of VMs to be turned on and off at the beginning of time period t, according to their number at the end of time period $t-1$.

Regarding the network, the following constraint is used to define the amount of used bandwidth during t:

$$b^t_{ij} = \sum_{k \in \mathcal{K}} \big(B_k \sum_{l \in \mathcal{L}} x^t_{ijkl}\big); \forall i \in \mathcal{I}, \forall j \in \mathcal{I}, \forall t \in \mathcal{T} \tag{4.40}$$

The rest of constraints are the same used in the previous model which are: (4.14), (4.15), (4.16), (4.17), (4.18), (4.19), (4.20), (4.21), (4.22), (4.23), (4.24), (4.25), (4.26), (4.27), (4.28), (4.29) and (4.30).

4.5.2 Experimental Tests

For evaluating the proposed model and to show the influence of using batteries on Cloud Data Centers energy consumption and expanses, we

performed a set of tests with different instances, and under a variety of workload volumes and batteries capacities.

The input parameters are the same used in the previous version, the only difference is the number of considered Data centers. Inspiring from Google Data Centers infrastructure, we use 36 Data Centers geographically distributed in the world [14].

Google data centers are considered among the most effective in the word, their power usage efficiency (PUE) has been improved significantly since Google started reporting the number in 2008. In 2014 the average of all Google data center is 1.12 [18]. If we consider other Data centers of another company different than Google, the PUE is around 1.7 as reported in [22]. Therefor we will varies the values of PUE between 1.1 and 1.7.

4.5.3 Results and discussion

In order to run the proposed model, we have used a CPLEX 12.2 as a MILP optimization solver. All tests were run on an Intel Nehalem dual socket quad-core CPUs @2.4 GHz with 24 GB of RAM running Ubuntu Server Linux 2011.4.

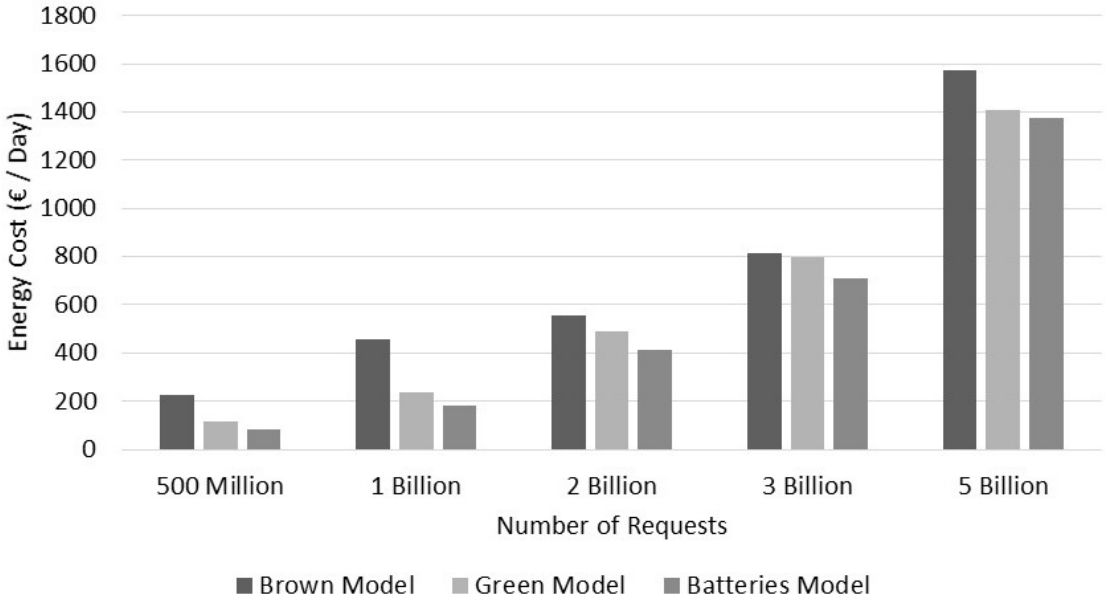

Figure 4.17: *Energy Costs Comparison*

Figure Fig. 4.17 shows the costs savings that we can achieve using batteries with a good policy of management. This figure presents an

overview of how three different models behave under various workloads. In the *Brown model* we did not consider any use of green energy resources. For the *Green model*, we used green energy without any storage, while in the *Batteries model* we used a 150 kwh battery. The graph reports the model objective function values, consisting of the overall daily cost, given by both Data Center and network costs. The differences of cost between this models is decreasing by incrementing the number of requests because the amount of green energy used in all the scenarios is fixed.

It can be easily noted how the use of batteries allows significant cost reduction in every test, up to 35% comparing to the *Green Model* and up to 64.5% compared to the *Brown model*. Furthermore, Data Centers, by using the green energy in the appropriate time, reduce their need to redirect requests to each other, which decrease the use of the network so saving more energy. Alongside, another an important behavior can be observed which is network energy consumption in each scenario, as it is shown in figure Fig. 4.18. We remark that by using batteries Data Centers use less Network operation, which means more autonomy comparing to the other approaches. Indicating that this significant difference is achieved by considering that the network workload is only the redirected requests between Data Centers and not the related users traffic produced due to Data Upload or Download.

More details about batteries model are shown in figure Fig. 4.19. We observe that the system need to store more energy by augmenting the number of generated requests which shows the importance of batteries for energy storage in Data Centers.

Since the capacity of the battery is very critical, we have performed a set of tests using different battery capacities, the obtained results are shown in figure Fig. 4.20. The total energy expenses savings augment by increasing the capacity of the used batteries. Owning more capacity lead to more savings in energy expenses without considering variations in the amount of green power generated which have a very important role also.

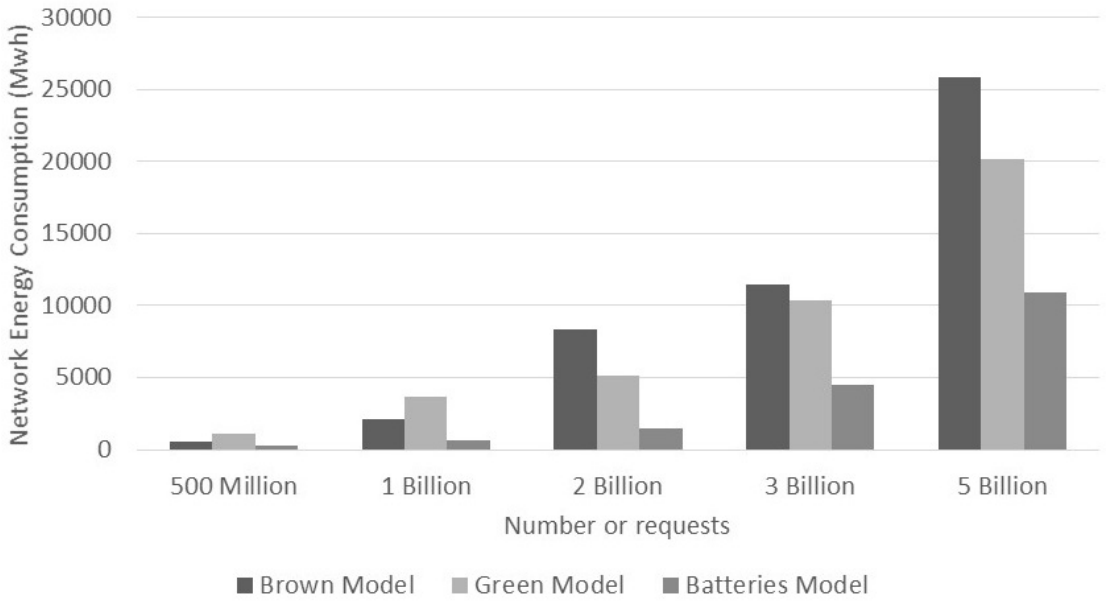

Figure 4.18: *Networks Energy Consumption Comparison*

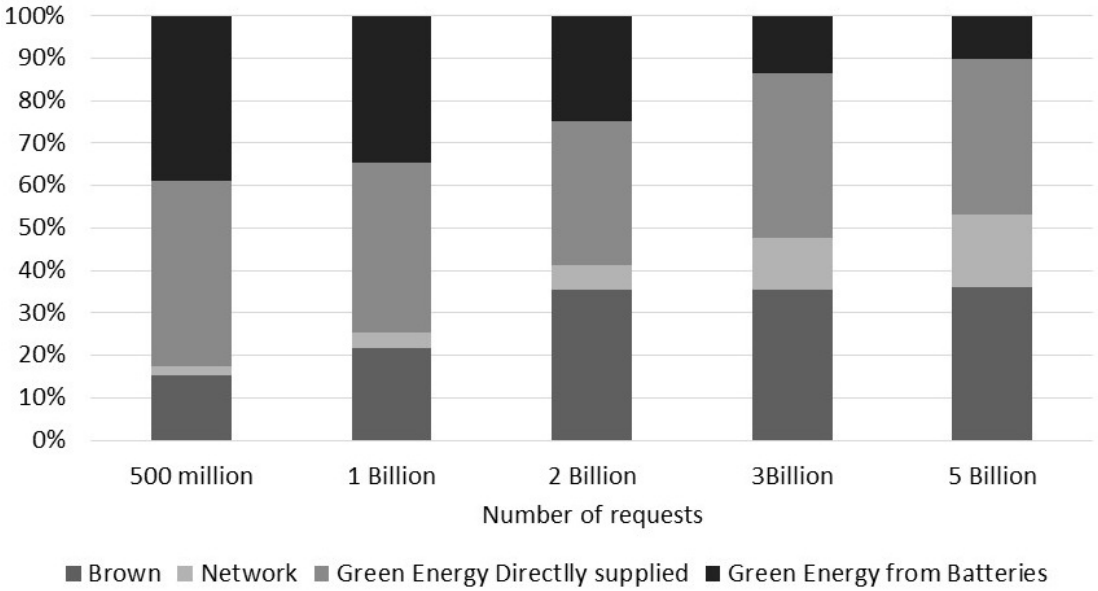

Figure 4.19: *Batteries Model Energy Split Percentage*

4.6 Conclusion

Most of existing work on energy optimization in Cloud systems manages separately data center servers and their interconnection network. In this chapter we presented a new optimization framework based on MILP for jointly management of Cloud data centers and their network.

The proposed model considers a set of data centers geographically

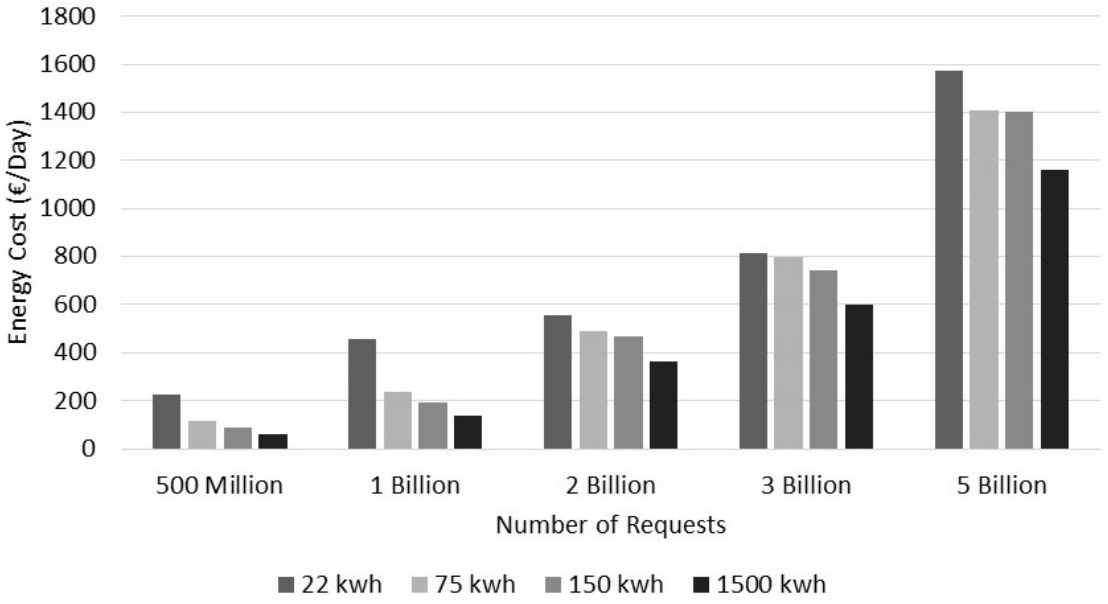

Figure 4.20: *Different Batteries Capacities*

distributed over different locations around the world. Data centers collaborate by migrating VMs between them when necessary to exploit different energy prices in various time zones. Another factor that we consider is the availability of green energy resources in some data centers and the possibility to store this energy using batteries.

In Cloud scenarios, migrating VMs between different sites needs additional network resources due to the size of VMs themselves and their data. Besides managing both data centers and their network, we also manage both the use of brown and green energies. Our strategy consists on redirecting the load to sites with more available green energy. We suppose also that some data centers are able to store the generated energy for later use, therefore we can save the clean power to use it when its generation is not possible or during peak energy price periods, thus we solve the problem of discontinuity of the renewable resources.

We show that the proposed optimization model can be solved using a state of the art MILP solver (CPLEX) in a reasonable time even for big size instances. The obtained results are very promising and shows that our approach allows significant cost saving compared to the base scenarios used nowadays. Moreover, from an environmental point of view, our model reduces greenhouse gas emission by pushing the Cloud to use more green power resources, alongside with optimizing its use in

each data center using local energy storage.

