Energy efficient resource allocation in cloud computing environments

Chaima Ghribi

▶ **To cite this version:**

Chaima Ghribi. Energy efficient resource allocation in cloud computing environments. Networking and Internet Architecture [cs.NI]. Institut National des Télécommunications, 2014. English. NNT : 2014TELE0035 . tel-01149701

HAL Id: tel-01149701
https://theses.hal.science/tel-01149701

Submitted on 7 May 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THESE DE DOCTORAT CONJOINT TELECOM SUDPARIS et L'UNIVERSITE PIERRE ET MARIE CURIE

Ecole doctorale : Informatique, Télécommunications et Electronique de Paris

Présentée par

Chaima Ghribi

Pour obtenir le grade de

DOCTEUR DE TELECOM SUDPARIS

Energy Efficient Resource Allocation in Cloud Computing Environments

---

Soutenue le : 22 Décembre 2014

devant le jury composé de :

Prof. Pascal Lorenz	Rapporteur	Université de Haute-Alsace
Prof. Samir Tohmé	Rapporteur	Université de Versailles Saint-Quentin
Prof. Guy Pujolle	Examinateur	Université Paris 6
Dr. Lila Boukhatem	Examinateur	Université Paris-Sud
Dr. José Neto	Examinateur	Télécom SudParis
Prof. Djamal Zeghlache	Directeur de thèse	Télécom SudParis

Thèse n° 2014TELE0035

JOINT THESIS BETWEEN TELECOM SUDPARIS AND UNIVERSITY OF PARIS 6 (UPMC)

Doctoral School : Informatique, Télécommunications et Electronique de Paris

Presented by

Chaima Ghribi

For the degree of

DOCTEUR DE TELECOM SUDPARIS

Energy Efficient Resource Allocation in Cloud Computing Environments

---

Defense Date : 22 December 2014

Jury Members :

Prof. Pascal Lorenz	Reporter	University of Haute Alsace
Prof. Samir Tohmé	Reporter	University of Versailles Saint-Quentin
Prof. Guy Pujolle	Examiner	University of Paris 6
Dr. Lila Boukhatem	Examiner	Paris-Sud University
Dr. José Neto	Examiner	Telecom Sud Paris
Prof. Djamal Zeghlache	Director of thesis	Telecom Sud Paris

Thesis n° 2014TELE0035

Abstract

Cloud computing has rapidly emerged as a successful paradigm for providing IT infrastructure, resources and services on a pay-per-use basis over the past few years. As, the wider adoption of Cloud and virtualization technologies has led to the establishment of large scale data centers that consume excessive energy and have significant carbon footprints, energy efficiency is becoming increasingly important for data centers and Cloud. Today data centers energy consumption represents 3 percent of all global electricity production and is estimated to further rise in the future.

This thesis presents new models and algorithms for energy efficient resource allocation in Cloud data centers. The first goal of this work is to propose, develop and evaluate optimization algorithms of resource allocation for traditional Infrastructutre as a Service (IaaS) architectures. The approach is Virtual Machine (VM) based and enables on-demand and dynamic resource scheduling while reducing power consumption of the data center. This initial objective is extended to deal with the new trends in Cloud services through a new model and optimization algorithms of energy efficient resource allocation for hybrid IaaS-PaaS Cloud providers. The solution is generic enough to support different type of virtualization technologies, enables both on-demand and advanced resource provisioning to deal with dynamic resource scheduling and fill the gap between IaaS and PaaS services and create a single continuum of services for Cloud users.

Consequently, in the thesis, we first present a survey of the state of the art on energy efficient resource allocation in cloud environments. Next, we propose a bin packing based approach for energy efficient resource allocation for classical IaaS. We formulate the problem of energy efficient resource allocation as a bin-packing model and propose an exact energy aware algorithm based on integer linear program (ILP) for initial resource allocation. To deal with dynamic resource consolidation, an exact ILP algorithm for dynamic VM reallocation is also proposed. This algorithm is based on VM migration and aims at constantly optimizing energy efficiency at service departures. A heuristic method based on the best-fit algorithm

has also been adapted to the problem. Finally, we present a graph-coloring based approach for energy efficient resource allocation in the hybrid IaaS-PaaS providers context. This approach relies on a new graph coloring based model that supports both VM and container virtualization and provides on-demand as well as advanced resource reservation. We propose and develop an exact Pre-coloring algorithm for initial/static resource allocation while maximizing energy efficiency. A heuristic Pre-coloring algorithm for initial resource allocation is also proposed to scale with problem size. To adapt reservations over time and improve further energy efficiency, we introduce two heuristic Re-coloring algorithms for dynamic resource reallocation. Our solutions are generic, robust and flexible and the experimental evaluation shows that both proposed approaches lead to significant energy savings while meeting the users' requirements.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past few years, cloud computing has rapidly emerged as a successful paradigm for providing IT infrastructure, resources and services on a pay-per-use basis. The wider adoption of Cloud and virtualization technologies has led to the establishment of large scale data centers that provide cloud services. This evolution induces a tremendous rise of electricity consumption, escalating data center ownership costs and increasing carbon footprints. For these reasons, energy efficiency is becoming increasingly important for data centers and Cloud.

The fact that electricity consumption is set to rise 76% from 2007 to 2030 [2] with data centers contributing an important portion of this increase emphasizes the importance of reducing energy consumption in Clouds. According to the Gartner report [3], the average data center is estimated to consume as much energy as 25000 households, and according to McKinsey report [4], "*The total estimated energy bill for data centers in 2010 is* 11.5 *billion and energy costs in a typical data center double every five years*". Face to this electronic waste and to these huge amount of energy used to power data centers, energy efficient data center solutions have become one of the greatest challenges.

A major cause of energy inefficiency in data centers is the idle power wasted when resources are under used. In addition, this problem of low resources utilization, servers are permanently switched on even if they are not used and still consume up to 70% of their peak power. To address these problems, it is necessary to eliminate the power waste, to improve efficiency and to change the way resources are used. This can be done by designing energy efficient resource allocation solutions at different Cloud levels, which is the focus of this thesis.

In addition to these challenges, provided solutions should scale in multiple dimensions and Cloud providers must also deal with the users' requirements which are being more and more complex. Requested services are more sophisticated and complete since users need to deploy their own applications with the topology they choose and with having the control on both infrastructure and programs. This means combining the flexibility of IaaS and the ease of use of PaaS within a single environment. As a result, the classic three layer model is changing and the convergence of IaaS and PaaS is considered as natural evolutionary step in cloud computing. Cloud resource allocation solutions should be flexible enough to adapt to the evolving Cloud landscape and to deal with users requirements. This key dimension of cloud levels is essential for our research and we address it in depth in this thesis.

Another important dimension we consider is the type of the virtualization. In addition to traditional VM based technology, Cloud providers are also adopting new container-based virtualization technologies like LXC and Docker that enable the deployment of applications into containers. Hence, this resource variety aspect should be taken into account when modeling the problem of resource allocation to scale with the Cloud evolution and with new users requirements.

One last important dimension at which we are interested in this work is the resource provisioning plan. Cloud providers could offer two types of resource provisioning: on-demand and advance or long-term reservation. Advance reservation concept has many advantages especially for the co-allocation for resources. It provides simple means for resource planning and reservation in the future and offers an increased expectation that resources can be allocated when demanded. Although advance reservation of resources in cloud is very advantageous, the focus has been mostly on the on-demand plan.

Solving the problem of resource allocation in Cloud while maximizing energy efficiency and adopting the previously cited dimensions, is a very challenging issue. In this thesis, we address the problem with its multiple facets and levels to provide not only a specific solution, but also a generic and complete approach.

1.1 Research Problem and Objectives

Energy efficient Cloud resources allocation consists in identifying and assigning resources to each incoming user request in such a way, that the user requirements are met, that the least possible number of resources is used and that data center energy efficiency is optimized.

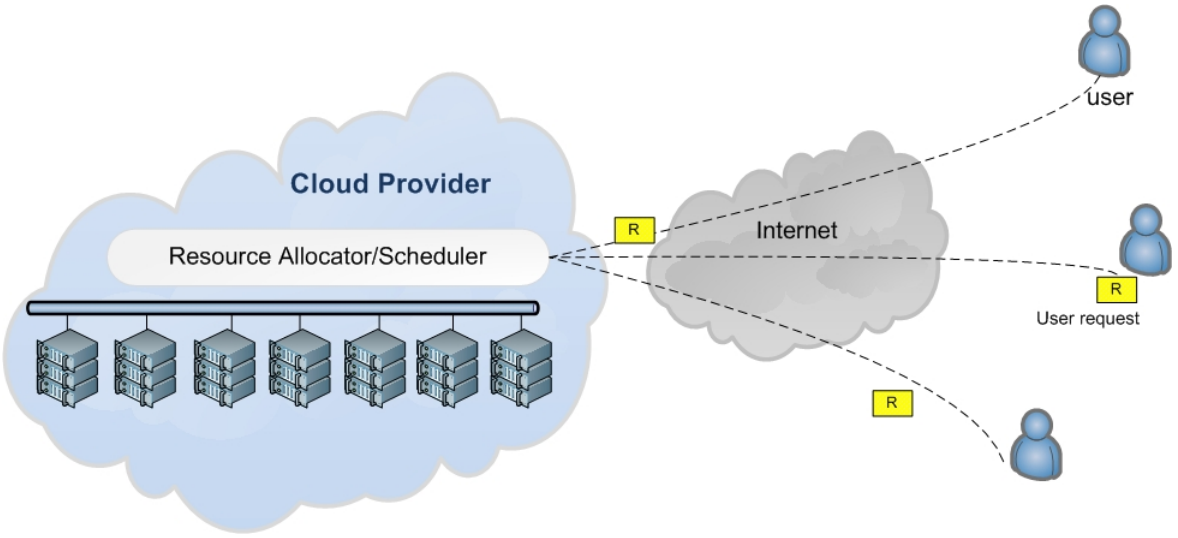

FIGURE 1.1: Resource Allocation in Cloud Computing

Even if Cloud resource allocation problem has been studied in the literature, much of the interest was focused on the IaaS layer and the dimensions of virtualization type and of provisioning plan were also not investigated enough. Some heuristic solutions for IaaS were proposed but there is still a lack of optimal algorithms to ensure energy efficient resource allocation. New hybrid Cloud solutions that combine Iaas and PaaS (e.g. openstack Heat) are evolving over time and being more and more attractive since they enable the joint deployment of infrastructure and applications. However, these solutions still lack energy efficient resource (VM or container) scheduling and no attention was paid to solve the problem at this level.

The main focus of this thesis is on the design and development of models and algorithms for energy efficient resource allocation in Cloud data centers. The first goal of this work is to propose, develop and evaluate optimization algorithms of resource allocation for traditional IaaS architectures that are widely used to manage clouds. The approach is VM based and it should enable on-demand and

dynamic resource scheduling while reducing the power consumption of the data center. This initial objective is naturally extended to deal with the new trends in Cloud. We aim to provide a new model and optimization algorithms of energy efficient resource allocation for IaaS-PaaS cloud providers. The solution should be generic enough to support different type of virtualization technologies, to enable both on-demand and advanced resource provisioning plans, to deal with dynamic resource scheduling and to fill the gap between IaaS and PaaS to create a single continuum of services for cloud users.

1.2 Contributions

Based on the objectives defined previously, we outline the main contributions of this thesis:

1. A survey of the state of the art on energy efficient resource allocation in cloud environments.
2. A bin packing based approach for energy efficient resource allocation:
    - We formulate the problem as a bin-packing model. The model is VM based and provides on-demand resource allocation in IaaS Clouds.
    - An exact energy aware algorithm based on integer linear program (ILP) for initial resource allocation.
    - An exact ILP algorithm for dynamic VM reallocation. It is based on VM migration and aims to optimize constantly the energy efficiency after service departures.
    - Combination of both previous exact algorithms in one algorithm that runs each of them when convenient.
    - A heuristic method based on best-fit algorithm adapted to the problem.
    - Evaluation and performance analysis of the proposed algorithms.
3. A graph coloring based approach for energy efficient resource allocation:
    - New graph coloring based model for energy efficient resource allocation in IaaS-PaaS providers. The model supports both VM and container virtualization and provides on-demand and advanced reservation resource provisioning.

- An exact Pre-coloring algorithm for initial/static resource allocation while maximizing energy efficiency.
- A heuristic Pre-coloring algorithm for initial/static resource allocation is proposed to scale with problem size.
- Two heuristic Re-coloring algorithms for dynamic resource reallocation are proposed to adapt reservations over time and to improve further energy efficiency.
- Evaluation and comparison of the exact and heuristic solutions in terms of energy efficiency, resource usage and convergence time.

1.3 Thesis Organization

This thesis is organized into six chapters. Chapter 2 provides an introduction to both Cloud computing and energy efficiency trends. We show how cloud is transforming IT and how sustainability is becoming increasingly important for Cloud data centers.

Chapter 3 describes the problem of resource allocation in Cloud environments. We provide background and state of the art solutions for energy efficient resource allocation. Then, we discuss the related issues and problems, as well as the challenges.

In Chapter 4, we present a bin-packing based solution for energy efficient resource allocation in IaaS Clouds. We propose exact and heuristic algorithms to perform initial resource allocation and dynamic resource reallocation while minimizing energy consumption and VM migration costs. Simulations are conducted to show the performance of our exact algorithms and to demonstrate their ability to achieve significant energy savings while maintaining feasible convergence times when compared with the heuristic solution.

Chapter 5 introduces a new graph coloring based solution for energy efficient resource allocation in integrated IaaS-PaaS environments. Both on-demand and advanced reservation plans are considered. We present exact and heuristic algorithms for initial resource allocation and dynamic resource reallocation while satisfying users' requirements and maximizing energy efficiency. Experimentations are conducted to assess the efficiency of our solution.

Chapter 6 draws conclusions, summarizes our major contributions and discusses perspectives, challenges and future work directions.