Finally, we discussed green energy storage in Data Centers, more specifically using batteries. We presented multiple types of batteries which are can be used in this case, while highlighting the most important characteristics of a battery. Then we modified the proposed framework to analyze the impact of batteries. The model was tested on a set of realistic data of batteries as well as Data Centers and Networks. The numerical results have shown the validity of our proposed model in managing effectively the available green power, as well as the significant importance of using batteries in Data Centers.

CHAPTER 5

Joint Planning and Energy Management of Cloud Radio Access Networks

5.1 Introduction

With the arrival of mobile Internet and the rapid growth of data traffic, the traditional architecture of the access network in telecommunications networks is struggling to deal with the dramatic increase of users demands over the last decade. The idea of installing any other base station is no longer satisfying for many reasons, including the fact that revenues are not progressing same as construction costs, new difficulties of planning, and the increase of energy consumption.

To overcome these problems, operators need a cost-effective way to combine several data standards (LTE, GSM and WiFi ...), frequency bands and transport network solutions, in addition of reducing latencies and the rate of data manipulation. This means that, above all, cellular infrastructure must be flexible and can support a deployment and management of simplified heterogeneous radio access networks, which is the part of the system that connects users to the service provider.

Cloud Computing has been widely adopted in many fields. In a different context than service based data centers considered in the previous

chapter, the concept of Cloud was utilized in mobile access networks to overcome the upper-mentioned problems, which gave birth to a new paradigm named Cloud Radio Access Network or C-RAN.

In this chapter, we propose an efficient method to minimize energy consumption toward a green C-RAN deployment. The proposed approach tackle the problem from a different perspective comparing to existing work in the state of the art. We precede the energy hungry behavior of the network by introducing power management at the first stage of its lifespan which is the planning stage. Before going through the details of the proposed approach, we first discuss C-RAN architecture, technology and its related issues, and we address energy efficiency of this specific type of Cloud systems.

5.1.1 What is C-RAN?

C-RAN is a novel type of mobile network architecture to improve the systems currently used in mobile communications. The letter C can have many interpretations: *Centralized, Clean, Collaborative or Cooperative*, but most commonly it stands for *Cloud*. The concept was proposed first time in [104] by IBM under the name Wireless Network Cloud (WNC), then described in details by China Mobile in [112]. It derives from the traditional Radio Access Network (RAN), which were built with multiple base stations (BTS) across a region. A BTS (eNodeB), covers a small area at a time, and the system includes everything that is needed for wireless communications, GSM, 2G to LTE.

The main problem of a traditional RAN is its cost. First, there are the costs associated with the capital called CAPEX (Capital Expenditure); which are investment spending. Indeed, as the BTS must have all the necessary functionality to manage small parts of a network, it must therefore procure own equipment, have all permits, and engineering fees for designing the network. The need for dedicated hardware occurs especially at the level of the treatment of information from and to the antennas.

The need for dedicated hardware can also result from the need to support the decoding and encoding functions. Companies like Alcatel Lucent with the *Light Radio* and Nokia Siemens Networks with *Liquid Radio*, offer equipment dedicated for the treatment of the information

from the Remote Radio Head (RRH). Within the RAN base stations, we find general processors that perform the rest of the treatments, but some functions are accelerated with a dedicated material.

Then, there are operating costs commonly named OPEX (Operating Expenditures), i.e. energy costs, cooling systems, rental or purchase of many locations to be able to install the stations, and also update devices when a new service appears. RAN platforms are less flexible due to their architecture, and their energy demand can hardly be managed [112]. Through 24 hours, the traffic during day time is more intense than during the night. It is difficult to lower energy consumption, since disabling a BTS in full is not an option because we lose the coverage. Therefore, we need an alternative that meets the energy demands, and allows a dynamic management of resources within a base station. Hence, it is necessary to centralize the RAN to avoid this problems, which is the idea of C-RAN.

Unlike the traditional RAN, in C-RAN, the BTS which is mainly composed by the Remote Radio Heads (RRHs) and Baseband Units (BBUs) is distributed. BBUs are centralized in a virtualized pool, and shared among RRHs located in different sites. This pooling makes the BTS more efficient since fewer BBUs are needed for processing compared to the traditional RAN architecture. It has the potential to decrease the energy consumption, as well it eases the scalability and network maintenance.

5.1.2 C-RAN Architecture

In the 1G and 2G of mobile networks deployment, baseband and radio processing (which are the first sub-functions of processing the information received by an antenna) are done inside the base station. The radio module is located just a few meters from the antenna as it shows figure Fig. 5.1

In the third generation of mobile networks (3G), a new architecture was introduced. The base station is separated into a signal processing unit called BBU (or Data Unit DU) and a radio unit called RRH (also named RRU standing for Remote Radio Unit). Basically, some functions were placed closer to the antenna, which are: power amplification and filtering, digital to analog and analog to digital conversions and digital processing [96]. By doing so, the BBU containing the rest of func-

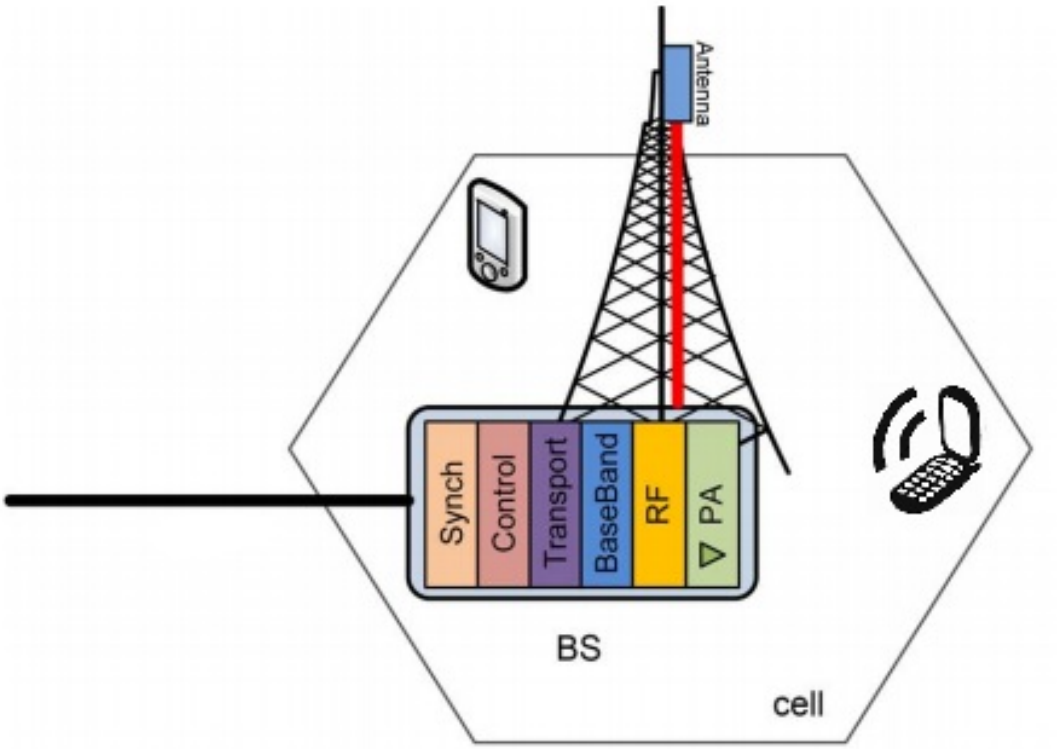

Figure 5.1: *Classical RAN Architecture*

tionalities, can be placed in a better location, where installation is easier and renting or purchasing prices of the site are more convenient. For connecting RRHs to its BBU, generally, optical fiber is used. Due to the limitation of processing and propagation delay, the distance between the two units cannot exceed 40km [52]. Figure Fig. 5.2 [52] explains more this architecture.

C-RAN is a natural evolution of the RRH based architecture. Different BBUs are collected into one site named Pool or Hotel. It consists of a virtualized server or data center that performs information processing. Depending on the function splitting between RRHs and BBU pool, there are two types of C-RAN architecture: Fully centralized and Partially centralized. The main difference between this two architectures is the position of the first layer which is baseband processing. C-RAN is illustrated in Fig. 5.3 [52].

Fully centralized C-RAN: in this architecture, the baseband processing is placed in the virtualized BS pool, as shown in figure It allows better resource sharing and eases the collaboration between multiple cells. It is

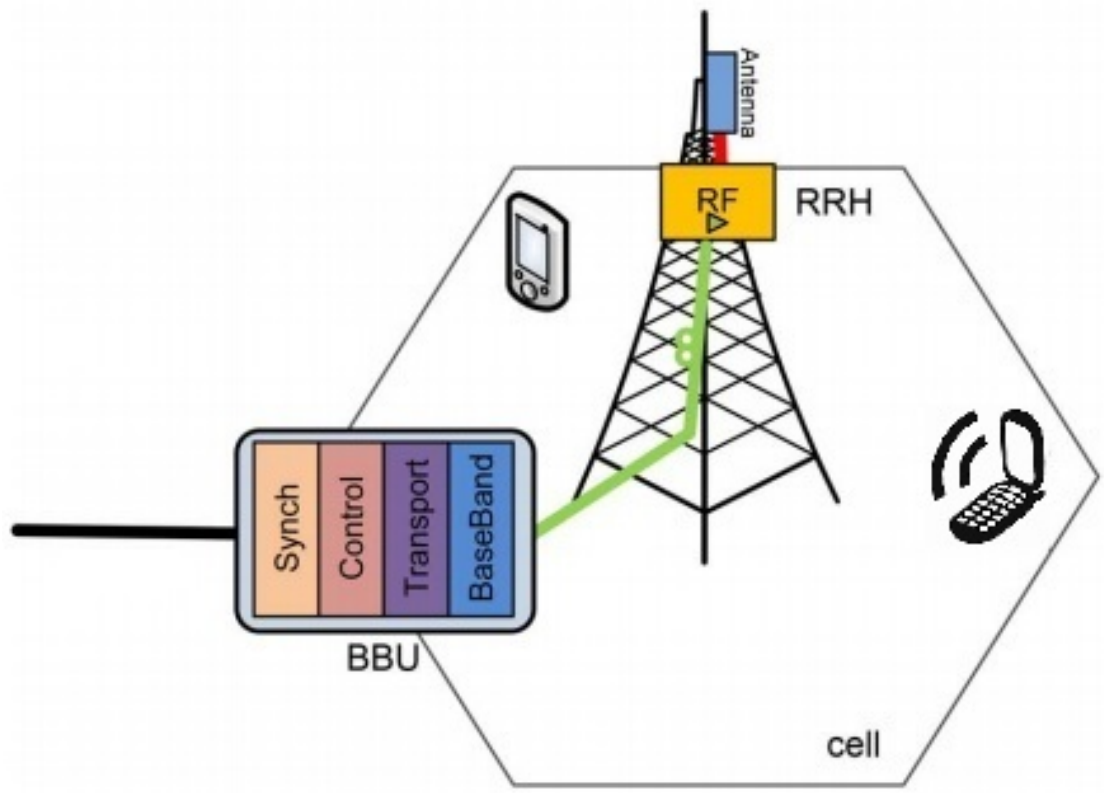

Figure 5.2: *3G RAN Architecture*

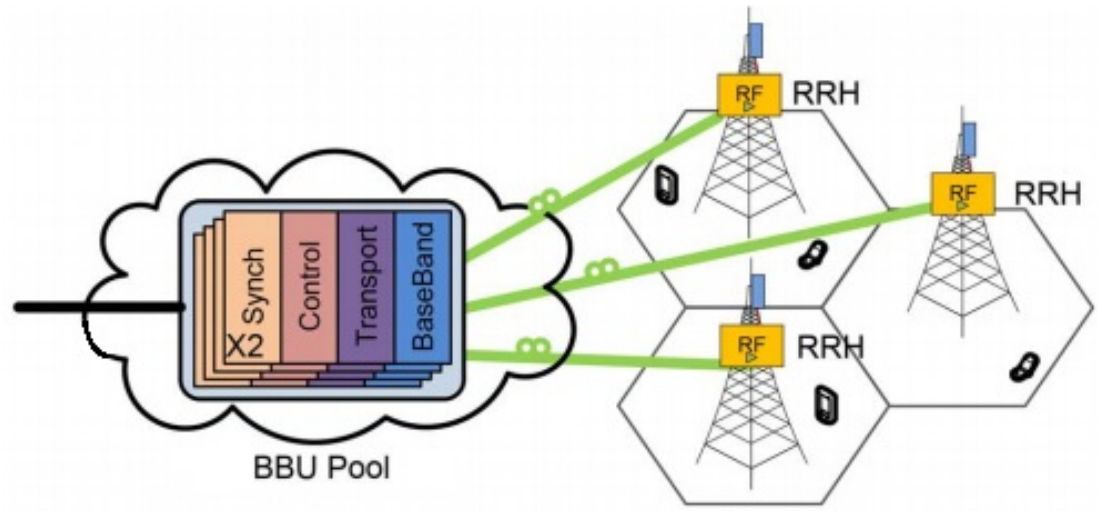

Figure 5.3: *C-RAN Architecture*

also easy to upgrade and increase the network capacity. The main problem is the high communication between a RRH and the BBU pool to carry the baseband signal, which require a high bandwidth in the transport link. Figure Fig. 5.4 [52] explains this architecture .