Chapter 2

Cloud Computing and Energy Efficiency

2.1 Introduction

Energy efficiency is becoming increasingly important for Cloud data centers. Their growing scale and their wide use have made a great issue of power consumption. Before beginning to solve the problem, it is important to study it in depth and to identify the reasons behind it.

This chapter introduces the concepts of Cloud computing and virtualization that serves as its enabling technology. We further investigate the problem of energy efficiency in Cloud data centers by studying the major causes of energy waste, presenting the different power saving techniques and introducing energy measurement and modeling in Cloud environments. Finally, we highlight the orientation and the focus of this thesis.

2.2 Cloud Computing

2.2.1 What is Cloud Computing?

Cloud computing has become one of the fastest growing paradigms in computer science. It is a model for providing IT resources as a service in a cost efficient

and pay-per-use way. By adopting Cloud services, companies and simple users are enabled to externalize their hardware resources, services, applications and their IT functions.

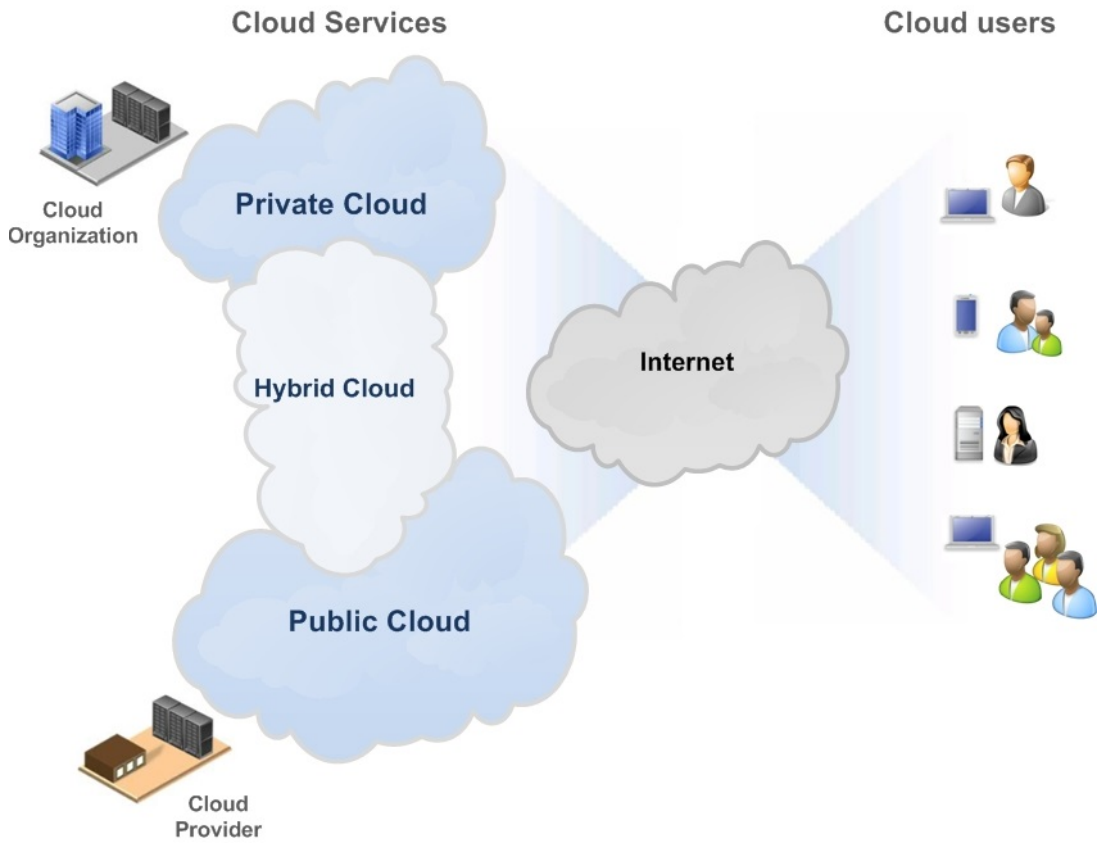

FIGURE 2.1: Cloud Computing

Although various definitions of cloud appear in the literature, there is no consensus on a clear and complete definition of this paradigm. The most widely accepted definition of cloud computing is that proposed by the National Institute of Standards and Technology (NIST). The proposed definition was: " *Cloud computing is a pay-per-use model for enabling convenient, on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications, and services. It can be rapidly provisioned and released with minimal management effort or service provider interaction*". From this definition we can identify the following key features of Cloud computing:

- On-demand self-service: automated on-demand resource provisioning.

- Broad network access: Resources can be accessed remotely over the network.
- Resource pooling: Resources are pooled and dynamically assigned independently from their physical location.
- Rapid elasticity: Capability can scale to cope with demand peaks.
- Measured Service: Resource usage is metered to enable the pay-per-use model.

An important aspect to consider with the Cloud is the ownership and use of the Cloud infrastructure. Different approaches can be used to deploy Cloud infrastructures:

Private cloud:
Refers to cloud infrastructures owned and managed by a single company, used in a private network and not available for public use.

Community cloud:
Refers to shared cloud infrastructures for specific communities composed by multiple users.

Public cloud:
Refers to high-performance and large infrastructures operated by external companies that provide IT services for many consumers via the Internet.

Hybrid cloud:
As the name already indicates, a hybrid cloud is a combination of both a private and public cloud. Parts of the service run on the company's private cloud, and parts are outsourced to an external public cloud.

2.2.2 Cloud Computing Actors

Cloud computing involves three main actors that have distinct roles and interactions inside the Cloud environment: providers, brokers and users.

Cloud Provider:
The provider possess the Cloud infrastructure on which Cloud services are deployed. This actor is responsible for the management and the control of cloud resources and for handling users' requests.

Cloud user:
A Cloud user is a person or an organization that consumes Cloud services.

Cloud Broker:
The Broker is an intermediate player between Cloud users and provider. It is responsible for the distribution incoming requests between the different providers based on users' requirements. To make a simple analogy, a Cloud broker is like a travel agency that acts as an intermediary between clients and service providers.

2.2.3 Cloud Services Overview

Cloud service models describe how services are made available to users. We distinguish between two different types of models : classic Cloud service models and new hybrid ones.

2.2.3.1 Classic Cloud service models

Classic Cloud service models can be categorized into three types: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).

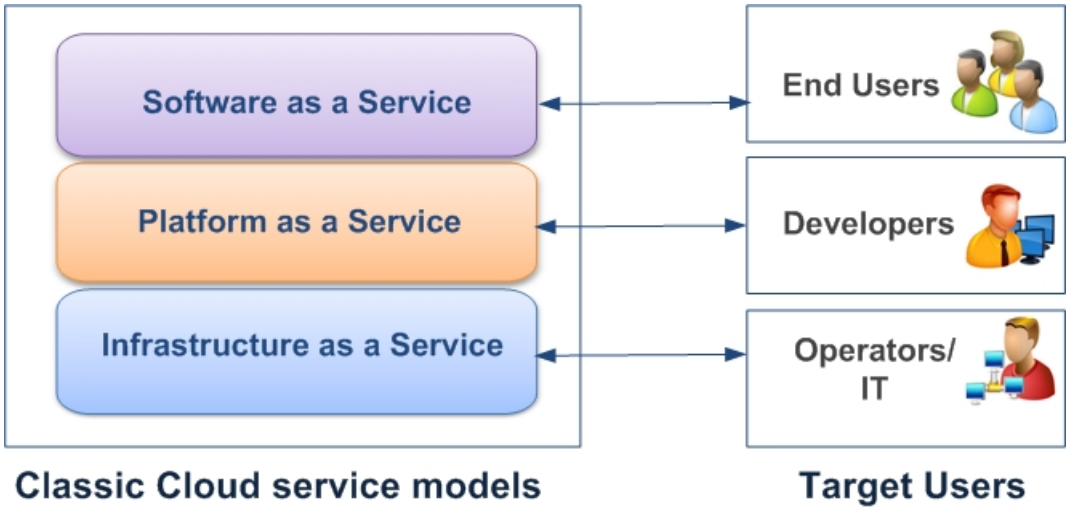

FIGURE 2.2: Classic Cloud service models

Infrastructure as a Service (IaaS):
IaaS is the most straightforward model for delivering cloud services. It refers to the

provisioning and the delivery of basic resources such as virtual machines, physical servers, network and storage. Instead of investing in their own infrastructure, companies are able to rent resources and use them on demand rather than having their resources locally. With IaaS, users have direct access to the lowest level in the stack and are able to build their application environments from scratch. An example of a popular IaaS Cloud is Amazon EC2[5].

Platform as a Service (PaaS):
PaaS is on a more sophisticated and higher level service compared to IaaS. It provides application development environments and software platforms to develop, deploy, and manage Cloud applications while not worrying about the technology and hiding the low-level details from the user. The most popular cloud platforms are Microsoft Azure Services[6] and Google App Engine[7].

Software as a Service (SaaS):
SaaS is the highest level of the Cloud service model. In this scenario, complete applications are provided to users through the internet. SaaS providers manage infrastructure and have complete control of the application softwares. Users just access their applications as if they were hosted locally and don't need to know anything about the Cloud or even be aware about the technologies details. SaaS examples are social media plateforms, mails and project management systems and the most popular SaaS applications are Google Documents[8] and Google Apps[9].

2.2.3.2 New hybrid service models

As Cloud is maturing and as users are requesting more flexibility and more control since they need to deploy their applications with the topology they choose and with having the control on both infrastructure and programs, the classic three layer concept has often been a subject of speculation and discussion. The consolidation of IaaS and PaaS is one of the key predictions for this year. Lines between Cloud services are blurring which results in the combination of IaaS and PaaS and to the appearance of new hybrid Cloud providers that enable users to create a single continuum of services.

Leading companies like Amazon[5], Microsoft[10] and Google[11] are concurrently going to blend IaaS and PaaS and don't want users to think strictly about IaaS or PaaS when they require Cloud services. An example of this trend of new combined

Cloud services is Kubernetes[12]. It is a newly released Google solution that solves both IaaS and PaaS. Openstack [13] IaaS provider is also gaining PaaS features and providing combined IaaS-PaaS services by orchestrating Docker [14] containers via Openstack Heat. Details about this mechanism are given in Appendix B.

451 Research[15], a leading global analyst and data company focused on the business of enterprise IT innovation, states that: " *Although it is maturing in technology and market, PaaS is getting squeezed between consolidation with IaaS and heavy use of SaaS. PaaS will most likely survive as a category, but not necessarily as we know it today*".

2.2.4 Virtualization and Cloud Computing

Virtualization technology is the main enabler of Cloud computing. It is based on physical resources abstraction in a way that several virtual resources are multiplexed on a physical one. Virtualization is used to provide isolation, flexibility, higher resource utilization, easy resource management as well as resource elasticity and to enable heterogeneous services to co-exist on the same physical hardware.

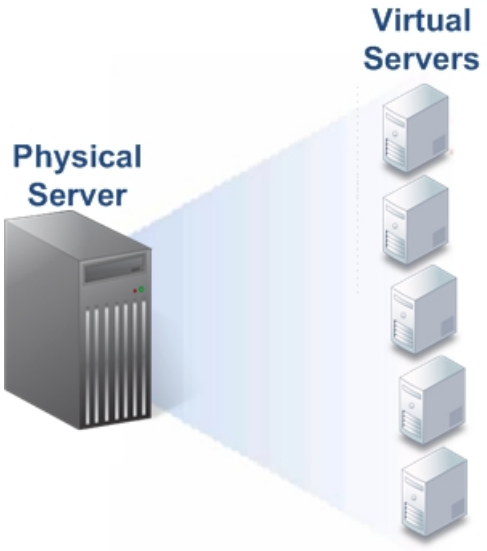

FIGURE 2.3: Server Virtualization

2.2.4.1 Virtualization Forms

Virtualization refers to a number of different technologies. The main types of virtualization are server virtualization, storage virtualization and network virtualization. All these types are based on the same idea of physical resource abstraction and partitioning.

In this thesis, we focus on server virtualization which is the most common resource abstraction technique in Cloud computing. This kind of virtualization allows multiple isolated virtual servers to run on a single one and can be implemented in different ways. Implementation approaches cover full virtualization, para-virtualization and OS-level virtualization. Both full virtualization and para-virtualization use a hypervisor to share the underlying hardware but differ on how the host operating system and the guest operating systems are modified to support virtualization and also on how they interact with each others. In contrast to full virtualization and para-virtualization, operating system level virtualization does not use a hypervisor at all. In this approach, all the virtual servers run the same host OS that performs all the functions of a fully virtualized hypervisor. Hence, based on the approach through which the virtualization is achieved, server virtualization can be classified into two main categories: hypervisor based virtualization and OS or container based virtualization. This classification is further detailed in the next section.

2.2.4.2 Server virtualization categories

There are two common ways to virtualize resources in Cloud computing: via hosted virtualization using a hypervisor or via container-based virtualization.

Hypervisor based virtualization:

Hypervisor based virtualization is the traditional way of doing virtualization in the Cloud. This technology is based on a layer of software, called hypervisor, which manages the physical server resources. Examples of hypervisors are KVM[16], VMWare[17], Microsoft Hyper-V[18], Xen[19] and Virtual Box[20]. Guests are called virtual machines VMs and they run different operating systems such as Windows and Linux on top of the same physical host. Even if this type of virtualization introduces an additional software layer, it enables resource consolidation

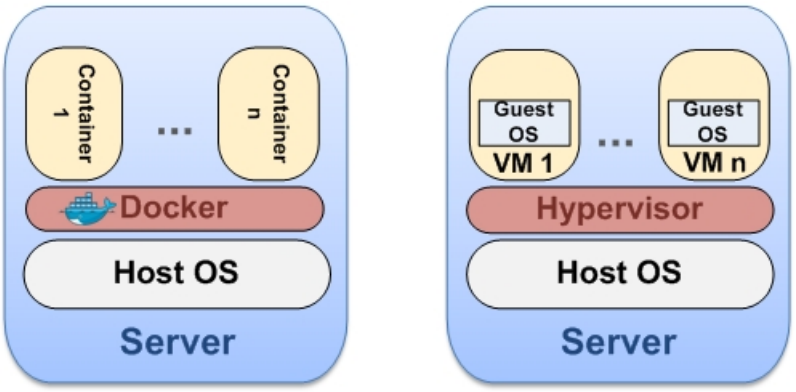

FIGURE 2.4: Container based virtualization vs hypervisor based virtualization

into virtualized servers [21] and also offers live migration feature [22] to move VMs to other servers without shutting them down.

Container based virtualization:

Container based virtualization is a lightweight alternative to the hypervisors [23][24]. It is an operating system level technology that allows running multiple isolated virtual environments on the same host. Containers are based on shared operating systems and unlike traditional VMs, they don't run different OSes but use a single operating system (the host's OS). Figure 2.4 shows the difference between the two kinds of virtualization. Some examples of container based solutions are: Docker[14], Linux containers (LXC)[25], Solaris Containers[26], Virtuozzo Containers [27] and OpenVZ[28].

It's more appropriate to use hypervisor based virtualization when more security and flexibly are required and when heterogeneous operating systems are needed [29]. Container based virtualization is convenient when performance is required. It provides better manageability with a near-native performance and gives a much higher consolidation ratio and most efficient resource usage as it supports large number of instances on a single host. In addition to being lightweight, this solution provides portability, transport, and process-level isolation across hosts.

Although being different, hypervisor and container based virtualization are not exclusive but complementary and increasingly used together. As container based virtualization is commonly used for building lightweight PaaS environments and

hypervisors are used for IaaS Cloud services, using both solutions enables the deployment of complex services that combine both applications and underlying infrastructures over hybrid IaaS/PaaS cloud providers. Some solutions like Proxmox [30] offer both technologies on the same physical server.

2.3 Energy Efficiency in Cloud Data Centers

2.3.1 Potential power consuming units in cloud datacenters

To improve energy efficiency in the Cloud, it is important to study the power flow in typical data centers and to understand how power is distributed. In fact, more than half of the electrical power is feeding the IT loads (see Figure 2.5). According to the EPA's Report to Congress on Server and Data Center Energy [31], servers consume 80% of the total IT load and 40% of total data center power consumption. The rest of power is consumed by other devices like transformers, distribution wiring, air conditioners, pumps, and lighting.

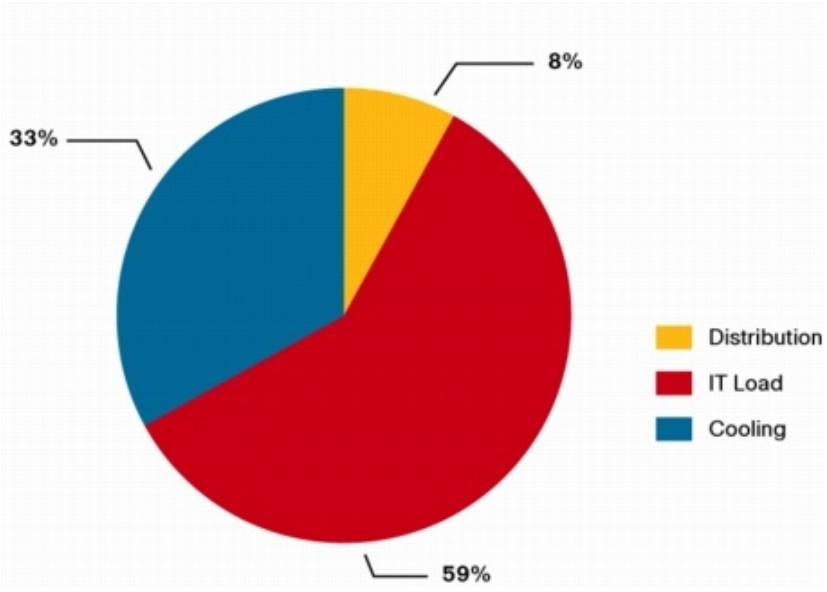

FIGURE 2.5: Typical power draw in a data center
Source: Cisco white paper [1]

The power consumption of cooling equipments is important but it is proportional to the IT power consumption. Technologies like free cooling that are used by big companies (e.g. Google, Facebook, ebay...), are interesting for reducing the power consumption of cooling. These approaches lower the air temperature in data centers by using naturally cool air or water instead of mechanical refrigeration. As a result, the electrical power needed for cooling has enormously decreased. Savings can even reach 100% in case of zero refrigeration which is possible in many climates.

2.3.2 Major causes of energy waste

As explained in the last section, servers are the main power consumers in Cloud data centers. The key reasons for this huge consumption are the following:

Low server utilization:
As data centers are growing in size, the number of servers is continuously increasing. Most data center servers are underused. According to the Natural Resources Defense Council (NRDC) report [32][33], average server utilization remained static between 12% and 18% from 2006 and 2012, while servers draw between 60% and 90% of peak power.

Consolidating virtual servers on a smaller number of hosts allows running the same applications with much lower power consumption. By increasing server utilization, the number of required servers and overall energy use will be greatly reduced.

Idle power waste:
Data center servers sit idly and are not processing useful work about 85-95% of the time[33]. An idle server consumes about 70% of its peak power even if it is not used [34]. This waste of idle power is considered as a major cause of energy inefficiency. Hence, idle servers in data centers could be turned off to reduce energy consumption.

Lack of a standardized metric of server energy efficiency:
To insure energy efficiency optimizations, it is important to use energy efficiency metric for servers to sort them according to their energy efficiency and to enable scheduling algorithms to make decisions and to select the best resources to maximize energy efficiency. Even though some metrics focusing on IT efficiency have appeared in recent years [35], they do not provide a simple benchmark that can drive the optimization of energy efficiency [33].

Energy efficient solutions are still not widely adopted:
As stated in the NRDC report [33], many big Cloud farms do a great job on energy efficiency, but represent less than 5% of the global data centers' energy use. The other 95% small, medium, corporate and multi-tenant operations are much less efficient on average. Hence, energy efficiency best practices should be more adopted and used especially for small and medium sized data centers that are typically very inefficient and consume about half of the amount of power consumed by all the data centers.

2.3.3 Power measurement and modeling in Cloud

Before dealing with power and energy measurement and modeling, it is important to understand power and energy relationship and to present their units of measure.

Power consumption indicates the rate at which a machine can perform its work and can be found by multiplying voltage and current while electrical energy is the amount of power used over a period of time. The standard metric unit of power is the watt (W) and the energy unit is watt-hour (Wh). Power and energy can be defined as shown in 2.1 and 2.2, where P is power consumption, I is current, V is voltage, E is energy and T is a time interval:

$$P = IV \tag{2.1}$$

$$E = PT \tag{2.2}$$

To quantify power and energy consumption in Cloud, we distinguish between measurement techniques and power and energy estimation models. The first one directly measures actual power consumption via instant monitoring tools. Power metering models estimate the power consumption of servers and VMs using hardware-provided or OS-provided metrics.

2.3.3.1 Power measurement techniques

Power direct measurement in Cloud can be achieved in data centers that embed monitoring capabilities and probes such as smart power distribution units (PDUs).

This section introduces several measurement methods to obtain information about the power consumption of servers and VMs.

Power measurement for servers:
The obvious way to get accurate information about energy consumption of servers is to directly measure it. However, this requires extra hardware to be installed in the hosts, need to add intelligent monitoring capabilities in the data center and to deal with huge amounts of data. Green Open Cloud (GOC) [36] is an example of energy monitoring and measurement framework that relies on energy sensors (wattmeters) to monitor the electricity consumed by Cloud resources. It collects statistics of the power usage in real-time and embeds electrical sensors that provide dynamic measurements of energy consumption and an energy-data collector.

Power measurement for VMs:
Even if power consumption of servers can be measured in real time, power consumption of VMs cannot be measured by any sensor and cannot be connected to a hardware measurement device. Some effort was done in [36] to measure VM power consumption. The virtual machine power consumption is computed by retrieving the idle power from the power consumption of the server when it hosts the VM, which is impractical and not very accurate. Alternative solutions based on extending a power monitoring adaptor between the server driver modules and the hypervisor are proposed in [37] and [38]. However, this solutions measure the total power consumed by the virtualization layer and don't provide per VM power usage.

2.3.3.2 Power and energy estimation models

As most servers in modern data center are not equipped with power measurement devices and as VM power cannot be measured by sensors, models that estimate the power and energy consumption as well as VM migration power cost are being more and more attractive for power metering. This section presents a general overview of power estimation models and tools in Cloud and introduces data center energy efficiency metrics.

Power and energy modeling for servers:
Power consumption models for servers have been extensively studied in the literature [39] and vary from complex to simple.

As the CPU of a server consumes the most important amount of power and as the relationship between power and CPU utilization is linear, CPU based linear models represent a lightweight and a simple way to estimate servers' power usage [40]. In [41], [42], [43] and [44] simple utilization based power models for servers are proposed. They assume that CPU is the only factor in their power models and present an approximation for total power against CPU utilization (U) as shown in 2.6 and 2.3:

$$P = P_{idle} + U * (P_{Peak} - P_{idle}) \tag{2.3}$$

P is total power consumption, P_{Peak} is peak power consumption, P_{idle} is idle power consumption, and U is CPU utilization (a fraction between 0 and 1).

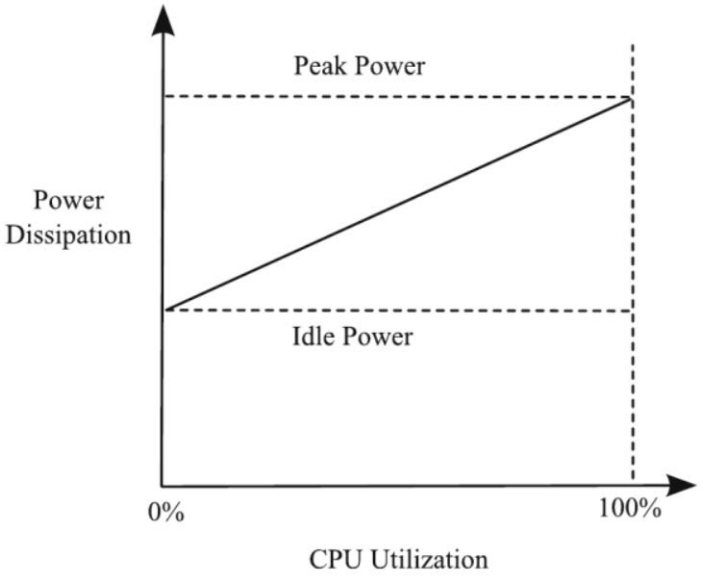

FIGURE 2.6: Server power model based on CPU utilization.
A linear model serves as a good approximation [1].

More complex power models enter into further details and present deeper analysis of power consumption. More parameters like network access rate, hard disk access rate and memory access rate are considered and implicated. Examples of these models are presented in [45], [46], [47] and [48].

Power modeling for VMs:
Virtual machines power estimating is important to better organize and schedule them in a way that minimizes the data center energy consumption.

Like the estimation models used for servers, CPU utilization could also be used to calculate the power consumption of the CPU by a VM [49] [50].

Models relying on information such as the resource utilization (CPU and memory) or/and on information provided by performance monitoring counters (PMC) known also as hardware performance counters (HPC) have been proposed in [51], [52], [53] and [54]. Based on the idea of combining PMC and CPU utilization, authors in [46] present a VM power metering approach and a software of VM power estimation called Joulemeter. This latter has the ability to accurately infer the power consumption without adding any additional hardware or software instrumentation.

Power modeling for VM migration:
Virtual machine live migration consists in moving VM between physical hosts without service interruption. This mechanism allows VM consolidation to achieve better energy efficiency however it brings also additional power consumption and its cost in terms of energy is not negligible [55].

Energy cost of migration have not been almost considered when migrating VMs. Key points for efficient VM consolidation are how to estimate the energy consumption of each VM migration and how to take migration decisions [56].

Some studies have been performed in [57], [55], [58], [59] and [60] to investigate the energy cost of VM migration and to model it. The energy overhead of live migration depends essentially on the amount of memory used by the VM and on the available network bandwidth. It increases with an increasing VM size and decreases with an increasing network bandwidth that influences it the most. Author in [61] proposed a lightweight mathematical model to estimate the energy cost of VM live migration. The model is derived through linear regression and the relationship between the energy cost of migration, the network bandwidth and the VM size is expressed in Eq. 2.4 where s represents VMs size, b represents the network bandwidth and A, B and C represent constant values.

$$E_{mig} = A + B * s + C * b \tag{2.4}$$

Energy efficiency metrics:
In addition to power models, improving energy efficiency in Cloud data centers requires metrics that reflect data centers and servers' efficiency and provide the necessary information for high level management and scheduling decisions.

Some metrics of energy efficiency have been proposed for data centers. The Green Grid [62] defined two data centers efficiency metrics : Power Usage Effectiveness (PUE) and Data Center Efficiency (DCE). Power Usage Effectiveness (PUE) is defined as the total power consumed by the data center divided by the power used by the IT equipment, as shown in Eq. 2.5:

$$PUE = \frac{TotalFacilityPower}{ITEquipementPower} \tag{2.5}$$

Data center Efficiency (DCE) is the indicator ratio of IT data center energy efficiency and is defined as the reciprocal of PUE (see Eq. 2.6).

$$DCE = \frac{1}{PUE} = \frac{ITEquipementPower}{TotalFacilityPower} \tag{2.6}$$

These two metrics measures only the proportion of power used by IT equipment and can be used to compare data center efficiency. Energy efficiency metrics for servers that could be used to sort them according to their efficiency and to enable scheduling algorithms to make decisions have not been widely investigated.