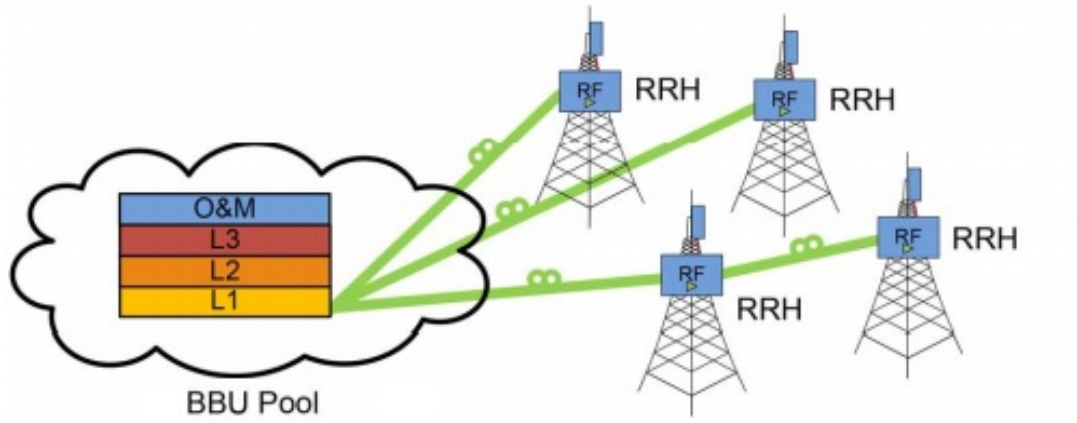

Figure 5.4: *Fully Centralized Architecture*

Partially centralized C-RAN: this architecture was introduced to avoid the high transmission between RRH and BBU. The baseband processing layer is integrated into RRH instead of the BBU as shown in figure , therefore, the transmissions are only the demodulated data, which can be 50 times less in volume compared to the original signal. Its problem is the difficulty of multiple cells collaboration, as it is less convenient in upgrading. Figure Fig. 5.5 [52] explains this architecture.

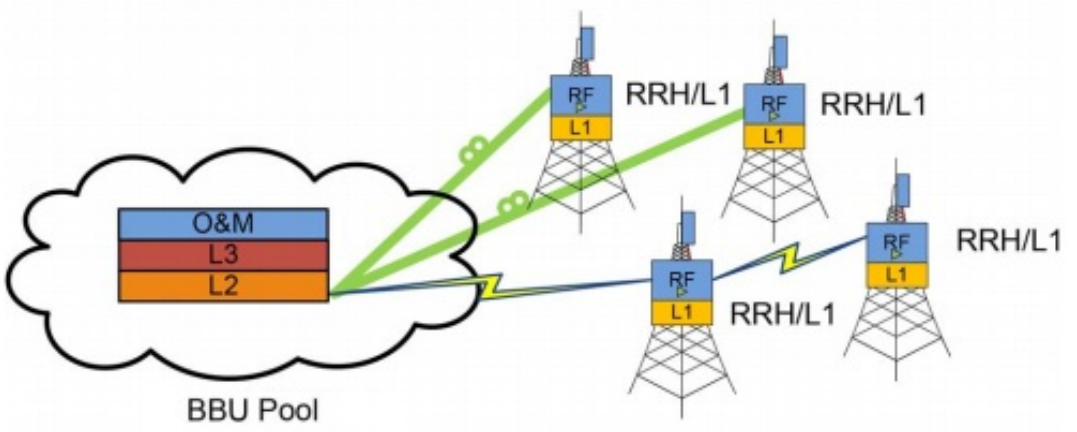

Figure 5.5: *Partially Centralized Architecture*

5.1.3 Virtualization in C-RAN

RAN virtualization is one of the most challenging cases of virtualization to do since the system has to respond in real time to the radio frequency signal, and to deal with the dynamic changes of the cell load. In a vir-

tualized RAN, a part of the base band will be replaced by a software running in the data center. Basically, it is about how much baseband we can virtualize in the cloud data center, and what are the physical elements that we have to leave in the radio head. The more real time functions we leave on the radio head the better performance will be our system. Virtualization allows to introduce a good level of flexibility and scalability. However, the quality and performance of the system relies heavily on the bandwidth and latency offered by the link used for transmission between the BBU pool and the RRH.

In C-RAN functionalities of the traditional base station are implemented as software and named Virtual Base Station (VBS). Several VBSs run within an operating system. Different VBSs running on the same hardware can be shared by multiple operators, which allows offering RAN as an infrastructure cloud service (RANaaS) [125]. By using RAN as a cloud service, operators reduce their CAPEX as well as OPEX since management and maintenance of the infrastructure will be the responsibility of the service provider.

Virtualization solutions available today in cloud computing are well defined. However, C-RAN have different requirements than a normal cloud, especially in terms of the time related to transported data (Latency, jitter, data rate ...). Therefore, the available virtualization solutions could not be the best choice, which urges to develop a dedicated solution. In collaboration with China Mobile, Intel has proposed a prototype of a virtualized BBU Pool based on Intel Xeon processors and processing TD-LTE signals [127]. As well as many other solutions were discussed in the literature [28, 29, 62, 157].

5.1.4 Advantages of C-RAN

Costs Minimization: The main advantage of C-RAN is that it reduces costs. The centralization of equipments in the BBU pool allows savings on both CAPEX and OPEX. Centralizing BBUs in a pools makes it much easier to install and more effective in management and maintenance compared to the traditional RAN. In the other side, RRH is simpler which decrease its size, so that they can be installed in buildings with a minimum of management. Thus, Operators can have great savings on: the rental of installation sites, management and maintenance of the sys-

tem, leading to lower the overall cost.

Lower energy consumption: Employing C-RAN offers potential savings on energy consumption. Inside the BBUs pool, many techniques of energy management can be applied. The centralization allows to turn off some BBUs to save power when traffic is low especially during the night, and without having any impact on the network coverage. In addition, RRHs can be naturally cooled by placing them on rooftops of houses and building walls, which leverages efficient savings on air-conditioning equipments. Some researches realized by China Mobile claim that reduction in energy consumption in C-RAN networks can be up to 71% comparing to the traditional RAN [54].

Improving spectral efficiency: Within the same pool, BBUs communicate with each other with a very high speed and low latency using Cooperative techniques, such as CoMP [92] which minimizes inter cell interference by avoiding them, and eICIC [64] which relies on turning the interference into a useful signal.

Network scalability: In order to respond to the growing number of users or to improve network coverage, instead of installing a whole new base station operators have only to add new RRHs and connect them to the pool. However, to meet capacity demands due to the increase in the number of mobile users, BBUs pool can be upgraded by adding new base band units, or increasing servers' capacity, depending on the type of system (Only centralized RAN or a Virtualized RAN).

5.1.5 Challenges of C-RAN

C-RAN brings a lot of benefits compared to the traditional RAN, however, there are also many technical challenges that need to be addressed before its deployment by mobile operators. Beside the virtualization issues discussed before, in the following we site some of the other challenges:

High performance and low cost transport network:

In the traditional RAN architecture, BBU and RRH are co-located at the same site, in which, configuration of the latency budget is easily maintained because of the small distance between this two units. In C-RAN, BBU and RRH are geographically separated, which results in extending the path in which they communicate, therefore, latency begins

to grow to a point where it needs to be maintained in order to maintain the performance of the network.

BBUs Cooperation:

Many issues need to be addressed for collaboration of BBUs including: security, resilience and reliability of interconnection. Joint processing allows to introduce coordinated multi-point which requires a centralized front-haul solution. Thus, a low latency, high bandwidth and extensibility in designing the topology are key points to be considered. However, efficient Cooperative processing algorithms that are based on the use of special channel information and guarantee also the cooperation among multiple RRHs in different locations should be developed [54].

Network Planing:

Planning of C-RAN networks is a complex problem to be solved. It consists of designing the infrastructure to meet the demands of customers spread over a geographical area. More precisely, it is about how many and where to place RRH and BBU pools within a given area. This problem is known in GSM and UMTS networks as *Automatic Cell Planning* [89], as we can find many other similar problems (with or without association of clients) such as the Facility Location Problem [68], the p-median problem [122], or the Set Covering Problem [46] which are NP-hard problems.

Energy Management:

C-RAN have the potential to reduce the energy consumption of mobile networks as it is expected to do. It is considered as a promising technology that can reduce both OPEX and CO2 emissions based on cloud computing and virtualization technology. Providing better performance compared to the classical RAN, C-RAN is easier to manage and can reduce power consumption if techniques such as load balancing and dynamic resource allocation are efficiently applied. However, in the following we will address this problem and discuss it in more details.

5.2 State of the Art

In mobile networks, the increases in energy consumption are especially remarkable in access networks due to the fact that more than half of consumption comes from this network section [59]. Two main reasons

explain this significant proportion of energy consumption in access networks: firstly, the ineffectiveness of the internal components of actual base station, especially, the power amplifier (PA) section that consumes about a 60-70% of the total energy consumed by the BS [61], which is also accompanied by the need to keep equipment of cooling continuously active to reduce system heating. Secondly, due to ongoing fluctuations of traffic, there is an inefficient use of the actual systems of base stations, having across the entire infrastructure of radio active on an ongoing basis. Based on [41], for a typical cell deployment, around 50% of the traffic on the network is handled by 10% of base stations, while about half of the remaining base stations are responsible for only 5% of the traffic. This problem is expected to be solved in C-RAN taking advantage of its architecture.

A lot of work has been done towards an energy efficient RAN. Ranging from efficient hardware design to equipments sleeping [121, 149]. However, since the architecture of C-RAN is different, many solutions were adapted or proposed to accommodate its new constraints.

We divide the existing work on energy efficient C-RAN in three types. Those who are focused on the BBUs pool side, those who are focused on RRHs, and joint solutions that consider the whole network.

Regarding work on BBUs pool, the authors in [134], address the energy efficiency of C-RAN for 5G networks by reducing the number of servers used in the BBUs pool. Their idea is to find the best matching between the baseband processing and traffic load with the servers running at their peak utilization. The problem was solved as bin packing problem, where the traffic have to be packed into the BBU pool servers. However, this solutions is more effective for low traffic periods, but when the load is high, it seems to be less efficient, as its power consumption tends to be close to the case of distributed BS architecture.

In [152], authors propose an energy efficient resource allocation approach for the computation tasks in C-RAN. The idea is based on migrating computation processes of smart mobile devices to the cloud due to processing and energy limitations. The problem is divided to two main subproblems, the first is to decide whether or not to offload the computation to the Cloud. The second is to allocate the resources for the computation tasks while considering latency constraints. The problem

was simplified to a knapsack problem then solved using a greedy algorithm. Even if the proposed idea is interesting, but the authors did not make enough tests to validate their solution.

For a dynamic allocation of BBUs to RRHs, Khan et al. in [98] propose a dynamic resource allocation scheme that depends on the traffic conditions. To reduce energy consumption, BBUs balance their state between switched ON and OFF depending on the traffic and resources usage. However, the obtained results showed that significant energy savings can be achieved using C-RAN compared to the traditional RAN.

Many similar approaches are available in literature [99,105,114,154]. However, in these solutions, the authors focus more on the BBUs pool side and do not include energy management for RRHs which could have significant changes on energy saving.

Regarding RRHs side, authors in [56] tried to introduce C-RAN architecture into Heterogeneous Networks (HetNet) to demonstrate both energy efficiency and spectral efficiency of this technology. They proposed a pre-coding antenna beamforming scheme that aims to reduce energy consumption of computations related to cooperation. While they base on the cooperative transmission among different RRHs for more efficiency of the spectrum.

In [155] a power saving for C-RAN is proposed based on RRHs selection and traffic density. In this work, energy consumption of the transport network between RRHs and the pool is considered. The authors formulate the problem as an NP-hard optimization problem, then they developed a local search algorithm that includes three local improvement operation to find out quickly optimal solutions.

Authors in [144] propose a dynamic frequency reuse scheme based on a mitigation of inter-cell interference technique named fractional frequency reuse (FFR). Graph coloring was used to allocate the spectrum frequencies among cells. This solution aims to reduce energy consumption through improving of spectrum utilization, which leads to less load on BBUs pool and so less energy consumption.

All of these approaches are either focused on RRHs side or BBUs side. However, some solutions try to jointly minimize energy consumption considering the whole C-RAN network.

Checko et al. in [53] try to prove that the implementation of C-RAN

architecture could be energy efficient compared to the traditional RAN. They used OPNET modeler and a real scenario for mobile traffic for implementation. The obtained results show that C-RAN allows reducing of signal processing resources around 4 times less than the traditional RAN, which reduce power consumption as well as CAPEX and OPEX.

Authors in [82] propose a joint optimization model that minimizes energy consumption taking into consideration both pre-coding design and allocation of RRHs, and BBU processing resources. The proposed algorithm tries to respect QoS required by users and fronthaul constraints.

In [115] a semi-static BBU-RRH switching approach is proposed. The scheme tries to determine the best combinations of BBUs to RRHs to deal with high traffic load during peak hours. Results obtained through simulations show that the proposed method can reduce the number of running BBUs which is an effective way to reduce power consumption.

However, none of the above solutions consider an optimal deployment that takes into consideration energy management at planning stage as a way to reduce energy consumption and OPEX, which is the solution that we try to develop in the following sections.