Performance per Watt (PPW) has became a popular metric as it can be used to measure and rank the energy efficiency of servers. It can be defined as the rate of transactions or computations that can be delivered by a computer for every watt of power consumed. Formally the PPW is defined by Intel [63] as : " *The term performance-per-watt is a measure of the energy efficiency of a computer architecture or a computer hardware. It can be represented as the rate of transactions or computations or a certain performance score that can be delivered by a computer for every watt of power consumed*". This metric provides scores and rank servers no matter their size or structure. The higher the performance per watt, the more energy efficient the server is.

2.3.4 Power saving policies in Cloud

The main power saving strategies in Cloud data centers are dynamic frequency voltage scaling (DVFS), servers powering down and VM consolidation.

Dynamic frequency and voltage scaling (DVFS):
Dynamic voltage frequency scaling (DVFS) is a power management tool that aims to reduce the power consumption of servers when the load is low [64]. DVFS, also known as CPU throttling, scales dynamically the voltage and frequency of the CPU at run-time. For example, Linux kernel allows for DVFS that can be activated in different policies: Performance, PowerSave, User-Space,Conservative, and OnDemand. Each policy has a governor that decides whether the frequency must be updated or not [65].

As this method decreases the number of instructions the processor executes in running a program, the program took a longer time and the performance reduce [66]. DVFS is also too dependent on hardware and is not controllable according to the changing needs, its resulting power savings are low compared to other methods.

Even if DVFS aims at reducing power consumption, it just acts at server level. As a completely idle server still consumes up to 70% of power, DVFS power savings remain narrow. These reasons have led to the appearance of other data center level solutions that consolidate workloads onto fewer servers and switch off or put in lower power mode the idle hosts.

Powering down:
Important reduction in energy consumption can be achieved by powering down or switching off servers when they are not in use. As many servers in the data center are idle most of the time, they could be powered down or put into sleep mode in the periods of time when they are not used and then powered up if needed.

This dynamic capacity provisioning or dynamic shutdown problem is challenging as it requires careful planning to select servers to power down and as different factor must be considered. On/Off approaches have been proposed in [67], [68], [69], [70] and [71] to dynamically turn on and off data center servers and thus minimizing the energy use. Although its complexity, this technique is efficient and can achieve significant reduction in power consumption.

Energy aware consolidation:
A key technique of power saving in Cloud data centers is workload consolidation onto a smaller number of servers. This approach aims to reduce the high consumption of energy by selecting the most energy efficient servers [21].

Dynamic Optimization and further workload consolidation into an even fewer number of server can be performed thanks to VM live migration. It is an essential mechanism that dynamically moves virtual machines to different hosts without rebooting the operating system inside the VM.

Energy aware consolidation problem for Cloud has been significantly studied in the literature. A detailed overview will be provided in the next chapter to present the related works in the area.

2.4 Research orientation and focus

This thesis deals with the problem of energy efficient resource allocation in Cloud data centers. We aim at reducing the power consumption of data centers by reducing the power consumption of servers. We focus essentially on energy aware consolidation techniques and optimization models that minimize the number of active servers in order to increase the energy efficiency of Cloud data centers. To quantify power consumption and energy efficiency we rely on power and energy estimation models as well as energy efficiency metrics.

Both classic Cloud service models and new hybrid models are considered and targeted. We aim to bring energy efficiency to the commonly used and widespread IaaS providers and to support also the new trend of hybrid IaaS/PaaS Cloud providers. On-demand and advanced reservation plans are also important aspects that we consider when allocating resources to users.

2.5 Conclusions

This chapter introduced the concepts of Cloud computing and virtualization and investigated the problem of energy efficiency in Cloud. We presented the major causes of energy waste in Cloud data centers, presented the energy measurement

and modeling methodologies and described the power saving techniques in Cloud data centers. This chapter has also concluded with a discussion of the orientation and focus of this thesis.

The next chapter explores in more details the problem of resource allocation or scheduling in Cloud. We provide background and state of the art solutions for energy efficient resource allocation. Then, we discuss the related issues and problems, as well as the challenges.

Chapter 3

Background & Related Work on Energy Efficient Cloud Resources Allocation

3.1 Introduction

As mentioned earlier in the report, the main objective of this thesis is the design and development of models and algorithms for energy efficient resource allocation in Cloud data centers while considering different dimensions of the problem. These key dimensions are the resource provisioning plan, the dynamicity of the solution, the type of the virtualization and the Cloud service model.

To provide efficient solutions, to address the issue from different angles and to handle the constraints of the problem at different levels, existing state of the art methods and models need to be studied and discussed.

This chapter presents the current state of the art and work of the areas related to this thesis. We describe in more details the problem of energy efficient resource allocation in Cloud data centers then we provide an overview on the state of the art of energy efficient Cloud resource allocation at different dimensions and levels. The chapter also presents the research objectives and the thesis positioning in relation to existing research.

3.2 Energy Efficient Resource Allocation in Cloud

Resource allocation or scheduling is one of the most important tasks in cloud computing. It consists in identifying and assigning resources to each incoming user request in such a way that the user requirements are met and specific goals of the cloud provider are satisfied. These goals could be optimizing energy consumption or cost optimizing, etc. Based on the resource information like resource usage and monitoring, the requests information and the Cloud provider goal, the resource allocator or scheduler finds out resource allocation solutions, see Figure 3.1. Schedulers could just ensure the initial and static resource allocation after request arrival or ensure both static and dynamic resource allocation to manage resources in a continuous way and to further optimize and readjust the old requests.

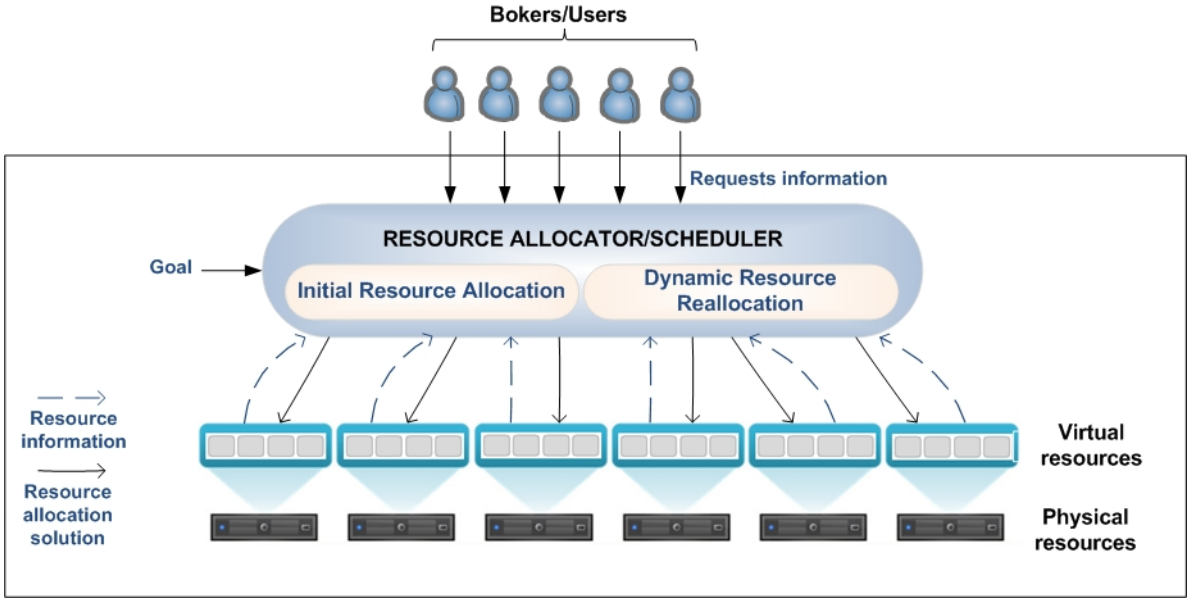

FIGURE 3.1: Resource Allocation in Cloud Computing

The wider adoption of cloud computing and virtualization technologies has led to cluster sizes ranging from hundreds to thousands of nodes for mini and large data centers respectively. This evolution induces a tremendous rise of electricity consumption, escalating data center ownership costs and increasing carbon footprints. For these reasons, energy efficiency is becoming increasingly important for data centers and Clouds.

Solving the problem of resource allocation in Cloud while maximizing energy efficiency is a very challenging issue. This problem is known as NP-hard and has been studied in the context of Cloud computing. The objective of this chapter is to review the existing literature regarding energy efficient resource allocation in Cloud. Different important dimensions will be considered in our literature study. These dimensions cover the type of the resource provisioning plan, the Cloud service model and also the static or dynamic aspects of the solutions.

3.3 On-demand resource allocation vs advanced resource reservation

Cloud providers could offer different kinds of provisioning plans. The most two important ones are on-demand and reservation plans. The on-demand plan allows users to access resources at the time when they need. In Reservation plan the resources could be reserved earlier and the resource availability is ensured in future.

On-demand resource allocation:

Most of the Cloud providers rely on simple policies like on-demand (immediate) to allocate resources. These solutions allocate the resources if available, otherwise the requests are not accepted.

In [21], [72],[73] and [74], authors proposed energy-aware heuristic algorithms and policies in order to save energy by minimizing the number of running servers. The key idea is to consolidate applications or tasks on a minimum number of servers to switch off machines in surplus. Another study is presented in [75] where the authors presented a nature-inspired VM consolidation algorithm influenced by Ant Colony Optimization. This algorithm aims also at reducing the number of used physical machines and thus saves energy.

All the above work discusses how to reduce energy consumption of cloud data centers using on-demand and immediate algorithms for energy efficient resource allocation. These algorithms are derived for homogeneous data centers that embed monitoring capabilities and probes (e.g smart power distribution units (PDUs)) or that embed power consumption estimation tools. Or most of today's data centers are considered mega data centers (composed of heterogeneous servers[76]) and still lack energy monitoring capabilities. More details on on-demand Cloud resources

allocation algorithms are given in the next section where the solutions are classified into static and dynamic.

Advanced resource reservation:

Advance resource reservation provides simple means for resource planning in the future and offers an increased expectation that resources can be allocated when demanded. Although advance reservation of resources in Cloud is very advantageous, most of the Cloud providers use simple resource allocation policies like on-demand and best effort that did not incorporate the dimension of time and support future planning of resource provisioning.

Haizea scheduler [77] is an open source resource lease manager. It supports four kinds of resource allocation policies: immediate, best-effort, advance reservation (AR) and deadline sensitive. AR lease is requested by users when they need to use infrastructure for fixed start and end time of lease. Resource reservation is achieved by a mapping function that uses a slot table which has two dimensions: the physical nodes and the duration. This mapping function takes a set of requested resources and maps them to physical servers based on the availability in the slot table in a specified time interval. Haizea uses also a greedy algorithm to determine how VMs are mapped to servers. This latter sorts servers from lower to higher loaded. Then, it traverses the list of nodes and tries to map as many lease nodes into each server before moving on to the next. The existing scheduling algorithms in Haizea are simple, greedy and do not address energy efficiency [78].

Advance resource reservation algorithms for IaaS infrastructure as a Service are proposed in [79] and [80]. These are queuing models based algorithms that check whether enough resources are available or not for the requested duration. They only aim at resource reservation and disregard energy efficiency requirements.

3.4 Static vs dynamic Cloud resources allocation

Two different types of resource allocation are static and dynamic allocation. Static resource allocation is performed initially when requests arrive. Dynamic resource allocation is used to manage resources in a continuous way and to further optimize and readjust the old requests. The dynamic resource allocation or consolidation

is handled by VM live migration and aims to minimize the number of used or activated servers.

Energy efficient algorithms for initial Cloud resources allocation:

Currently, resource allocation mechanisms used in Cloud data centres include load balancing, round robin and greedy algorithms. The existing scheduling algorithms used by OpenNebula [81], Eucalyptus [82] and OpenStack [13] Cloud managers are greedy or simple round robin based and do not address energy efficiency.

Authors in [83] propose a simple energy-aware policy incorporating allocation schemes of virtual servers to achieve the aim of green computing. The considered allocation schemes are round robin, first fit, etc. This work saves energy by setting servers to the lower power consumption state when they do not host VMs. The proposed policy governs servers to a low-energy consuming state when they are idle and manages them into the operating state of full functionality when they are used.

The works in [84], [85], [86], [87] and [88] try to save energy by proposing policies for dynamically powering servers on and off. These policies are based on queuing models and heuristic-based methods are presented. An approach based on Dynamic Voltage Frequency Scaling (DVFS) is proposed in [89]. This proposed work focus on scheduling virtual machines to reduce power consumption via the technique of DVFS.

The energy efficient algorithms proposed in [72] and [90] go further by adopting consolidation policies that strives to use a minimal number of servers to accommodate all requested VMs. Both works have proposed heuristics for the bin packing problem as algorithms for VMs consolidation.

Authors in [21] consolidate applications or tasks on reduced number of physical machines to switch off machines in surplus. They propose a heuristic for multidimensional bin packing and show that using less physical hosts can save energy consumption. Authors in [91] present also a multi-tiered resource scheduling scheme that provides on-demand capacities to the hosted services via resources flowing among VMs. A global resource flowing algorithm was introduced to optimize resource allocation among applications. Both approaches are achieved at the task level and hence fit better the Platform or Software as a Service (PaaS, SaaS)

levels. Allocation or placement is also static as opposed to dynamic placement according to workload where migration is applied to reallocate resources.

Energy efficient algorithms for VMs migration:

In [92], the authors present a power-aware server consolidation framework, called pMapper that continuously optimize the VM placement to minimize power consumption. It relies on greedy heuristics for bin packing problem and it introduces the cost of VM migration but without providing information about its calculation. Another similar framework called Entropy is proposed in [93]. It is a resource manager for homogeneous clusters that performs dynamic consolidation based on constraint programming and it takes migration overhead into account.

Reference [72] addresses policies for dynamic VMs reallocation using VMs migration according to CPU performance requirements. Their most effective policy, a double threshold policy, is based on the idea of setting upper and lower utilization thresholds for hosts and keeping the total utilization of the CPU of all the VMs between these thresholds. If the CPU utilization of a host exceeds the upper threshold, some VMs are migrated and if it falls below the lower threshold, all the hosted VMs should be migrated.

Authors in [94] treat the problem of consolidating VMs in a server by migrating VMs with steady and stable capacity needs. They proposed an exact formulation based on a linear program described by a too small number of valid inequalities. Indeed, this description does not allow solving, in reasonable time and in an optimal way, problems involving allocation of a large number of items (or VMs) to many bins (or Servers). In order to scale and find solutions for large sizes, the authors resorted to a heuristic using a static and a dynamic consolidation of VMs to reduce energy consumption of the hosting nodes or servers.

In [73], authors presented a server consolidation (Sercon) algorithm which consists of minimizing the number of used nodes in a data center and minimizing the number of migrations at the same time. They compared their algorithm with the heuristic FFD (First-Fit Decreasing) [95] that solves the Bin-Packing problem and have shown the efficiency of Sercon to consolidate VMs and minimize migrations. However, Sercon is a heuristic that can not always reach or find the optimal solution.

In [74], authors presented an approach EnaCloud for dynamic live placement taking into account energy efficiency in a cloud platform. They proposed an energy-aware heuristic algorithm in order to save energy by minimizing the number of running servers. Another study relying on dynamic resource allocation is presented in [75]. The authors presented a nature-inspired VM consolidation algorithm inspired from an Ant Colony Optimization. This algorithm aims at reducing the number of used physical machines and thus saves energy.

Authors in [96] propose two heuristic algorithms for energy-aware virtual machine scheduling and consolidation. These algorithms are respectively based on a dynamic round-robin approach (DRR) and on an hybrid one which combines DRR and First-Fit. Another VM consolidation method for power saving in data centers that relies on the bin packing First-Fit heuristic is proposed in [97]. This method migrates VMs on the basis of server ranks where the rank represents server selection priority and is uniquely assigned to each server.

3.5 IaaS vs hybrid IaaS/PaaS Cloud providers

Scheduling and energy efficiency have been discussed and investigated in IaaS Clouds. Almost all of the related works presented in the two last sections are dedicated for the IaaS level.

As the well known PaaS solutions for service orchestration like Windows Azure [6], Google App Engine [7], and Heroku [98] are not open source and provide a black-box solution for the public Cloud, some open-source projects like CloudFoundry [99] and OpenShift [100] that provide private PaaS are becoming more and more popular. These PaaS systems can be built on IaaS and conduct to construct PaaS on IaaS.

In fact, new hybrid Cloud solutions that combine Iaas and PaaS like OpenStack Heat [13] are evolving over time and being more and more attractive since they enable the joint deployment of infrastructure and applications. Thanks to this hybrid IaaS-PaaS solutions, users can deploy their own applications with the topology they choose and with having the control on both infrastructure and programs. Like classical PaaS solutions, the new hybrid IaaS-PaaS solutions are using LXC containers [25] and Docker [14] that are radically changing the way applications are built, shipped, deployed, and instantiated. However, these solutions still lack

energy efficient resource (VM or container) scheduling and no attention was paid to solve the problem at this level.

OpenStack Heat is an openstack service that handles the orchestration of complex deployments on top of OpenStack clouds. Orchestration basically manages the infrastructure but it supports also the software configuration management. Heat provides users the ability to define their applications in terms of simple templates. This component has also enabled OpenStack to provide a combined IaaS-PaaS service. Orchestrating Docker containers in OpenStack via Heat provides orchestration of composite cloud applications and accelerates application delivery by making it easy to package them along with their dependencies (this mechanism is described in details in Appendix B). Even if this approach based on Docker integration into OpenStack is very advantageous and provides users with complete services and with more control, OpenStack Heat is still based on static assignment and requires VM and container scheduling. The energy efficiency was also completely disregarded.

A new container as a service solution called Kubernetes [12] was released by Google to manage containerized applications across multiple hosts and to provide basic mechanisms for deployment of applications. Kubernetes's scheduler is currently very simple and relies on a first-come-first-served (FCFS) algorithm. No attention was paid to energy efficiency in this solution.

3.6 Scope and positioning of the thesis

Table 3.1 presents a summary of the comparison between relevant related works. We compare the various research efforts in terms of provisioning plan, Cloud service level, virtualization category, dynamicity and power saving methods.

Proposed solutions of initial Cloud resource allocation [72], [90] and of VM migration at IaaS level [92], [97], [96], [75], [74], [73], [94], [72], [93] are heuristic based and can not reach or find the optimal solution. Another important aspect which was not always considered when moving VMs is the energy cost of migration. This cost should be taken into account before making decisions as migration brings additional power consumption and its cost in terms of energy is not negligible [55].

As the comparison table shows, hybrid IaaS/PaaS solutions still lack energy efficient policies that schedule both VMs and containers to provide users with complete services. Cloud resource assignment is static or simple and no attention was paid to energy efficiency.

Most of the proposed solutions are based on policies for the on-demand plan to allocate resources. Advance resource reservation has received less attention and existing solutions [79], [80], [78], [77] are based on simple heuristics and do not consider energy efficiency. However, this concept has many advantages especially for the co-allocation for resources. Advance reservation provides simple means for resource planning and reservation in the future and offers an increased expectation that resources can be allocated when demanded.

TABLE 3.1: Related work comparison

	Provisioning Plan	Service model	Virtualization Category	Static vs Dynamic	Power Saving
[84], [85], [86], [87], [88]	On-demand	IaaS	Hypervisor based	Static	Powering down
[72], [90]	On-demand	IaaS	Hypervisor based	Static	Consolidation
[21], [91]	On-demand	PaaS	Hypervisor based	Static	Consolidation
[77], [78]	On-demand and advanced reservation	IaaS	Hypervisor based	Static	No power saving
[79], [80]	advanced reservation	IaaS	Hypervisor based	Static	No power saving
[92], [97], [96], [75], [74], [73], [94], [72], [93]	On-demand	IaaS	Hypervisor based	Dynamic	Dynamic Consolidation
[12], [13]+[14]	On-demand	IaaS/PaaS	Hypervisor and container based	Static	No power saving

This thesis investigates the problem of energy efficient Cloud resources allocation. We aim at reducing the power consumption of data centers by reducing the power consumption of servers. We focus essentially on energy aware consolidation techniques and optimization models that minimize the number of active servers in order to increase the energy efficiency of Cloud data centers. To quantify power

consumption and energy efficiency we rely on power and energy estimation models as well as energy efficiency metrics.

The first objective of our work is to propose, develop and evaluate optimization algorithms of resource allocation for traditional IaaS architectures that are widely used to manage clouds. The approach is VM based and it should enable on-demand and dynamic resource scheduling while reducing the power consumption of the data center. We propose algorithms that are based on exact formulations of the consolidation problem and of the VM migrations to optimally consolidate VMs in servers while minimizing the energy cost of migrations.

This initial objective is naturally extended to deal with the new trends in Cloud. We aim to provide a new model and optimization algorithms of energy efficient resource allocation for IaaS-PaaS Cloud providers. The solution should be generic enough to support different type of virtualization technologies, to enable both on-demand and advanced resource provisioning plans, to deal with dynamic resource scheduling and to fill the gap between IaaS and PaaS in order to create a single continuum of services for Cloud users.

3.7 Conclusions

This chapter described the main research efforts in the area of energy efficient Cloud resource allocation. We mainly focus on the reservation plan dimension to classify the related work. Dimensions of type of the virtualization type and the Cloud service model are also considered in the discussion. This chapter presented also the thesis position in relation to existing work.

The main direction of this thesis is the design and development of models and algorithms for resource allocation in Cloud data centers while increasing energy efficiency. The next chapters describe in detail our contributions for this research direction.

Chapter 4

Bin packing based Approach for Energy Efficient Resource Allocation

4.1 Introduction

Cloud data centers are electricity guzzlers especially if resources are permanently switched on even if they are not used. An idle server consumes about 70% of its peak power [34]. This waste of idle power is considered as a major cause of energy inefficiency. An important way to bring energy efficiency to Cloud environments is to introduce energy aware scheduling and placement algorithms and enhanced resource management.

This work is a contribution to the reduction of such excessive energy consumption using energy aware allocation and migration algorithms to have a maximum number of idle servers to put into sleep mode. Intel's Cloud Computing 2015 Vision [63] stresses also the need for such dynamic resource scheduling approaches to improve power efficiency of data centers by shutting down and putting to sleep idles servers. This work proposes an exact energy aware allocation algorithm using the formulation of the Bin-Packing problem. The aim of this algorithm is to reduce the number of used servers or equivalently maximize the number of idle servers to put in sleep mode. To take into account workloads and service times a linear integer programming algorithm is used to optimize constantly the number of

used servers after service departures. This migration algorithm is combined with the exact allocation algorithm to reduce overall energy consumption in the data centers.

The proposed algorithms act as an energy consumption aware VM scheduler and can be used to enhance current infrastructure managers and schedulers such as OpenNebula [81] and OpenStack [13]. The power consumption indicators can be provided by energy consumption estimation tools such as joulemeter [46]. A dedicated simulator is used to assess performance and crosscheck with the performance results produced by the exact algorithms. Evaluation results show that the exact allocation algorithm combined with migration reduces considerably the number of required servers to serve a given load and can thus minimize power consumption in data centers.

4.2 The System Model

The model considers infrastructure providers allocating physical resources instances to host users' and tenants' applications or, equivalently for this work, VMs. The physical resources are seen as servers. It is assumed that applications are packaged into virtual machines to be hosted by the infrastructure providers. The cloud providers save energy and reduce power consumption by packing and consolidating through migration of VMS to maximize the number of idle servers to put to sleep mode.

Figure 4.1 depicts the system model composed of the proposed energy efficient allocation and migration algorithms (contributing to scheduling), an energy consumption estimator and a cloud manager (handling infrastructure resource instantiation and management). Each module is briefly described to set the stage for the analytical modeling of the energy efficient resource allocation problem in clouds.

- **Cloud IaaS manager** (e.g. OpenStack [13], OpenNebula [81] and Eucalyptus [82]) control and manage cloud resources and handle clients requests, VM scheduling and fetch and store images in storage spaces.
- **Energy estimation module** is an intermediate module between the cloud infrastructure manager and the energy-aware scheduler. The module can rely

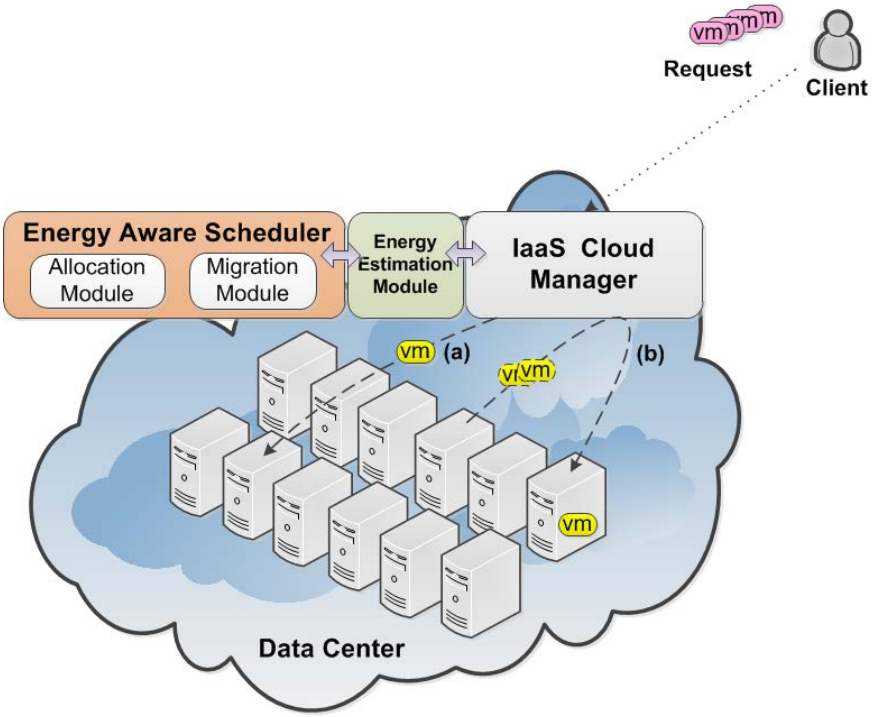

FIGURE 4.1: The system model

for example on an energy estimation tool such as Joulemeter [46] that uses power models to infer power consumption of VMs or servers from resource usage.

- **Energy-aware VM scheduler** responsible for the energy aware VM placement in the data center is the focus of our energy consumption optimization model. This green scheduler is basically composed of two modules. An allocation module and a migration module. The role of the allocation module is to perform the initial VM placement using our exact VM allocation algorithm. The dynamic consolidation of virtual machines is handled by the migration module that minimizes the number of used or activated servers thanks to our exact VM migration algorithm. The unused servers are shut down or put into sleep mode. All the needed information (servers and VMs) to run the algorithms are retrieved via the Cloud IaaS manager that is also used to execute the VM deployment and migration actions.

To derive the system model, we consider the size n of client requests in terms of the number of required VMs and the types of desired VM instances (e.g., small,

medium, large). Each VM_i is characterized by a lifetime t_i and a maximum power consumption p_i. Each server or hosting node j, from the data center, has a power consumption limit or power cap noted $P_{j,Max}$. This can be fixed by Cloud administrators. We assume that all servers are homogeneous; extending the model to heterogeneous servers is trivial but will increase complexity and will not necessarily provide additional insight.