5.3 Green Planning of C-RAN

5.3.1 Model description

Energy consumption of radio access network varies depending on the type and radio coverage of the deployed cells. Installing macro-cells is cheaper and may be easier to configure, but then, applying an energy saving policy is difficult, since turning off some RRHs or decrease their emission power to save energy have significant impact on the size of the covered area. On the other side, small cells are easier to manage and may be more energy efficient, at the expense of their high total installing price, since we need many equipments to guarantee the same level of network coverage. Differently, combining different cell sizes makes the network more flexible and adaptable to various traffic loads.

In this proposed model we address energy efficiency of C-RAN from a different perspective, in which we account for energy management at a very early stage which is network planning. We jointly optimize energy consumption and planning of the network with a view to minimize both

CAPEX and OPEX costs, taking into account constraints imposed by C-RAN architecture.

Planning for C-RAN and mobile networks in general can be an iterative process and includes several phases. A first phase consists of collecting data related to the cost of installation and maintenance of equipments. The traffic demand which will be generated in the network is evaluated, and the choice of technology to be used is carried out. In this phase also, constraints that must be taken into account are defined, in order to ensure a better performance of the network in terms of reliability, transmission delay, bandwidth speed, etc. The exact amount of mobile traffic cannot be known in advance, because it depends all on the behavior of users. Therefore, it is often necessary to base on the historical traffic and some estimations, to define the potential traffic demands of users in the given area.

Let TPC be the set of traffic generators point which can be in reality one or a group of users, and TPT the set of coverage points, which do not generate any traffic, but are used to guarantee a maximum coverage. The collected data in the first phase serve as input for the phase of design which consists of determining the topology of the network. The proposed model set up, locate and position the RRHs in order to ensure a better distribution of signals, and a maximum coverage of the area considering both TPC and TPT. The model tries also to find the best location for the BBUs pools, taking into consideration the limitations imposed by the transmission medium that links it with different RRHs. Theoretically, it is possible to place different nodes everywhere, but when it comes to practice, the potential positions should be known in advance. It consists usually of sites and buildings rented or sometimes owned by the network operator. So the goal here is to find among a set of locations of RRHs denoted by SR, and of BBUs Pools denoted by SB, the ones that guarantee a maximum coverage with the minimum costs, and that respect network constraints. The model determines also the configuration of equipments to be installed. Several RRHs with different performances and capacities can be possible candidates denoted by the set CR, as well as various characteristics of BBUs pools can be considered denoted as set CB.

To take into consideration energy management, we consider a one-day horizon, divided into many time intervals within a set T, differenti-

ated based on different traffic levels. Therefore, while planning for the network, our model manages power by turning on and of RRHs to optimize energy saving, as it activates BBUs inside the pool depending on the number of active RRHs. Alongside, to guarantee a better quality of service, the model try to assign users to the closest active RRH.

The last step is performance analysis phase, to check if the resulting network respects the predetermined constraints. Otherwise, modification in the input parameters must be done, and the planning process is repeated until a satisfactory solution is achieved.

We formulate the problem as a mixed integer linear programed model (MILP). We solve it based on one day traffic profile variation. The model mainly jointly optimize the planning and energy management of C-RAN based on the level of importance that we decide. This level is set as a given weight allocated to energy management and users assignment parts of the problem, which we explore in more details in the objective function section.

5.3.2 Model Formulation

In this section, we introduce decision variables of our problem. After that, we formulate the objective function and we define the related constraints.

5.3.2.1 Decision Variables

The goal of our model is to minimize the total cost of C-RAN by balancing expenses between installation cost CAPEX, and operating cost OPEX in terms of energy consumption, as well as to allocate users to the closest available RRH to guarantee the best quality of service.

Several set of variables are used to formulate the model. Regarding CAPEX, the binary variable z_{jk} represents weather a RRH with a configuration k have to be installed or not in the site j that belongs to the set of candidate sites SR. A similar binary variable w_{lf} is for BBU pool installation. The distance between selected sites of RRHs and BBU pools is a critical choice because of the high price of transport medium installation. We use the binary variable m_{jl} to represent whether or not a link between site j and site l need to be installed.

The second set of decision variables is related to the OPEX, more specifically to energy consumption expenses. The binary variable y_{jk}^t indicates the energy level of a RRH with a configuration k during time t. In case if a RRH is in installed in a site j, we consider two energy levels depending on weather the RRH is on or off. As for the BBU pool, the integer variable $NBBU_l^t$ indicates the number of active BBUs inside the pool if it is installed in a site l.

We linearize the non linear multiplication of tow binary variables y_{jk}^t and m_{jl} by introducing the variable s_{jlk}^t. In particular, it indicates weather or not an activated RRH installed in a site j is linked to a BBU pool installed in site l. This variable helps in accounting the number of RRHs that are processing their baseband in a specific pool.

Finally, concerning the quality of service, all users within the considered geographical area have to be assigned to active RRHs in which they are covered by. The binary variable x_{ij}^t is used for traffic point assignment to RRHs.

All defined variables are summarized in Table 5.1.

CAPEX variables	
z_{jk}	weather or not a RRH is installed in j with a configuration k
w_{la}	weather or not a BBU is installed in l with a configuration f
m_{jl}	weather or not a link exists between site j and site l
OPEX variables	
y_{jk}^t	equal to 1 if RR j is active during t
$NBBU_l^t$	number of active BBUs inside a pool
s_{jlk}^t	linearization variable
QoS variables	
x_{ij}^t	weather or not a i is assigned to j during time t

Table 5.1: *Decision Variables*

5.3.2.2 Objective function

The objective of this model is to minimize the total cost and guarantee a level of quality of service. We formulate the objective function based on three main parts: CAPEX, OPEX and Quality of service.

$$\begin{aligned}\min & \sum_{j\in SR}\sum_{k\in CR} z_{jk}\gamma_j + \sum_{l\in SB}\sum_{f\in CB} w_{lf}\tau_f + \sum_{j\in SR}\sum_{l\in SB} m_{jl}\,d_{jl}\,cp \\ & + \alpha\left(\sum_{j\in SR}\sum_{k\in CR}\sum_{t\in T} y_{jk}^t\, e_k \delta_t + \sum_{l\in SB}\sum_{T\in T} NBBU_l^t\, eb\delta_t\right) \\ & + \beta \sum_{i\in TPT}\sum_{j\in SR}\sum_{t\in T} x_{ij}^t \delta_t r_{ij}\end{aligned} \tag{5.1}$$

The first term of this objective function accounts for CAPEX. We consider the sum of installation costs of different configurations of RRHs and BBU pools, where parameter γ_j represents the installation cost of a RRH in a site j, and parameter τ_a refers to installation cost of a BBU hotel in site l. As for the transport link, the price of installing the optical fiber cp is multiplied by the distance d_{jl} between installed RRH in site j and its related pool in site l.

The second term in this equation accounts for OPEX. In order to minimize the energy consumption price, we sum the energy consumption e_k of activated RRHs with different configurations, taking into account the duration of a time band δ_t. Concerning BBU pools, the energy consumption of each pool is proportional to the number of activated BBUs inside, where eb indicates the energy consumption of a single BBU.

The last term is introduced to guarantee the best quality of connection between users and RRHs. The model tries to assign users to the closest RRH based on the distance between them denoted as r_{ij}.

Parameters α and β are used for trade-off between the three terms of the objective function. Therefore, more weight we give to this parameters, more importance their subproblem get while running the model.

5.3.2.3 Constraints

In this section we present different constraints used to describe the proposed model.

$$\sum_{j\in SR}\sum_{k\in \mathcal{CR}} a_{ijk}\, y_{jk}^t \geq 1 \quad \forall i \in \mathcal{TPC} \cup \mathcal{TPT}, \forall t \in \mathcal{T} \tag{5.2}$$

In particular, constraint (5.2) guarantees a minimal coverage by ensuring that all traffic points and coverage points are within the service area of one of the RRHs. Note that we consider a_{ijk} as coverage matrix.

The installed RRHs should be assigned and linked to only one BBUs pool, which we ensure using constraint (5.3):

$$\sum_{j \in SB} m_{jl} = \sum_{k \in CR} z_{jk} \quad \forall j \in \mathcal{SR} \tag{5.3}$$

Traffic points or users within service area of more than one remote radio head have to be assigned during transmission periods to only one active RRH. To guarantee this we use both constraints (5.4) (5.5):

$$x_{ij}^t \leq \sum_{k \in CR} a_{ijk} y_{jk}^t \quad \forall i \in \mathcal{TPT}, \forall j \in \mathcal{SR}, \forall t \in \mathcal{T} \tag{5.4}$$

$$\sum_{j \in SR} x_{ij}^t = 1 \quad \forall i \in \mathcal{TPT}, \forall t \in \mathcal{T} \tag{5.5}$$

The transport link between RRHs and BBUs have to be installed between two sites only if there are installed equipments on these sites, which is ensured by both constraints (5.6) and (5.3):

$$m_{jl} \leq w_{lf} \quad \forall j \in \mathcal{SR}, \forall l \in \mathcal{SB}, \forall f \in \mathcal{CB} \tag{5.6}$$

Satisfying traffic demand is the goal of mobile network operators. For each RRH configuration we set a maximum traffic that it can handle denoted by $crrh_k$. Each user within service area generates an amount of traffic indicated by p_i^t. The following equation presents traffic capacity constraint:

$$\sum_{i \in TP} x_{ij}^t p_i^t \leq \sum_{k \in CR} y_{jk}^t crrh_k \quad \forall j \in \mathcal{SR}, \forall t \in \mathcal{T} \tag{5.7}$$

Each pool also has a maximum number of RRHs that it can handle, we denote this number by $cpool_f$, and we use equation (5.8) to respect pool capacity constraint.

$$\sum_{j \in SR} m_{jl} \leq \sum_{f \in CB} w_{lf} cpool_f \quad \forall l \in \mathcal{SB} \tag{5.8}$$

Constraints (5.9) guarantee the correlation between RRH activation and installation site. In other words, a RRH cannot be activated in a site where it is not installed. On the other hand, constraints (5.10) (5.11) ensure that at most one configuration of RRH is used at the same site.

$$y_{jk}^{t} \leq z_{jk} \quad \forall j \in \mathcal{SR}, \forall k \in \mathcal{CR}, \forall t \in \mathcal{T} \tag{5.9}$$

$$\sum_{k \in CR} z_{jk} \leq 1 \quad \forall j \in \mathcal{SR}, \forall j \in \mathcal{SR} \tag{5.10}$$

$$\sum_{k \in CR} w_{lf} \leq 1 \quad \forall l \in \mathcal{SB}, \forall f \in \mathcal{SB} \tag{5.11}$$

A distance of 40km [52] is set as a maximum length of the transport link for latency constraint. To guarantee that we do not exceed this distance we use constraint (5.12), in which $maxi$ represents the maximum link length that we can set.

$$m_{jl}\, d_{lj} \leq maxi \quad \forall l \in \mathcal{SB}, \forall j \in \mathcal{SR} \tag{5.12}$$

Constraint (5.13) defines how many BBUs need to be active for satisfying the number of active RRHs. This number has to be lower than the pool capacity N, which is guaranteed by constraint (5.14)

$$\Big(\sum_{j \in SR} \sum_{k \in CR} s_{jlk}^{t}\Big)/N \leq m_{jl}\, d_{lj} \quad \forall l \in \mathcal{SB}, \forall t \in \mathcal{T} \tag{5.13}$$

$$m_{jl}\, d_{lj} \leq NBBU_{l}^{t} \quad \forall l \in \mathcal{SB}, \forall t \in \mathcal{T} \tag{5.14}$$

The last set of constraints (5.15), (5.16) and (5.17) is used to define the variable s_{jlk}^{t} which is used to linearize the multiplication of y_{jk}^{t} and m_{jl}.

$$s_{jlk}^{t} \leq m_{jl} \quad \forall l \in \mathcal{SB}, \forall t \in \mathcal{T}, \forall j \in \mathcal{SR}, \forall k \in \mathcal{CR}, \tag{5.15}$$

$$s_{jlk}^{t} \leq y_{jk}^{t} \quad \forall l \in \mathcal{SB}, \forall t \in \mathcal{T}, \forall j \in \mathcal{SR}, \forall k \in \mathcal{CR}, \tag{5.16}$$

$$s^t_{jlk} \geq m_{jl} + y^t_{jk} - 1 \quad \forall l \in \mathcal{SB}, \forall t \in \mathcal{T}, \forall j \in \mathcal{SR}, \forall k \in \mathcal{CR}, \qquad (5.17)$$

In table 5.2 we summarize all model parameters.

TPC	Set of Coverage points
TPT	Set of traffic points (Users)
SR	Set of candidate sites to install RRHs
SB	Set of candidate sites to install BBU pools
CR	Set of configurations of RRHs
SB	Set of configurations of BBUs pools
γ_j	Price of installing a RRH in site j
τ_a	Price of installing a BBU pool in site l
cp	Price of Optical fiber between RRH and BBU pool
e_k	Energy consumption of RRH type k
eb	Energy consumption of a BBU
δ_t	Duration of a time interval t
r_{ij}	Distance between traffic points and RRHs sites
d_{lj}	Distance between RRHs sites and BBUs sites
a_{ijk}	Weather user i is in service area of RRH j With configuration k (Coverage matrix)
α	Trade-off parameter
β	Trade-off parameter
N	Maximum number of RRHs running per one BBU
p^t_i	Traffic generated by user i during t
$crrh_k$	Capacity of RRH with configuration k
$cpool_f$	Capacity of BBU pool with configuration f
$maxi$	Maximum length between RRHs and BBU pool

Table 5.2: *Model Parameters*

5.4 Model Evaluation

In order to evaluate the efficiency of the proposed model, the state-of-the-art solvers can be used. In this section we present a set of the tests done using realistic data, and we discuss the obtained results.