The approach adopted to achieve energy efficiency in our proposal is to use a bin packing algorithm for optimal placement of user requests and to follow with dynamic consolidation once a sufficient number of departures have occurred. The dynamic consolidation is handled by the migration algorithm which regroups VMs to free as many servers as possible to put them into sleep mode or to shut them down.

4.3 Energy Efficient Static Resource Allocation

4.3.1 Exact Allocation Algorithm

The proposed exact VM allocation algorithm is an extended Bin-Packing approach through the inclusion of valid conditions expressed in the form of constraints or inequalities. The objective is to pack items (VMs in our case) into a set of bins (servers or nodes hosting the VMs) characterized by their power consumptions. In addition to n, the number of requested VMs, we define the number of servers, m, available in the data center. The servers are assumed to have the same power consumption limit: $P_{j,Max}, \{j = 1, 2, ..., m\}$. At run-time, each server j hosting a number of VMs is characterized by its current power consumption: $P_{j,current}$.

Since the objective is to minimize the energy consumption of the data centers, we define as key decision variable e_j for each server j that is set to 1 if server j is selected to host VMs, 0 if it is not selected. In addition, we define the bivalent variable x_{ij} to indicate that VM_i has been placed in server j and set x_{ij} to 1; $x_{ij} = 0$ otherwise. The objective function to place all the demands (or VMs) in a minimum number of servers can be expressed using:

$$\min Z = \sum_{j=1}^{m} e_j \tag{4.1}$$

This optimization is subject to a number of linear constraints reflecting the capacity limits of the servers and obvious facts such as a VM can only be assigned to one server or a server can only host VMs according to the amount of remaining resources:

1. Each server has a power limit $P_{j,Max}$ that cannot be exceeded when serving or hosting VMs and this occurs according to remaining capacity:

$$\sum_{i=1}^{n} p_i x_{ij} \leq P_{j,Max} e_j - P_{j,Current}, \forall j = 1, \ldots, m \tag{4.2}$$

2. A cloud provider has to fulfil all requests within a prescribed SLA or quota and each requested VM will be assigned to one and only one server:

$$\sum_{j=1}^{m} x_{ij} = 1, \forall i = 1, \ldots, n \tag{4.3}$$

3. For servers verifying the condition $P_{j,Max} > P_{j,current}$ and $P_{j,current} \neq 0$, the total number of used servers is lower bounded by $\left\lceil \frac{\sum_{j=1}^{m} P_{j,current}}{P_{j,Max}} \right\rceil$. This adds he following inequality to the model:

$$\sum_{j=1}^{m} e_j \geq \left\lceil \frac{\sum_{j=1}^{m} P_{j,current}}{P_{j,Max}} \right\rceil \tag{4.4}$$

The exact and extended Bin-Packing VM allocation model can be summarized by lumping the objective function with all the constraints and conditions into the following set of equations:

$$\min Z = \sum_{j=1}^{m} e_j \tag{4.5}$$

Subject To:

$$\sum_{i=1}^{n} p_i x_{ij} \leq P_{j,Max} e_j - P_{j,Current}, \forall j = 1, \ldots, m \tag{4.6}$$

$$\sum_{j=1}^{m} x_{ij} = 1, \forall i = 1, \ldots, n \tag{4.7}$$

$$\sum_{j=1}^{m} e_j \geq \left\lceil \frac{\sum_{j=1}^{m} P_{j,current}}{P_{j,Max}} \right\rceil \tag{4.8}$$

$$e_j = \begin{cases} 1, & \text{if the server } j \text{ is used;} \\ 0, & \text{otherwise.} \end{cases} \tag{4.9}$$

$$x_{ij} = \begin{cases} 1, & \text{if the } VM_i \text{ is placed in server } j; \\ 0, & \text{otherwise.} \end{cases} \tag{4.10}$$

All the variables and constants used in the model are listed for easy reference below:

- n is the size of the request in number of requested VMs.
- m is the number of servers in the data center.
- p_i represents the power consumption of VM_i.
- x_{ij} is a bivalent variable indicating that VM_i is assigned to a server j.
- e_j is a variable used to indicate whether the server j is used or not.
- $P_{j,Max}$ represents the maximum power consumption of server j.
- $P_{j,current}$ represents the current power consumption of server j ($P_{j,current} = P_{j,idle} + \sum_k p_k$ with VM_k hosted by server j).
- $P_{j,idle}$ represents the power consumption of server j when it is idle.

Constraints in server CPU, memory and storage are also added to the model to confine even further the model convex hull:

$$\sum_{i=1}^{n} cpu_i x_{ij} \leq CPU_j e_j \tag{4.11}$$

where cpu_i is the requested CPU by VM_i. CPU_j is the CPU capacity of server j.

$$\sum_{i=1}^{m} mem_i x_{ij} \leq MEM_j e_j \tag{4.12}$$

where mem_i is the requested memory by VM_i and MEM_j is the memory capacity of server j.

$$\sum_{i=1}^{m} sto_i x_{ij} \leq STO_j e_j \tag{4.13}$$

where sto_i is the requested storage by VM_i and STO_j is the storage capacity of server j.

In this work we assume that these constraints are met and verified and we hence only need to focus on the energy efficiency constraints through (4.2).

4.3.2 Modified Best Fit Heuristic Algorithm

The exact and extended Bin-Packing is compared to a Best-Fit heuristic adaptation of the Best-Fit algorithm [95]. The heuristic proposed to achieve energy efficient VM placement consists of two steps:

1. sorting the requested VMs in decreasing order of power consumption. This builds somehow an ordered stack that is used in the second step for packing VMs in available servers;

2. The sorted VMs are handled starting from the top of the stack and attempting to place the most power consuming VMs in the server with the smallest remaining power consumption budget until a VM down the stack fits in this target server. The process repeats or continues until all VMs in the stack are placed and packed as much as possible in the most occupied servers. This will tend to free servers for sleep mode or switching off.

As this Best-Fit heuristic algorithm tries to approximate the Bin-Packing algorithm, it is selected for comparison with our exact VM allocation proposal. The allocation algorithms are combined with a migration algorithm to minimize overall data center power consumption. In our case, the objective is to benchmark the exact VM allocation and migration algorithms with a heuristic algorithm. The Best-Fit heuristic was selected since it is known to achieve good suboptimal performance compared with classical Bin-Packing.

4.4 Energy Efficient Dynamic Resource Allocation (Re-allocation)

4.4.1 Exact Migration Algorithm

The placed and running VMs in the servers will gradually leave the system as their related jobs end. These departures are the opportunity to re-optimize the placement by migrating VMs always in the system for consolidation in a minimum number of fully packed severs. A migration algorithm based on an integer linear program (ILP) is presented to achieve the consolidation. This ILP algorithm consists in introducing a number of valid inequalities to reduce the span of the convex hull of the migration problem.

The mathematical model for the VM consolidation via migration relies on a linear integer programming formulation. The objective for the algorithm is to migrate VMs from nodes selected as source nodes (those the algorithm aims at emptying so they can be turned off) to other selected destination nodes (those the algorithm aims at filling so they serve a maximum number of VMs within their capacity limits).

Ideally, the algorithm should minimize the number of active nodes, maximize the overall number of VMs handled by the active nodes and hence maximize the number of unused, empty or idle nodes. The algorithm should also minimize the power consumption caused by migrations. If the power consumption or cost of VM migration is uniform or homogeneous across hosting nodes or servers, the objective reduces to minimizing the number of migrations.

The migration concerns the set of non idle servers m', $m' < m$, whose power consumptions are lower than $P_{j,Max}$ with j in m'. Despite the slight reduction in size $m' < m$, the problem remains NP-hard. Hence, we resort to an exact algorithm based on linear integer programming to address optimal migration for practical problem sizes or number of instances.

The objective function for the optimal VM migration and consolidation can be expressed as the maximization of the number of idle servers in the infrastructure:

$$\max M = \sum_{i=1}^{m'} P_{i,idle} y_i - \sum_{i=1}^{m'} \sum_{j=1}^{m'} \sum_{k=1}^{q_i} p'_k z_{ijk} \tag{4.14}$$

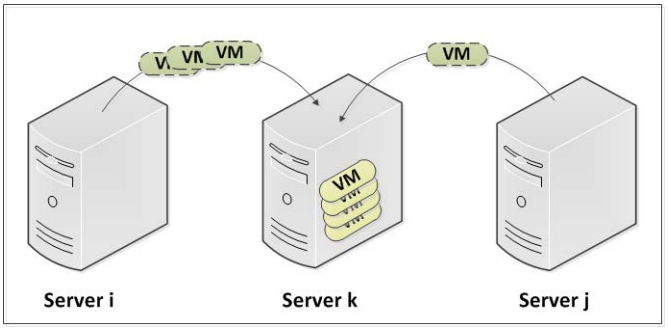

FIGURE 4.2: Example of VMs' migration

where $y_i = 1$ is used to indicate that server i is **idle** and $y_i = 0$ means that at least one VM is active in server i. $P_{i,idle}$ is the power consumed by idle servers, p'_k is the cost in terms of consumed power when migrating VM_k.

Variable z_{ijk} is the bivalent variable expressing migration of VM_k from server i to server j. Variable q_i is the total number of VMs hosted on server i and that are candidate for migration into destination servers, especially server j in equation (4.14).

The objective function (4.14) is subject to the migration constraints cited earlier. These conditions are formally expressed through valid inequalities and constraints that have to be respected when minimizing overall energy consumption.

1. When migrating VM_k from a server i to a server j (see figure 4.3), the algorithm must prevent backward migrations and can only migrate into one specific destination node. Stated in an equivalent way: if a VM_k is migrated from a server i (source) to a server j (destination), it can not be migrated to any other server l ($l \neq j$). The proposed inequality (4.15) also ensures that VMs in destination node and VMs migrated to destination nodes are not migrated as we are aiming at filling these nodes instead of emptying them obviously. This is reflected by the inequality :

$$z_{ijk} + z_{jlk'} \leq 1; \tag{4.15}$$

2. To strengthen further the previous condition, a valid inequality is added to ensure that when a VM_k is migrated from server i to server j (see figure 4.4), migrations to other nodes l ($l \neq j$) are prevented or forbidden:

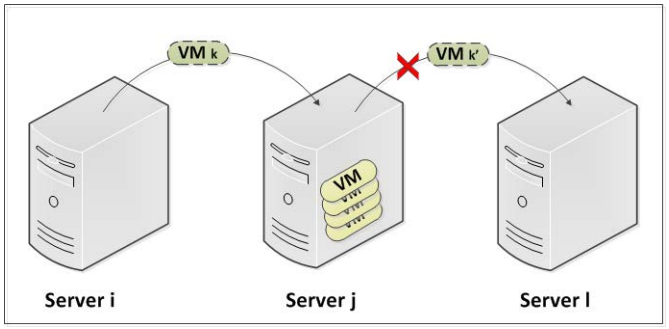

FIGURE 4.3: A server candidate to a migration should not migrate its own VMs

$$\sum_{j=1, j\neq i}^{m'} z_{ijk} \leq 1; \tag{4.16}$$

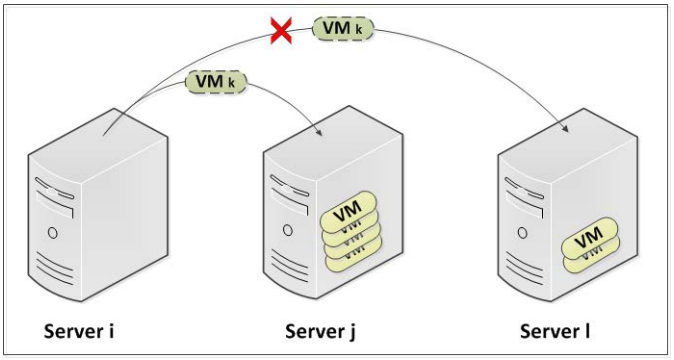

FIGURE 4.4: A VM_k can not be migrated to many servers at the same time

3. A server j is limited by its power consumption limit $P_{j,Max}$. The inequality (4.17) allows each server j to host VMs without exceeding its power limit:

$$\sum_{i=1}^{m'} \sum_{k=1}^{q_i} p_k z_{ijk} \leq (P_{j,Max} - P_{j,Current})(1 - y_j) \tag{4.17}$$

Where $P_{j,Current}$ is the current power consumption of server j.

4. If a non-idle server i is a source of VM migration, then it should migrate all of its hosted VMs in order to be put to sleep mode or shut down once completely emptied:

$$\sum_{j=1}^{m'} \sum_{k=1}^{q_i} z_{ijk} = q_i y_i, \forall i = 1, \ldots, m', j \neq i \tag{4.18}$$

5. Another valid inequality is the upper bound in the total number of empty servers:

$$\sum_{i=1}^{m'} y_i \leq m' - \left\lceil \frac{\sum_{j=1}^{m'} P_{j,Current}}{P_{j,Max}} \right\rceil \tag{4.19}$$

6. Another important aspect is to avoid migration of VMs whose lifetime or leftover lifetime t_k is shorter than the time needed to make migration decisions T_0:

$$z_{ijk} \Delta t_k \geq T_0, \tag{4.20}$$

where $\Delta t_k = t_k - CurrentTime$, where $CurrentTime$ represents current or VM migration handling time.

The optimal VM consolidation and migration model and objective function (4.14) can be summarized for convenience with all the valid conditions as:

$$\max M = \sum_{i=1}^{m'} P_{i,idle} y_i - \sum_{i=1}^{m'} \sum_{j=1}^{m'} \sum_{k=1}^{q_i} p'_k z_{ijk} \tag{4.21}$$

Subject To:

$$z_{ijk} + z_{jlk'} \leq 1 \tag{4.22}$$

$\forall i = 1, \ldots, m', \forall j = 1, \ldots, m', \forall k = 1, \ldots, q_i, \forall k' = 1, \ldots, q_j, j \neq i$, and $\forall l = 1, \ldots, m', l \neq j$, $k \neq k'$.

$$\sum_{j=1, j \neq i}^{m'} z_{ijk} \leq 1 \tag{4.23}$$

$\forall i = 1, \ldots, m'$,
$\forall j = 1, \ldots, m', \forall k = 1, \ldots, q_i$, $\forall l = 1, \ldots, m', l \neq j$.

$$\sum_{i=1}^{m'} \sum_{k=1}^{q_i} p_k z_{ijk} \leq (P_{j,Max} - P_{j,Current}) (1 - y_j) \tag{4.24}$$

$\forall j = 1, \ldots, m, j \neq i$

$$\sum_{j=1}^{m'} \sum_{k=1}^{q_i} z_{ijk} = q_i y_i, \forall i = 1, \ldots, m', j \neq i \tag{4.25}$$

$$\sum_{i=1}^{m'} y_i \leq m' - \left\lceil \frac{\sum_{j=1}^{m'} P_{j,Current}}{P_{j,Max}} \right\rceil \tag{4.26}$$

$$z_{ijk} \Delta t_k \geq T_0, \tag{4.27}$$

$$z_{ijk} = \begin{cases} 1, & \text{if the } VM_k \text{ is migrated from a server } i \text{ to a server } j; \\ 0, & \text{otherwise.} \end{cases} \tag{4.28}$$

$$y_i = \begin{cases} 1, & \text{if the Server } i \text{ is idle;} \\ 0, & \text{otherwise.} \end{cases} \tag{4.29}$$

4.5 Combination of allocation and migration algorithms

Figure 4.5 summarizes how the allocation and migration algorithms are combined to achieve minimal energy consumption in infrastructure nodes and hence data centers. Both the exact Bin-Packing extension and the Best-Fit heuristic are used to ensure optimal and suboptimal placement respectively.

Recall that the two algorithms allow us to cross check their relative performance and benchmark the modified Best-Fit with the proposed exact allocation solution since the algorithms exhibit different optimality and convergence time characteristics.

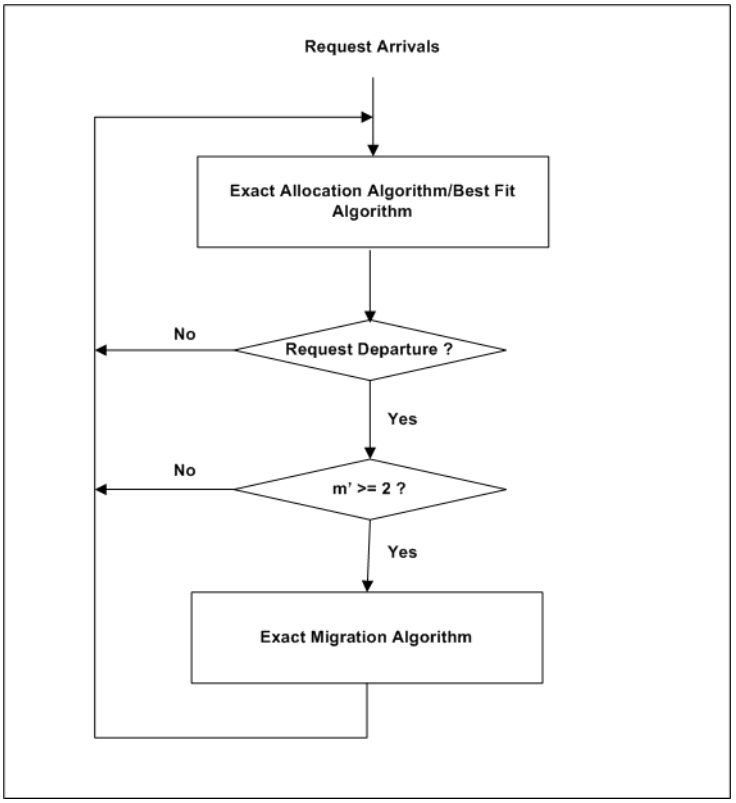

FIGURE 4.5: Combination of the migration algorithm with the two allocation algorithms

Upon arrival of VM placement and resource requests these two algorithms select the most appropriate nodes to host new VMs in available and active nodes. Whenever necessary these algorithms may resort to turning new nodes on, when the set of active nodes are full and cannot host the new arriving VM instances (small, medium or large). Both algorithms will attempt serving the requests in the currently active nodes and will of course typically avoid turning any new nodes on.

The algorithms are combined with the migration algorithm that is launched if a number of VM jobs terminate since their dedicated resources become available for opportunistic reuse and for more efficient resource allocation and distribution. These departures are the opportunity for the consolidation algorithm to rearrange allocations by moving VMs into the smallest possible set of nodes. All emptied or freed servers (or nodes) are turned off to minimize energy consumption.

The consolidation is achieved by the exact migration algorithm that moves VMs from selected source nodes to selected destination nodes. The end result is the

activation and use of the smallest set of nodes in the data centers.

4.6 Performance evaluation

Our proposed algorithms are evaluated through a Java language implementation and the linear solver CPLEX [101]. A dedicated simulator is developed to conduct the performance assessments and the comparison. The objective of the numerical evaluation is to quantify the percentage of energy savings or power consumption savings that can be expected when combining the exact allocation algorithm and the consolidation process using our proposed exact migration algorithm. The answers provided by the numerical analysis concern also the scalability and complexity of the proposed algorithms in the size of the data centers and the arrival rate of requests for resources to host VMs which is also synonymous to load on the system. Note, however, that the simulation are conducted for an arrival rate strictly lower than the rate of VM job departures from the system; thus simulations correspond to cases where the likelihood of finding an optimal or a good solution is high.

The assessment scenarios correspond to data centers with 100 servers or nodes for the first five experiments. In the last two experiments 200 servers are considered. We collect essentially as performance indicators, the percentage of used servers (which automatically provides the energy consumed or saved by the algorithms) and the time required for the algorithms to find their best solutions (optimal for the exact algorithms). All the servers have a power consumption cap $P_{j,Max}$ set to 200 watts (the peak power of a typical server is around 250 watts [102]). To perform per-VM power estimation we referred to a power estimation model proposed in [103]. Three $SPECcpu2006$ [104] workloads (454.calculix, 482.sphinx and 435.gromacs) with high, medium and low power consumption were considered. Their associated power consumption is close to 13 watts, 11 watts and 10 watts respectively. The power estimation model proposed in [46] provided additional insight. The power consumption of other $SPECcpu2006$ [104] workloads (471.omnetpp, 470.lbm and 445.gobmk) were evaluated. Estimated power consumptions were found to be between 25 and 28 watts for these elements. Without loss of generality and to ease intuitive verification of the results, we refer to these published consumption to associate to each VM type (small, medium and large) an energy consumption p_i respectively equal to 10 watts (low), 20 watts (medium)

and 30 watts (high) to stay in line with published values. The requests for resources to serve VMs have a constant arrival rate. The requested VM instance types are discrete uniform in $[1, 3]$ (1 for small, 2 for medium and 3 for large instances). The VM sizes are arbitrarily drawn as uniform in $[1, 3]$ and classified according to their type. We retained only the random drawings that fulfill the VM characteristics in size and type.

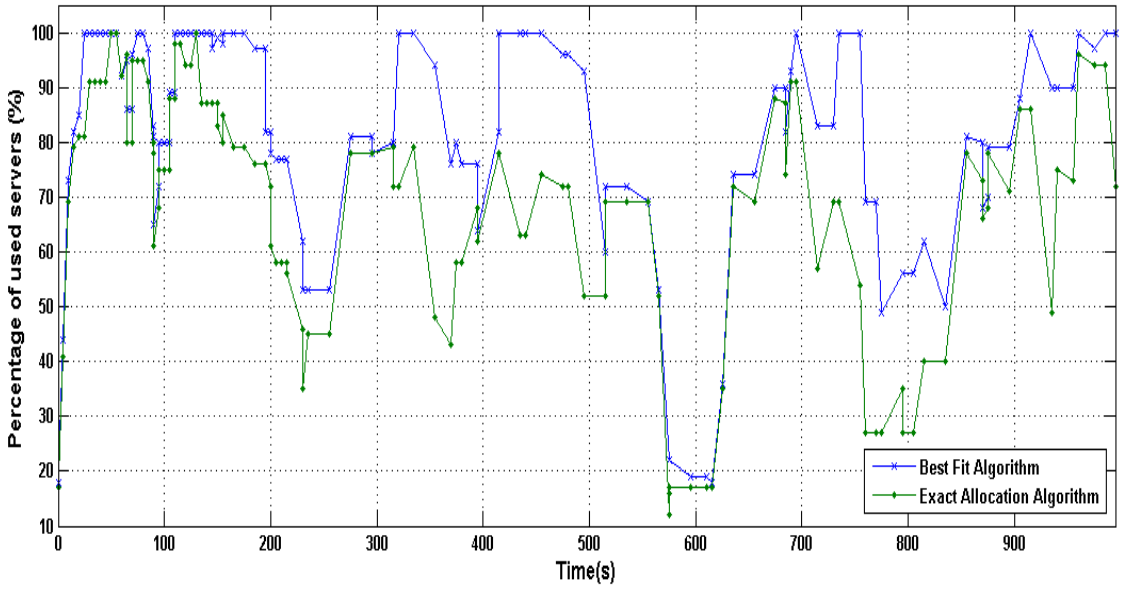

FIGURE 4.6: Comparison between the exact and heuristic allocation algorithms

Figure 4.6 depicts results of a comparison between the adapted Best-Fit heuristic and our exact extended Bin-Packing allocation algorithms. The simulations correspond to 100 servers and resource requests in number of VMs in the $[1, 200]$ range. The lifetime of the VMs are uniform in $[30s, 180s]$. That is VM jobs last at least $30s$ and will terminate in less than $180s$. The exact allocation algorithm as expected outperforms the Best-Fit heuristic for the $1000s$ time interval simulated and reported in Figure 4.6. The Best-Fit heuristic uses more often all available nodes or servers (100% ordinate value in Figure 4.6) while the exact algorithm manages to use fewer nodes with 10 to 50% more unused servers that Best-Fit.

Figure 4.7 extends the analysis for the exact and extended Bin-Packing allocation algorithm by comparing its performance with and without consolidation. When the exact algorithm is combined with the migration algorithm (that uses migration to empty some nodes) it can significantly provide additional power savings or

energy consumption reduction. The average gain can be estimated to be 10 to 20% more servers that could be turned off. The average line for the exact algorithm is around 80% of servers used while the average for the exact algorithm with migration is more in the order of 60%.

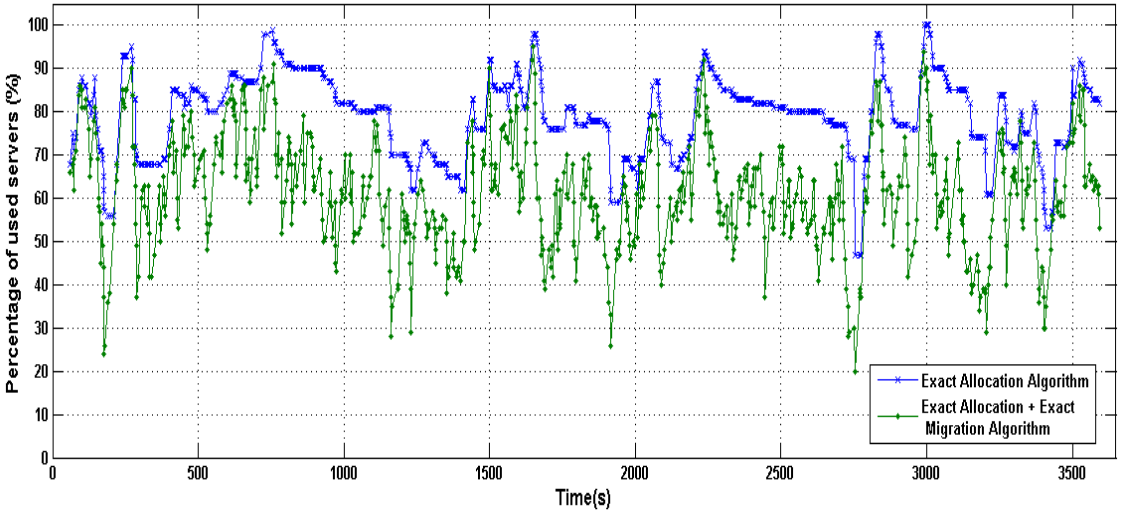

FIGURE 4.7: Performance comparison of the exact allocation algorithm with and without migration

Figure 4.8 pursues the analysis for the exact bin-Packing VM allocation algorithm by reporting performance as a function of data center sizes and VM requests induced load. The time before convergence to the optimal placement is reported as a function of data center size (from 100 to 1000 nodes or servers) for request sizes ranging from 50 to 500 VMs. Clearly, because the problem is NP-Hard, the convergence time of the exact algorithm grows exponentially for requests exceeding 300 VMs; especially for number of servers beyond 400. The time needed to find the optimal solutions remains acceptable and reasonable, within 10 s, for data center sizes below 500 receiving requests less than 400 VMs. The time needed for convergence grows unacceptably high outside of this operating range for the simulated scenarios (tens of seconds to few minutes). This motivated the use of the Best-Fit algorithm to find solutions faster even if they are bound to be suboptimal as reported in Figure 4.6.