5.4.1 Parameter Setting

We considered C-RAN technology test scenarios, with three different types of RRHs that are able to be turned off in periods of low traffic. Each RRH has a different transmission power, therefore, we can have cell sizes ranging between Macro, Micro and Pico cells. In table 5.3 we summarize

used values for RRHs types. We consider one type of BBUs pool which is composed by a set of BBUs fitted with a virtualization technology. For the transmission link, optical fiber is considered to link RRHs with BBUs. Realistic values regarding installation prices, capacity and power consumption of the considered equipments have been obtained from both Vodafone and the results of EARTH project [35].

RRH Type	Installation price (€)	Power Consumption (*W*)	Traffic Capacity (*Mb/s*)	Coverage (*m*)
RRH 1	3000	1310.5	210	1230
RRH 2	2000	725.3	70	850
RRH 3	1000	11.7	70	241

Table 5.3: *RRHs Parameters*

Regarding users traffic, it varies depending on users' behavior as well as users density in a service area. In general, mobile traffic reaches its peak during day time, while it decreases significantly at night. Moreover, traffic demand in urban areas is higher compared to suburban and rural areas. In our tests, we base on the approximated pattern presented in [35] to generate users traffic. The traffic is generated on a daily basis, in which one day is divided to time periods depending on users' demands fluctuation. We consider eight time periods, where the duration of each period is δ_t.

To ensure a maximum coverage of the geographical area, we use coverage traffic points. Which is a set of points distributed as a grid in the given area. They do not generate any traffic but they help in guaranteeing service availability.

The used values are summarized in table 5.4.

Parameter	Value
τ_a	35000 €
eb	600 *W*
N	4
cp	50 €/m
$cpool_f$	10
$maxi$	40 *Km*

Table 5.4: *Model Parameters*

5.4.2 Numerical Results

In order to study the efficiency of the proposed optimization model we use AMPL linear programming language and IBM ILOG CPLEX 12.1 solver [148]. We run the model using an 8-core 2.4GHz Intel Xeon server with 96 GB of RAM. We test the model under a variation of the trade-off parameters in the objective function to study the effect of including energy management on the planning of C-RAN network.

We consider two different scenarios, in the following named *Scenario1* and *Scenario2*. In each scenario we vary the size of the geographical area expected to be served with mobile network, as well, we vary the number of candidate sites for RRHs and BBUs.

In *Scenario1*, the surface is considered to be 4x4 km^2, with 30 traffic points, 30 candidate sites for RRHs and 10 candidate sites for BBUs pools all distributed randomly. In the second scenario *Scenario2*, we increase the size of the area to 10x10 km^2, as well as the number of the traffic points to 100, and the candidate sites for RRHs to 80, while for BBUs pools we consider 20 candidate sites.

We tested the model under a variation of the weight parameters α and β. The percentage in the results are calculated compared to the case where $\alpha = 0$, in which energy management is not considered at the design phase. In this case, we consider only CAPEX and we ignore OPEX in the objective function. To calculate the energy consumption, all RRHs are considered to be on during all the day. As for the BBU pool, we consider a fixed number of kept on BBUs, which is the minimum to serve installed RRHs.

Regarding the "two steps" test, we use an energy management strategy for RRHs by turning them on and off, but we do it separately from planning process. In other words, we consider $\alpha = 0$ and we run the model to get a topology of the network, then we use the obtained results concerning positions of different equipments to run again the model but this time by ignoring the CAPEX part in the objective function.

In the rest of tests where $\alpha \neq 0$, we increase the weight given to the energy management as well as quality of service, and we observe the changes that occur on CAPEX, OPEX and energy consumption.

Figues: Figure 5.6, Figure 5.7, Figure 5.8 and Figure 5.9 illustrate the obtained results.

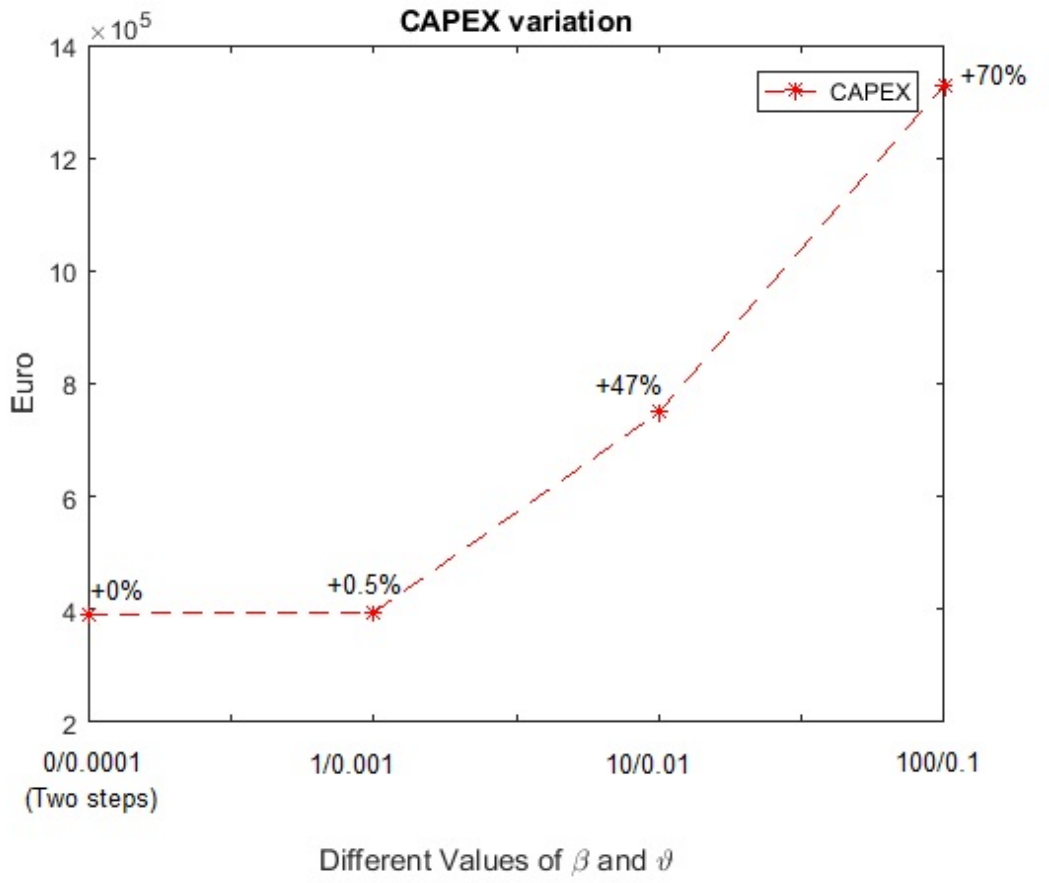

Figure 5.6: *CAPEX Scenario1*

Regarding computation complexity, the time needed to run the model is few seconds for the cases when α is equal to 0 and 1. By increasing the value of α more, the computation time needed is few minutes for the tested scenarios.

To analyze the results of *Scenario1* and *Scenario2*, we start by comparing the case when α = 0 to the "two steps" case. We observe that significant energy savings (42% in *Scenario1* and 65% in *Scenario2*) can be reached thanks to the energy management strategy, even if there is no difference in the obtained network topology for both cases. By increasing the value of α we remark increases in the number of installed RRHs, obviously accompanied with an increase in length the transport cable, which leads to increasing in CAPEX. While we observe a decreases in the consumed energy.

Comparing the case of α = 1 to the case where α = 0, installation

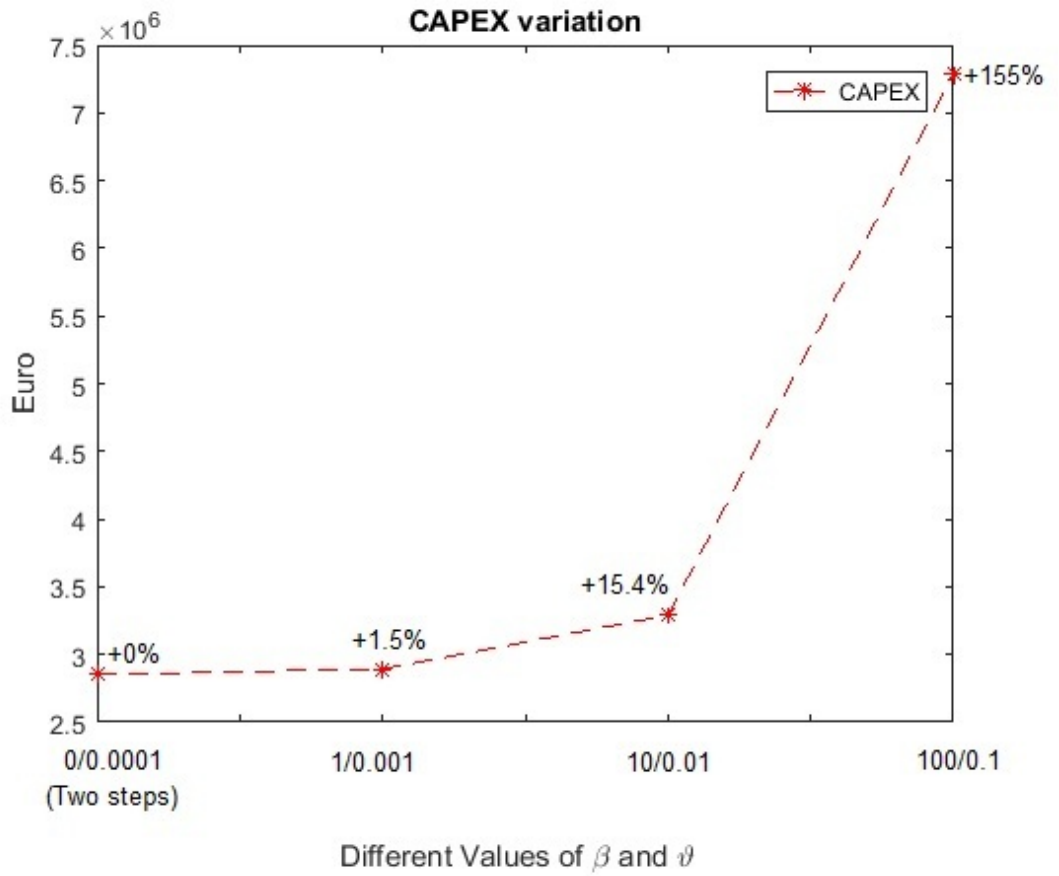

Figure 5.7: *CAPEX Scenario2*

costs increases slightly (1.5%) while energy saving are quite considerable (49% in *Scenario1* and 69% in *Scenario2*). By increasing the value of α to 10, we still get gains in terms of energy saving, with an increase of installation cost of 47% in *Scenario1*, while it is only 15% in *Scenario2*. For α = 100, we remark a slight increase in energy saving comparing to the additional installation expenditure required (70% in *Scenario1* and +155% in *Scenario2*), which lead us to set 10 as a compromise value of α for both scenarios, since more increases provide negligible gains with respect to the additional CAPEX. This value cannot be considered as a compromise value for all cases. It all depends on the parameters used in the model. The network designer have to run the model with different values of α to decide which one is best for the considered case.

In tables 5.5 and 5.6 we summarize the results obtained for both *Scenario1* and *Scenario2* successively under different tests and values of the weight parameter α. This two tables report the obtained values for

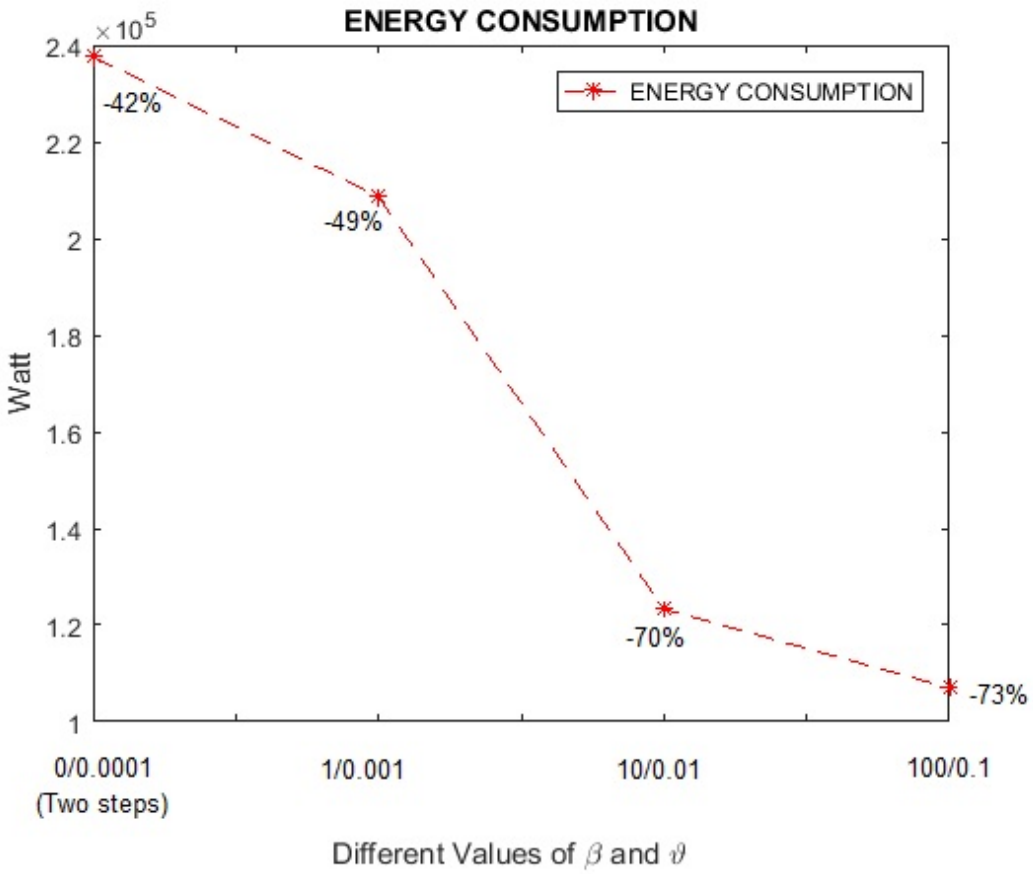

Figure 5.8: *ENERGY Scenario1*

CAPEX and OPEX, as well as the energy consumption of the network, and the number and the types of RRHs used ans also BBUs pools.

Another important observation is the size of the cells in each case. We noticed that by increasing the value of α the network tends to be more based on smaller cells than on big cells. We can observe this phenomenon in *Scenario1* and also more clearly in *Scenario2*. More importance we give to the energy management in the objective function, smaller the cells become in the network topology. The reason is that small cells have less coverage area therefore less users are served comparing to the big cells, which give the network the potential to turn off more equipments during low traffic periods. Regarding the BBUs pools, in *Scenario1* their number was 3 in almost of the cases, except the last case (α=100) where their number increases to 4 due to the increase of the number of RRHs. In *Scenario2*, their behavior is clearer. Their num-

	$\alpha = 0$ $\beta = 0.0001$	(Two steps) $\beta = 0.0001$	$\alpha = 1$ $\beta = 0.001$	$\alpha = 10$ $\beta = 0.01$	$\alpha = 100$ $\beta = 0.1$
CPAEX (€)	391945	391945 ***(+0%)***	394242 ***(+0.5%)***	749383 ***(+47%)***	1329308 ***(+70%)***
Energy (kWh)	410.400	237.600 ***(-42%)***	208.656 ***(-49%)***	123.312 ***(-70%)***	107.022 ***(-73%)***
OPEX (€)	143.64	83.16	73.02	43.15	37.45
RRH type 1	6	6	5	2	1
RRH type 2	0	0	1	9	17
RRH type 3	0	0	0	1	2
Number Of Pools	3	3	3	3	4

Table 5.5: *Results Scenario1*

	$\alpha = 0$ $\beta = 0.0001$	(Two steps) $\beta = 0.0001$	$\alpha = 1$ $\beta = 0.001$	$\alpha = 10$ $\beta = 0.01$	$\alpha = 100$ $\beta = 0.1$
CPAEX (€)	2851013	2851013 ***(+0%)***	2893496 ***(+1.5%)***	3289879 ***(+15.4%)***	7285683 ***(+155%)***
Energy (kWh)	11069.280	1219.680 ***(-65%)***	1060.704 ***(-69%)***	908.640 ***(-74%)***	866.440 ***(-75%)***
OPEX (€)	1228.24	426.88	371.24	318.02	303.25
RRH type 1	30	30	26	22	24
RRH type 2	5	5	9	15	35
RRH type 3	0	0	0	0	2
Number Of Pools	16	16	13	10	13

Table 5.6: *Results Scenario2*

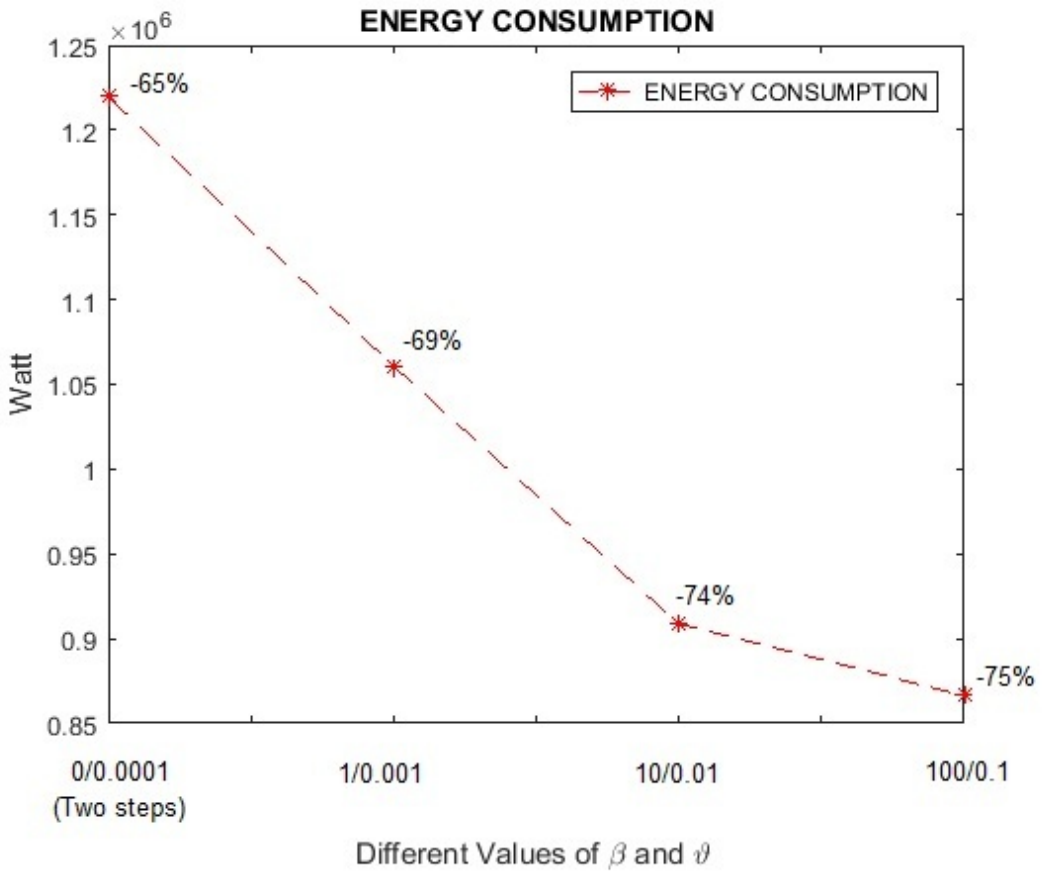

Figure 5.9: *ENERGY Scenario2*

ber were decreasing by increasing α, since the model tends to centralize the connected RRHs to a minimum number of pools to reduce power consumption of active BBUs. This behavior changes in the case of α = 100, were their number increased because more BBUs pools are needed to serve the number of installed RRHs.

5.5 Conclusion

C-RAN is emerging as a new paradigm for mobile telecommunication networks as a response to the continuous growth of mobile user devices demand. It is a promising technology for less operational expenditure and less energy consumption. The existing work on energy efficiency in C-RAN considers that the equipments are already installed and that the network is deployed. In contrast to them, in this chapter we proposed a

model that anticipates energy management at a very early stage, which is network planning stage.

We modeled the problem as a mixed integer linear programming (MILP) model. The proposed model is able to optimize the planning and energy management of C-RAN simultaneously, taking in to account quality of service of mobile users. Based on a realistic traffic data, we try to find among a set of candidate site of RRHs and BBUs pools, the ones that guarantee a maximum coverage of the served area taking into account the required constrains including the latency limits in the transport link. Meanwhile, we try to minimize the energy consumption of the network by switching on and off RRHs during low traffic periods to save energy, as well as optimizing the use of the pool by switching on the minimum number of BBUs that are able to process the baseband of the connected RRHs.

Regarding quality of service, mobile users tend to be served by the closest active RRH. This may help both users and service provider to reduce their energy consumption, by adjusting their transmission power instead of transmitting at a maximum power.

The model was solved using the state of the art solver CPLEX using realistic data. By variating the values of weights parameters, different topologies of the network were obtained as well as different energy consumption profiles. After a set of tests, we concluded that for the considered scenarios the value 10 for α is a compromise value, in other words, it allows to achieve significant energy savings compared to the additional installation cost required.

Another interesting observation is that more weight we give to the energy management, smaller the cells become, from which we can conclude that smaller cells if well managed can be more energy efficient that bigger cells.

CHAPTER 6

General Conclusion

Cloud computing is expanding and tends to emerge as a dominant paradigm in the computing landscape. Infrastructure offering cloud services are becoming more and more numerous, and more big to meet the growing demands of users. Obviously, this increase brings various problems, including that of energy consumption, or that of the effective use of resources. We must therefore design tools and techniques to respond to these new needs of management.

In this thesis, we have tried to resolve some aspects of the problem, more specifically to reduce the energy consumption of different cloud systems and networks. Unlike most of existing methods that handle with different related problems separately, in our works we have tried to enlarge the vision, and jointly consider different problems in single frameworks. Two types of Cloud systems were considered, and for each type joint optimization has proved to be more effective and more energy efficient with respect to the nature of considered problems.

In Chapter 4, we have introduced a mixed integer linear programming model based on a set of linear constraints. This model is based on the feature offered by virtualization technology to migrate virtual machines between different physical machines for load balancing. The defined

model has allowed to study different phenomena related to the problem, as well as the interactions between different parameters. We could define a multi objective model, that we can separate its sub-problems to three distinct problems: Data centers energy management, Network energy management, and Green energy management. The goal of the model is to optimize the three objectives concurrently with the aim of minimizing the total energy consumption and expenses of the Cloud system and its network.

The proposed model takes advantage of the differences between energy prices of various geographical locations of Data centers, as well as the availability of green energy resources. Basically, different cloud services are migrated between data centers through virtual machine migration technology. VM migration is performed when the price of energy is lower in the destination DCs or to explore more green energies. While migrating virtual machines, we take into consideration network traffic constraints, and also the routers consumption. Alongside, the model tends to balance cloud energy consumption between green and brown energies using storage technology. Green power can be stored in low traffic time periods for a later use when the price of electricity is so high. The amount of stored power can be used to address the requests received by the same data center, or received from other data centers from which services are migrated.

The obtained results by running the model under different scenarios show the importance of the joint optimization comparing to the separate optimization. Gains in term of energy expenses can be up to 34% comparing to the same proposed solution but considering a separate optimization, while enegry savings can be up to 70% comparing to only server based solution. Regarding energy consumption, savings can be really significant up to 43% beside the better exploitation of the environment friendly power.

In chapter 5, we have considered a new emerging type of Cloud systems which is Cloud Radio Access Network or C-RAN. Following the same principle of wider vision is more optimal, in this type of systems we have tried to minimize energy consumption at a very early stage which is network planning stage. We have introduced a mixed integer linear programming model in which we jointly optimize the planning and energy

management simultaneously, with the aim of minimizing both installation and operational costs (CAPEX and OPEX). In this multi objective model, we have tried to target three problems: equipments installation, energy management of RRHs and BBUs, and quality of service.

Basically, among a set of candidate sites for RRHs and BBUs pools, the objective of the model is to find the set of equipments to install in order to guarantee a maximum coverage of the targeted area, as well as to reduce the OPEX during the management phase. Power saving is achieved by turning On and Off RRHs depending on the traffic demands. Inside the BBU pool, the number of running BBUs is the minimum one that ensures the optimal serving for the connected RRHs. Regarding the Quality of Service, the model tries to connect users to the closest RRH for a better connection.

Weight parameters are used in the objective function for each subproblem. We tested the model using realistic data, and with variation of weight parameters. The obtained results show that significant energy savings can be achieved using joint optimization of planning and energy management of C-RAN. Energy savings with an additional CAPEX can reach 74% based on the tested scenarios. Also we observed that small cells are more energy efficient and easy to manage comparing to the big cells.

Regarding future works, different research directions can be started based on the proposed solutions. Concerning the first model, modeling in details the network topology and introducing routing algorithms is very interesting. Instead of the fully connected network, a real topology can be considered which makes the problem more close to reality. This is a critical aspect, since new constraints have to be introduced to the model thus increasing its complexity. Considering routing also in the model, may make the model unsolvable within a reasonable time. Concerning the second model of C-RAN, considering green power generation and management could be very interesting. The model can be modified to include in its CAPEX the renewable energy generators (Based on solar or wind) as well as battery systems for energy storage. Then in the OPEX part, green energy management including batteries maintenance can be included. For both models, any additional constraints will make the problem more complex to solve. Therefore, developing heuristic so-

lutions to suboptimal solve the aforementioned problems in a reasonable amount of time may be an interesting future research direction.

Bibliography

[1] http://docs.openstack.org/high-availability-guide/content/stateless-vs-stateful.html.

[2] http://docs.openstack.org/training-guides/content/module001-ch001-intro-text.html.

[3] http://opennebula.org.

[4] https://appengine.google.com/.

[5] https://aws.amazon.com/.

[6] https://aws.amazon.com/it/ec2/.

[7] https://cloudstack.apache.org.

[8] https://cloudstack.apache.org/docs/.

[9] https://it-it.facebook.com/.

[10] https://libvirt.org/.

[11] https://review.openstack.org/gitweb?p=openstack

[12] https://www.docker.com/.

[13] https://www.eucalyptus.com/.

[14] https://www.google.com/about/datacenters/inside/locations/index.html.

[15] https://www.google.com/gmail/.

[16] https://www.ipcc.ch.

[17] https://www.openstack.org/.

[18] http://www.google.ca/about/datacenters/efficiency/internal/.

[19] http://www.networkworld.com/article/2166407/cloud-computing/.

[20] http://www.nimbusproject.org/.

[21] Summary for policy makers: Renewable power generation costs. *International Renewable Energy Agency (IRENA), November 2012*.

[22] Uptime institute's 2014 data center survey.

[23] Juniper networks. *E120 and E320 Hardware Guide*, 2011.

[24] https://www.co2.earth. 2016.

[25] B. Addis, A. Capone, G. Carello, L.G. Gianoli, and B. Sansó. Energy management through optimized routing and device powering for greener communication networks. *IEEE/ACM Transactions on Networking*, 22(1):313–325, 2014.

[26] Bernardetta Addis, Antonio Capone, Giuliana Carello, Luca G Gianoli, and Brunilde Sanso. Multi-period traffic engineering of resilient networks for energy efficiency. In *Online Conference on Green Communications (GreenCom), 2012 IEEE*, pages 14–19. IEEE, 2012.

[27] Bernardetta Addis, Antonio Capone, Giuliana Carello, Luca G Gianoli, and Brunilde Sanso. Energy management through optimized routing and device powering for greener communication networks. *IEEE/ACM Transactions on Networking (TON)*, 22(1):313–325, 2014.

[28] Yahya Al-Hazmi and Hermann de Meer. Virtualization of 802.11 interfaces for wireless mesh networks. In *Wireless On-Demand Network Systems and Services (WONS), 2011 Eighth International Conference on*, pages 44–51. IEEE, 2011.

[29] Ghannam Aljabari and Evren Eren. Virtualization of wireless lan infrastructures. In *Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2011 IEEE 6th International Conference on*, volume 2, pages 837–841. IEEE, 2011.

[30] William Allcock, Joe Bester, John Bresnahan, Ann Chervenak, Lee Liming, and Steve Tuecke. Gridftp: Protocol extensions to ftp for the grid. *Global Grid ForumGFD-RP*, 20:1–21, 2003.

[31] Edoardo Amaldi, Antonio Capone, Luca G Gianoli, and Luca Mascetti. Energy management in ip traffic engineering with shortest path routing. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pages 1–6. IEEE, 2011.

[32] Inc. Amazon Web Services. Ec2 instance types amazon web services (aws), 2016.

[33] Michele Aresta. *Carbon dioxide as chemical feedstock*. John Wiley & Sons, 2010.

[34] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[35] Gunther Auer, Oliver Blume, Vito Giannini, Istvan Godor, M Imran, Ylva Jading, Efstathios Katranaras, Magnus Olsson, Dario Sabella, Per Skillermark, et al. D2. 3: Energy efficiency analysis of the reference systems, areas of improvements and target breakdown. *EARTH*, pages 1–69, 2010.

[36] Mario Baldi and Yoram Ofek. Time for a" greener" internet. In *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, pages 1–6. IEEE, 2009.

[37] Jayant Baliga, Robert WA Ayre, Kerry Hinton, and Rodney S Tucker. Green cloud computing: Balancing energy in processing, storage, and transport. *Proceedings of the IEEE*, 99(1):149–167, 2011.

[38] Masoud Barati and Saeed Sharifian. A hybrid heuristic-based tuned support vector regression model for cloud load prediction. *The Journal of Supercomputing*, 71(11):4235–4259, 2015.

[39] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *ACM SIGOPS operating systems review*, volume 37, pages 164–177. ACM, 2003.

[40] Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, 8(3):1–154, 2013.

[41] U Barth. Alcatel-lucent, bell labs stuttgart, how to reduce-green house gas emissions from ict equipment,(slides) wireless networks, earth research project, etsi green agenda. 2009.

[42] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5):755–768, 2012.

[43] Aruna Prem Bianzino, Claude Chaudet, Federico Larroca, Dario Rossi, and Jean-Louis Rougier. Energy-aware routing: a reality check. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 1422–1427. IEEE, 2010.

[44] Norman Bobroff, Andrzej Kochut, and Kirk Beaty. Dynamic placement of virtual machines for managing sla violations. In *Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on*, pages 119–128. IEEE, 2007.

[45] Raffaele Bolla, Franco Davoli, Roberto Bruschi, Ken Christensen, Flavio Cucchietti, and Suresh Singh. The potential impact of green technologies in next-generation wireline networks: Is there room for energy saving optimization? *Communications Magazine, IEEE*, 49(8):80–86, 2011.

[46] Sandro Bosio, Antonio Capone, and Matteo Cesana. Radio planning of wireless local area networks. *IEEE/ACM Transactions on Networking*, 15(6):1414–1427, 2007.

[47] Jens Buysse, Konstantinos Georgakilas, Anna Tzanakaki, Marc De Leenheer, Bart Dhoedt, Chris Develder, and Piet Demeester. Calculating the minimum bounds of energy consumption for cloud networks. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–7. IEEE, 2011.

[48] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616, 2009.

[49] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF 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, 2011.

[50] Ruay-Shiung Chang, Hui-Ping Chang, and Yun-Ting Wang. A dynamic weighted data replication strategy in data grids. In *Computer Systems and Applications, 2008. AICCSA 2008. IEEE/ACS International Conference on*, pages 414–421. IEEE, 2008.

[51] Jeffrey S Chase, Darrell C Anderson, Prachi N Thakar, Amin M Vahdat, and Ronald P Doyle. Managing energy and server resources in hosting centers. *ACM SIGOPS operating systems review*, 35(5):103–116, 2001.

[52] Aleksandra Checko, Henrik L Christiansen, Ying Yan, Lara Scolari, Georgios Kardaras, Michael S Berger, and Lars Dittmann. Cloud ran for mobile networks-a technology overview. *IEEE Communications surveys & tutorials*, 17(1):405–426, 2015.

[53] Aleksandra Checko, Henrik Lehrmann Christiansen, and Michael Stübert Berger. Evaluation of energy and cost savings in mobile cloud ran. *OPNETWORK 2013*, 2013.

[54] C Chen. C-ran: the road towards green radio access network. presentation, 2012.

[55] Gang Chen, Hai Jin, Deqing Zou, Bing Bing Zhou, Weizhong Qiang, and Gang Hu. Shelp: Automatic self-healing for multiple application instances in a virtual machine environment. In *Cluster Computing (CLUSTER), 2010 IEEE International Conference on*, pages 97–106. IEEE, 2010.

[56] Liming Chen, Hu Jin, Haoming Li, Jun-Bae Seo, Qing Guo, and Victor Leung. An energy efficient implementation of c-ran in hetnet. In *Vehicular Technology Conference (VTC Fall), 2014 IEEE 80th*, pages 1–5. IEEE, 2014.

[57] Luca Chiaraviglio, Delia Ciullo, Marco Mellia, and Michela Meo. Modeling sleep modes gains with random graphs. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 355–360. IEEE, 2011.

[58] Luca Chiaraviglio, Marco Mellia, and Fabio Neri. Energy-aware networks: Reducing power consumption by switching off network elements. In *FEDERICA-Phosphorus tutorial and workshop (TNC2008)*, 2008.

[59] Pei-Jung Chung. Green radio-sthe case for more efficient cellular base stations. *Moble Virtual Centre of Excellence (VCE) 2010*, 2010.

[60] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 273–286. USENIX Association, 2005.

[61] Holger Claussen, Lester TW Ho, and Florian Pivit. Effects of joint macrocell and residential picocell deployment on the network energy efficiency. In *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*, pages 1–6. IEEE, 2008.

[62] Hakan Coskun, Ina Schieferdecker, and Yahya Al-Hazmi. Virtual wlan: Going beyond virtual access points. *Electronic Communications of the EASST*, 17, 2009.

[63] Francisco Daz-Gonzlez, Andreas Sumper, Oriol Gomis-Bellmunt, and Roberto Villaffila-Robles. A review of energy storage technologies for wind power applications. *Renewable and Sustainable Energy Reviews*, 16(4):2154–2171, 2012.

[64] Supratim Deb, Pantelis Monogioudis, Jerzy Miernik, and James P Seymour. Algorithms for enhanced inter-cell interference coordination (eicic) in lte hetnets. *IEEE/ACM transactions on networking*, 22(1):137–150, 2014.

[65] Pierre Delforge. America's data centers consuming and wasting growing amounts of energy. *Natural Resource Defence Councle*, 2014.

[66] Umesh Deshpande and Kate Keahey. Traffic-sensitive live migration of virtual machines. *Future Generation Computer Systems*, 2016.

[67] Jiankang Dong, Xing Jin, Hongbo Wang, Yangyang Li, Peng Zhang, and Shiduan Cheng. Energy-saving virtual machine placement in cloud data centers. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pages 618–624. IEEE, 2013.

[68] Zvi Drezner and Horst W Hamacher. *Facility location*. Springer-Verlag New York, NY, 1995.

[69] Corentin Dupont, Fabien Hermenier, Thomas Schulze, Robert Basmadjian, Andrey Somov, and Giovanni Giuliani. Plug4green: A flexible energy-aware vm manager to fit data centre particularities. *Ad Hoc Networks*, 25:505–519, 2015.

[70] Truong Vinh Truong Duy, Yukinori Sato, and Yasushi Inoguchi. Performance evaluation of a green scheduling algorithm for energy savings in cloud computing. In *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–8. IEEE, 2010.

[71] Patrcia Takako Endo, Glauco Estcio Gonalves, Judith Kelner, and Djamel Sadok. A survey on open-source cloud computing solutions. In *Brazilian Symposium on Computer Networks and Distributed Systems*, volume 71, 2010.

[72] Weiwei Fang, Xiangmin Liang, Shengxin Li, Luca Chiaraviglio, and Naixue Xiong. Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers. *Computer Networks*, 57(1):179–196, 2013.

[73] Simson Garfinkel and Harold Abelson. *Architects of the information society: 35 years of the Laboratory for Computer Science at MIT*. MIT press, 1999.

[74] Erol Gelenbe and Yves Caseau. The impact of information technology on energy consumption and carbon emissions. *Ubiquity*, 2015(June):1, 2015.

[75] Chaima Ghribi, Makhlouf Hadji, and Djamal Zeghlache. Energy efficient VM scheduling for cloud data centers: Exact allocation and migration algorithms. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pages 671–678. IEEE, 2013.

[76] Google.ca. Efficiency: How we do it – data centers – google, 2016.

[77] Google.com. Data center locations – data centers – google, 2016.

[78] Lin Gu, Deze Zeng, Peng Li, and Song Guo. Cost minimization for big data processing in geo-distributed data centers. *Emerging Topics in Computing, IEEE Transactions on*, 2(3):314–323, 2014.

[79] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. Bcube: a high performance, server-centric network architecture for modular data centers. *ACM SIGCOMM Computer Communication Review*, 39(4):63–74, 2009.

[80] Chuanxiong Guo, Guohan Lu, Helen J Wang, Shuang Yang, Chao Kong, Peng Sun, Wenfei Wu, and Yongguang Zhang. Secondnet: a data center network virtualization architecture with bandwidth guarantees. In *Proceedings of the 6th International COnference*, page 15. ACM, 2010.

[81] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. Dcell: a scalable and fault-tolerant network structure for data centers. *ACM SIGCOMM Computer Communication Review*, 38(4):75–86, 2008.

[82] Vu Nguyen Ha, Long Bao Le, et al. Cooperative transmission in cloud ran considering fronthaul capacity and cloud processing constraints. In *Wireless Communications and Networking Conference (WCNC), 2014 IEEE*, pages 1862–1867. IEEE, 2014.

[83] Brandon Heller, Srinivasan Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. Elastictree: Saving energy in data center networks. In *NSDI*, volume 10, pages 249–264, 2010.

[84] Fabien Hermenier, Xavier Lorca, Jean-Marc Menaud, Gilles Muller, and Julia Lawall. Entropy: a consolidation manager for clusters. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 41–50. ACM, 2009.

[85] Michael R Hines and Kartik Gopalan. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 51–60. ACM, 2009.

[86] Kerry Hinton, Jayant Baliga, Michael Feng, Robert Ayre, and Rodney S Tucker. Power consumption and energy efficiency in the internet. *IEEE Network*, 25(2):6–12, 2011.

[87] Lei Huang, Qin Jia, Xin Wang, Shuang Yang, and Baochun Li. Pcube: Improving power efficiency in data center networks. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 65–72. IEEE, 2011.

[88] Qiang Huang, Fengqian Gao, Rui Wang, and Zhengwei Qi. Power consumption of virtual machine live migration in clouds. In *Communications and Mobile Computing (CMC), 2011 Third International Conference on*, pages 122–125. IEEE, 2011.

[89] X Huang, U Behr, and Werner Wiesbeck. Automatic cell planning for a low-cost and spectrum efficient wireless network. In *Global Telecommunications Conference, 2000. GLOBECOM'00. IEEE*, volume 1, pages 276–282. IEEE, 2000.

[90] Thanh Nguyen Huu, Christian Ngoc, et al. Modeling and experimenting combined smart sleep and power scaling algorithms in energy-aware data center networks. *Simulation Modelling Practice and Theory*, 39:20–40, 2013.

[91] Internetworldstats.com. World internet users statistics and 2015 world population stats, 2016.

[92] Ralf Irmer, Heinz Droste, Patrick Marsch, Michael Grieger, Gerhard Fettweis, Stefan Brueck, Hans-Peter Mayer, Lars Thiele, and Volker Jungnickel. Coordinated multipoint: Concepts, performance, and field trial results. *IEEE Communications Magazine*, 49(2):102–111, 2011.

[93] Wassim Itani, Cesar Ghali, Ayman Kayssi, Ali Chehab, and Imad Elhajj. G-route: an energy-aware service routing protocol for green cloud computing. *Cluster Computing*, 18(2):889–908, 2015.

[94] Joe Wenjie Jiang, Tian Lan, Sangtae Ha, Minghua Chen, and Mung Chiang. Joint vm placement and routing for data center traffic engineering. In *INFOCOM, 2012 Proceedings IEEE*, pages 2876–2880. IEEE, 2012.

[95] Hao Jin, Tosmate Cheocherngngarn, David Levy, Alex Smith, Deng Pan, Jiangchuan Liu, and Niki Pissinou. Joint host-network optimization for energy-efficient data center networking. In *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 623–634. IEEE, 2013.

[96] Georgios Kardaras and Christian Lanzani. Advanced multimode radio for wireless & mobile broadband communication. In *Wireless Technology Conference, 2009. EuWIT 2009. European*, pages 132–135. IEEE, 2009.

[97] Nikhil Karkare. A survey on the live migration of virtual machines. In *International Journal of Engineering Research and Technology*, volume 3. ESRSA Publications, 2014.

[98] M Khan, Raad S Alhumaima, and HS Al-Raweshidy. Reducing energy consumption by dynamic resource allocation in c-ran. In *Networks and Communications (EuCNC), 2015 European Conference on*, pages 169–174. IEEE, 2015.

[99] Zhen Kong, Jiayu Gong, Cheng-Zhong Xu, Kun Wang, and Jia Rao. ebase: A baseband unit cluster testbed to improve energy-efficiency for cloud radio access network. In *Communications (ICC), 2013 IEEE International Conference on*, pages 4222–4227. IEEE, 2013.

[100] 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.

[101] Rakesh Kumar and Sakshi Gupta. Open source infrastructure for cloud computing platform using eucalyptus. *Global Journal of Computers & Technology Vol*, 1(2):44–50, 2014.

[102] Bart Lannoo, S Lambert, W Van Heddeghem, M Pickavet, F Kuipers, G Koutitas, H Niavis, A Satsiou, MT Beck, A Fischer, et al. Overview of ict energy consumption. *Deliverable D8*, 1(02):2013, 2013.

[103] Bo Li, Jianxin Li, Jinpeng Huai, Tianyu Wo, Qin Li, and Liang Zhong. Enacloud: An energy-saving application live placement approach for cloud computing environments. In *Cloud Computing, 2009. CLOUD'09. IEEE International Conference on*, pages 17–24. IEEE, 2009.

[104] Yonghua Lin, Ling Shao, Zhenbo Zhu, Qing Wang, and Ravie K Sabhikhi. Wireless network cloud: Architecture and system requirements. *IBM Journal of Research and Development*, 54(1):4–1, 2010.

[105] Cheng Liu, Karthikeyan Sundaresan, Meilong Jiang, Sampath Rangarajan, and Gee-Kung Chang. The case for re-configurable backhaul in cloud-ran based small cell networks. In *INFOCOM, 2013 Proceedings IEEE*, pages 1124–1132. IEEE, 2013.

[106] Jie Liu, Feng Zhao, Xue Liu, and Wenbo He. Challenges towards elastic power management in internet data centers. In *Distributed Computing Systems Workshops, 2009. ICDCS Workshops' 09. 29th IEEE International Conference on*, pages 65–72. IEEE, 2009.

[107] Ruoyan Liu, Huaxi Gu, Xiaoshan Yu, and Xiumei Nian. Distributed flow scheduling in energy-aware data center networks. *Communications Letters, IEEE*, 17(4):801–804, 2013.

[108] Peter Mell and Tim Grance. The nist definition of cloud computing. *National Institute of Standards and Technology*, 53(6):50, 2009.

[109] R Miller. Facebook installs solar panels at new data center. *Datacenter Knowledge*, 16, 2011.

[110] Rich Miller. Wind-powered data center in wyoming | data center knowledge, 2014.

[111] Dejan Milojicic and Rich Wolski. Eucalyptus: Delivering a private cloud. *Computer*, 44(4):102–104, 2011.

[112] China Mobile. C-ran: the road towards green ran. *White Paper, ver*, 2, 2011.

[113] Tran Manh Nam, Nguyen Huu Thanh, Ngo Quynh Thu, Hoang Trung Hieu, and Stefan Covaci. Energy-aware routing based on power profile of devices in data center networks using sdn. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2015 12th International Conference on*, pages 1–6. IEEE, 2015.

[114] Shinobu Namba, Takashi Matsunaka, Takayuki Warabino, Shoji Kaneko, and Yoji Kishi. Colony-ran architecture for future cellular network. In *Future Network & Mobile Summit (FutureNetw), 2012*, pages 1–8. IEEE, 2012.

[115] Shinobu Namba, Takayuki Warabino, and Shoji Kaneko. Bbu-rrh switching schemes for centralized ran. In *Communications and Networking in China (CHINACOM), 2012 7th International ICST Conference on*, pages 762–766. IEEE, 2012.

[116] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In *NSDI*, volume 8, pages 323–336, 2008.

[117] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. The eucalyptus open-source cloud-computing system. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 124–131. IEEE Computer Society, 2009.

[118] Annual Energy Outlook et al. Energy information administration. *Department of Energy*, 92010(9), 2010.

[119] Hemanshu A Patel and Arvind D Meniya. A survey on commercial and open source cloud monitoring. *International Journal of Science and Modern Engineering (IJISME), ISSN*, pages 2319–6386, 2013.

[120] Emerson Network Power. Energy logic: Reducing data center energy consumption by creating savings that cascade across systems. *White paper, Emerson Electric Co*, 9, 2009.

[121] Alessandro Redondi, Ilario Filippini, and Antonio Capone. Context management in energy-efficient radio access networks. In *Digital Communications-Green ICT (TIWDC), 2013 24th Tyrrhenian International Workshop on*, pages 1–5. IEEE, 2013.

[122] Josh Reese. Solution methods for the p-median problem: An annotated bibliography. *Networks*, 48(3):125–142, 2006.

[123] Juan Camilo Cardona Restrepo, Claus G Gruber, and Carmen Mas Machuca. Energy profile aware routing. In *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, pages 1–5. IEEE, 2009.

[124] Pierre Riteau. Building dynamic computing infrastructures over distributed clouds. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 2097–2100. IEEE, 2011.

[125] Dario Sabella, Peter Rost, Yingli Sheng, Emmanouil Pateromichelakis, Umer Salim, Patricia Guitton-Ouhamou, Marco Di Girolamo, and Giovanni Giuliani. Ran as a service: Challenges of designing a flexible ran architecture in a cloud-based heterogeneous mobile network. In *Future Network and Mobile Summit (FutureNetworkSummit), 2013*, pages 1–8. IEEE, 2013.

[126] PN Sabharwal. Integrating netscaler with cloudstack. *Apache CloudStack Cloud Computing, Packt Publishing*.

[127] USA Santa Clara, CA. Intel heterogenous network solution brief. 2011.

[128] Greg Semeraro, Grigorios Magklis, Rajeev Balasubramonian, David H Albonesi, Sandhya Dwarkadas, and Michael L Scott. Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. In *High-Performance Computer Architecture, 2002. Proceedings. Eighth International Symposium on*, pages 29–40. IEEE, 2002.

[129] Peter Sempolinski and Douglas Thain. A comparison and critique of eucalyptus, opennebula and nimbus. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 417–426. Ieee, 2010.

[130] Li Shang, Li-Shiuan Peh, and Niraj K Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on*, pages 91–102. IEEE, 2003.

[131] Yunfei Shang, Dan Li, and Mingwei Xu. A comparison study of energy proportionality of data center network architectures. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, pages 1–7. IEEE, 2012.

[132] Subhadra Bose Shaw and Anil Kumar Singh. Use of proactive and reactive hotspot detection technique to reduce the number of virtual machine migration and energy consumption in cloud data center. *Computers & Electrical Engineering*, 2015.

[133] Hiroki Shirayanagi, Hiroshi Yamada, and Kono Kenji. Honeyguide: A vm migration-aware network topology for saving energy consumption in data center networks. *IEICE TRANSACTIONS on Information and Systems*, 96(9):2055–2064, 2013.

[134] Tshiamo Sigwele, Atm S Alam, Prashant Pillai, and Yim F Hu. Energy-efficient cloud radio access networks by cloud based workload consolidation for 5g. *Journal of Network and Computer Applications*, 78:1–8, 2017.

[135] James E Smith and Ravi Nair. The architecture of virtual machines. *Computer*, 38(5):32–38, 2005.

[136] Debora Souza, Rubens Matos, Jean Araujo, Vandi Alves, and Paulo Maciel. Eucabomber: Experimental evaluation of availability in eucalyptus private clouds. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 4080–4085. IEEE, 2013.

[137] Anja Strunk and Waltenegus Dargie. Does live migration of virtual machines cost energy? In *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*, pages 514–521. IEEE, 2013.

[138] Mina Taheri and Nirwan Ansari. Power-aware admission control and virtual machine allocation for cloud applications. In *Sarnoff Symposium, 2015 36th IEEE*, pages 134–139. IEEE, 2015.

[139] Giovanni Toraldo. *Opennebula 3 cloud computing*. Packt Publishing Ltd, 2012.

[140] Ward Van Heddeghem, Sofie Lambert, Bart Lannoo, Didier Colle, Mario Pickavet, and Piet Demeester. Trends in worldwide ict electricity consumption from 2007 to 2012. *Computer Communications*, 50:64–76, 2014.

[141] Luis M Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55, 2008.

[142] Akshat Verma, Puneet Ahuja, and Anindya Neogi. pmapper: power and migration cost aware application placement in virtualized systems. In *Middleware 2008*, pages 243–264. Springer, 2008.

[143] Tuan Viet-DINH. Cloud data management. *ENS de Cachan, IFSIC, IRISA, KerData Project-Team*, pages 1–5, 2010.

[144] Kaiwei Wang, Ming Zhao, and Wuyang Zhou. Graph-based dynamic frequency reuse in cloud-ran. In *Wireless Communications and Networking Conference (WCNC), 2014 IEEE*, pages 105–110. IEEE, 2014.

[145] Lin Wang, Fa Zhang, Jordi Arjona Aroca, Athanasios V Vasilakos, Kai Zheng, Chenying Hou, Dan Li, and Zhiyong Liu. A general framework for achieving energy efficiency in data center networks. *arXiv preprint arXiv:1304.3519*, 2013.

[146] Andreas Wolke. *Energy efficient capacity management in virtualized data centers*. PhD thesis, Universität München, 2015.

[147] Timothy Wood, Prashant J Shenoy, Arun Venkataramani, and Mazin S Yousif. Black-box and gray-box strategies for virtual machine migration. In *NSDI*, volume 7, pages 17–17, 2007.

[148] Www-01.ibm.com. Ibm cplex optimizer - united states, 2016.

[149] Elias Yaacoub. A practical approach for base station on/off switching in green lte-a hetnets. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 IEEE 10th International Conference on*, pages 159–164. IEEE, 2014.

[150] Behnam Zakeri and Sanna Syri. Electrical energy storage systems: A comparative life cycle cost analysis. *Renewable and Sustainable Energy Reviews*, 42:569–596, 2015.

[151] Wenhua Zeng, Jianfeng Zhao, and Min Liu. Several public commercial clouds and open source cloud computing software. In *Computer Science & Education (ICCSE), 2012 7th International Conference on*, pages 1130–1133. IEEE, 2012.

[152] Heli Zhang, Hong Ji, Xi Li, Ke Wang, and Weidong Wang. Energy efficient resource allocation over cloud-ran based heterogeneous network. In *Cloud Computing Technology and Science (CloudCom), 2015 IEEE 7th International Conference on*, pages 483–486. IEEE, 2015.

[153] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010.

[154] Tao Zhao, Jian Wu, Sheng Zhou, and Zhisheng Niu. Energy-delay tradeoffs of virtual base stations with a computational-resource-aware energy consumption model. In *Communication Systems (ICCS), 2014 IEEE International Conference on*, pages 26–30. IEEE, 2014.

[155] Wentao Zhao and Shaowei Wang. Traffic density-based rrh selection for power saving in c-ran. *IEEE Journal on Selected Areas in Communications*, 34(12):3157–3167, 2016.

[156] Kuangyu Zheng, Xiaodong Wang, Li Li, and Xiaorui Wang. Joint power optimization of data center network and servers with correlation analysis. In *INFOCOM, 2014 Proceedings IEEE*, pages 2598–2606. IEEE, 2014.

[157] ZhenBo Zhu, Parul Gupta, Qing Wang, Shivkumar Kalyanaraman, Yonghua Lin, Hubertus Franke, and Smruti Sarangi. Virtual base station pool: towards a wireless network cloud for radio access networks. In *Proceedings of the 8th ACM international conference on computing frontiers*, page 34. ACM, 2011.