Figures 4.9, 4.10 and 4.11 address the performance of the consolidation algorithm via the analysis of the time needed to achieve migrations of VMs from source nodes to destination nodes in order to free as many servers as possible and gain

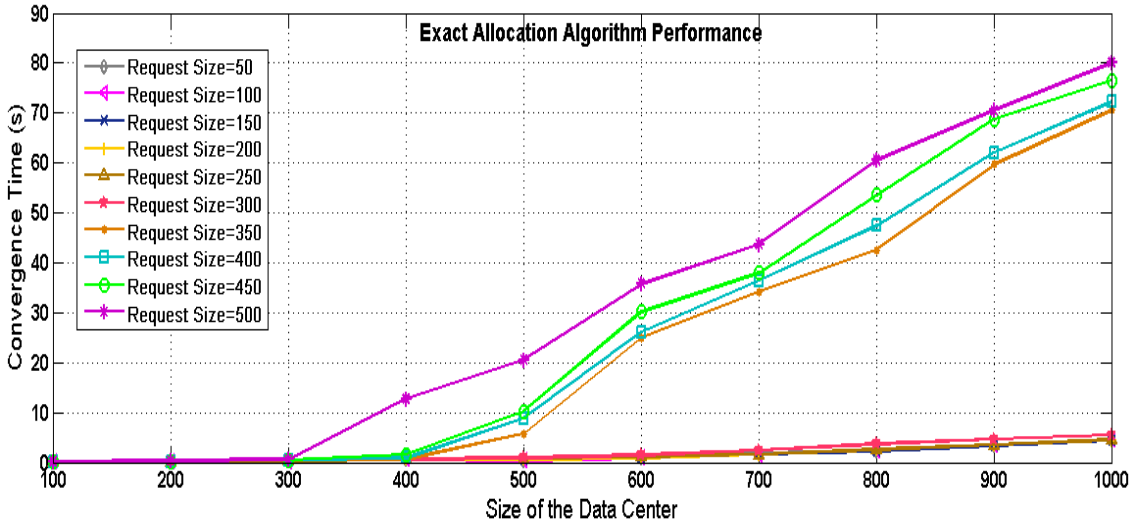

FIGURE 4.8: Execution time of the Exact Allocation Algorithm

the opportunity to shut them down. The assessment is performed consequently on the active servers, i.e. those currently serving VMs and candidate for consolidation. The performance as a function of increasing number of active nodes m' to consolidate is reported in the three figures for $m' = 5, m' = 10$ and $m' = 20$. For $m' = 5$, consolidation after migration from source to destination nodes can be achieved in the milliseconds time scales (few to 300 ms in Figure 4.9). The number of hosted VMs to consolidate varies from 5 to 30 for this simulation.

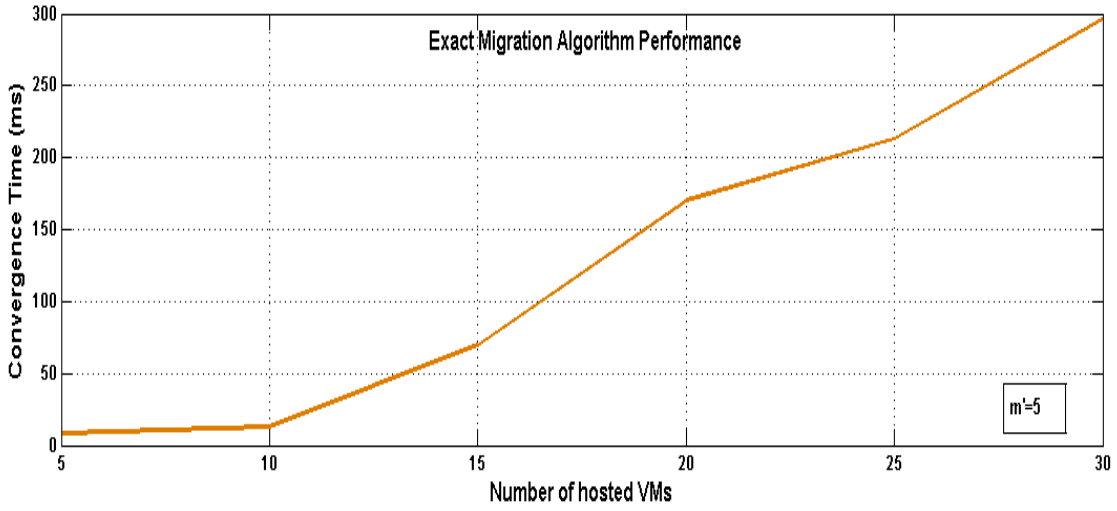

FIGURE 4.9: Execution time of the exact migration algorithm ($m' = 5$)

The time needed for consolidation increases to seconds in Figure 4.10 for $m' = 10$. The curve is reported for up to 60 VMs hosted in the m' nodes considered or subject to consolidation/migration. When the number of servers to consolidate increases further, as shown in Figure 4.11 for $m' = 20$, the convergence times move to orders of tens to hundreds of minutes (for the extreme case, on the curve upper right corner, this reaches 180 minutes for 120 hosted VMs). These three figures highlight the limits of the exact migration algorithm with increasing number of servers to consolidate.

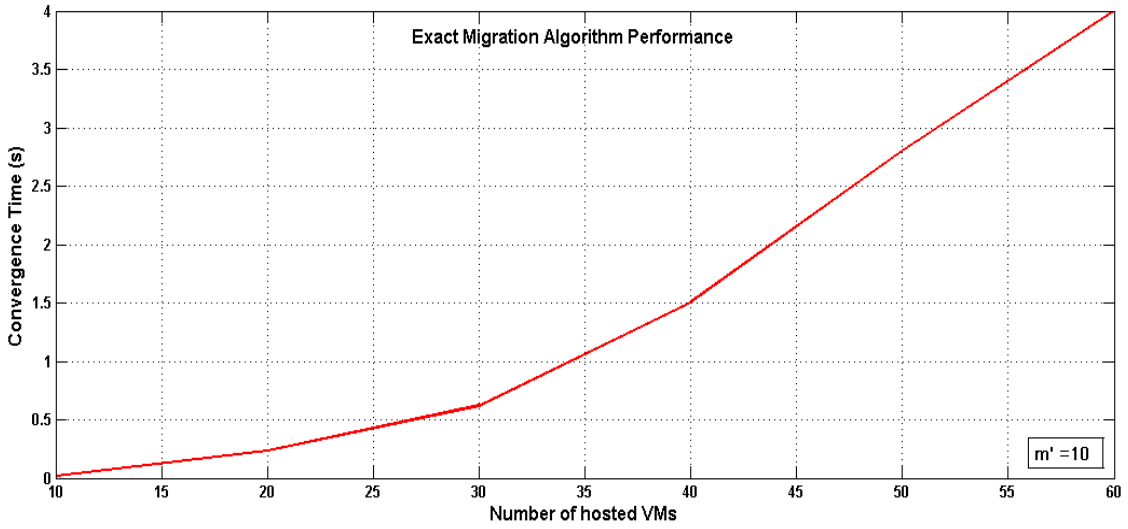

FIGURE 4.10: Execution time of the exact migration algorithm ($m' = 10$)

The next experiments and simulations address the achievable energy savings using different VM requests inter-arrival times (noted by λ^{-1}) and lifetimes (represented by μ^{-1} that also reflects the service rate μ^{-1}) in order to assess the performance for variable system loads since the ration λ/μ governs performance. The number of servers has been fixed to 200 hosting nodes for the reported results in Table 5.2. One hundred (100) simulation runs are average for each parameter setting in the table. Table 5.2 reports the energy savings with the migration algorithm compared to the allocation algorithm without migration.

Energy savings depend evidently on the service rate or the lifetime of VMs or the duration of their jobs relative to the load induced by the VM resource requests. Savings in the simulation can reach as high as 41.89% for inter-arrival times of 25 seconds and job durations of 30 seconds. For less favorable ratios or loads, the savings for the scenarios tested in the evaluation are less significant but remain

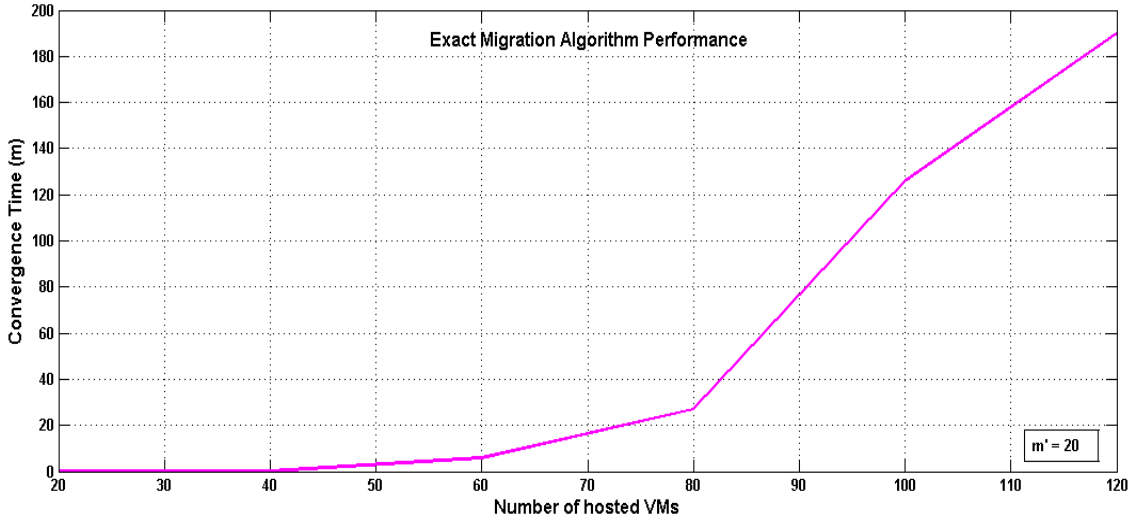

FIGURE 4.11: Execution time of the exact migration algorithm ($m' = 20$)

TABLE 4.1: Table of percentage of gained energy when migration is used

$\mu^{-1}(s)$ \ $\lambda^{-1}(s)$	5	10	15	20	25	30
10	35,55	36,59	00,00	00,00	00,00	00,00
20	27,29	34,00	35,23	38,50	00,00	00,00
30	17,48	27,39	35,21	40,32	**41,89**	36,58
40	16,77	18,85	22,02	32,31	39,90	40,50
50	10,86	16,17	19,85	22,30	39,20	36,52
60	08,63	14,29	18,01	22,13	25,15	30,68
70	08,10	14,00	14,86	15,90	22,91	23,20
80	07,01	10,20	10,91	15,34	17,02	21,60
90	06,80	09,52	10,31	14,70	16,97	19,20
100	05,90	07,50	08,40	12,90	16,00	14,97

respectable (5.90% for the highest loads (inter-arrivals time is equal to 5seconds) and longer job durations (of 100sec).

In order to complete the analysis, the energy savings that can be achieved by the Best-Fit, the exact allocation and the exact allocation combined with migration are compared for similar scenarios with a restricted set of parameter settings ($\lambda^{-1} = 10s$). All the servers are initially considered OFF, which means that the energy saving is initialized to 100%. Figure 4.12 depicts the evolution of the percentage of energy saved by the three algorithms. The obvious dependence on system load is reflected by the gradual decrease of energy savings for increasing VM lifetimes

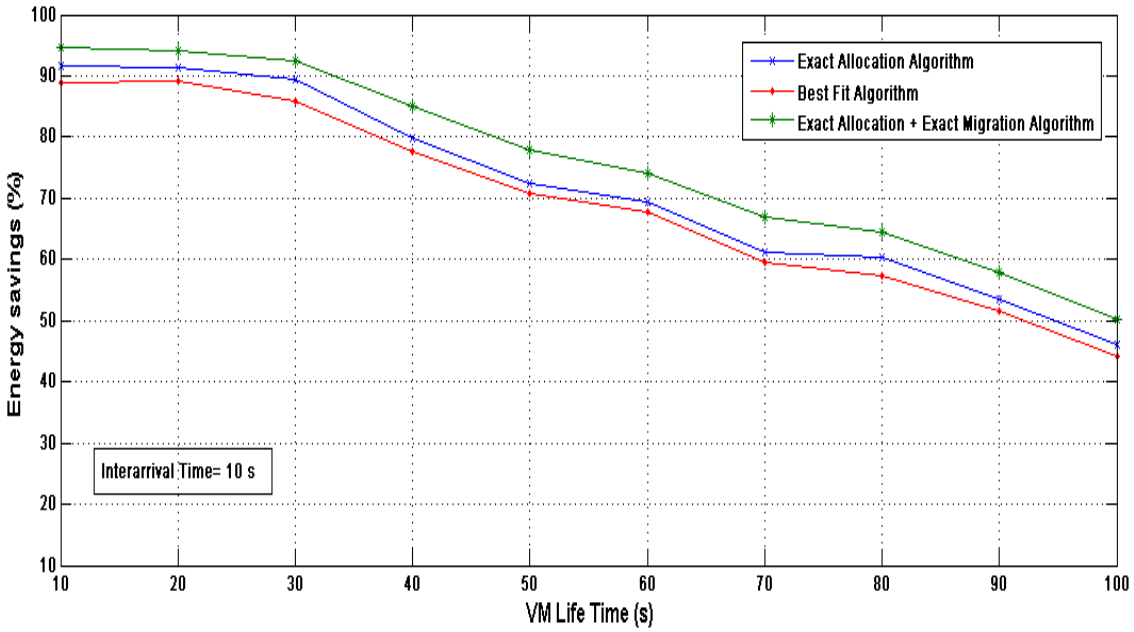

FIGURE 4.12: Energy savings

(or increasing job durations). For the exact Bin-Packing allocation algorithm, the energy savings remain quite high at low load; 90% of energy savings for the exact allocation only and 95% combined with migration. Energy savings achieved by the Best-Fit heuristic stay below the percentages achieved by the exact allocation algorithm with or without migration at all different loads.

4.7 Conclusions

In this chapter, we propose a bin packing based approach for energy efficient resource allocation for classical IaaS clouds. We formulate the problem of energy efficient resource allocation as a bin-packing model. This model is VM based and provides on-demand resource allocation. We propose an exact energy aware algorithm based on integer linear program (ILP) for initial resource allocation. To deal with dynamic resource consolidation, an exact ILP algorithm for dynamic VM reallocation was also proposed. It is based on VM migration and aims to optimize constantly the energy efficiency after service departures. A heuristic method based on best-fit algorithm was also adapted to the problem. Experimental results show benefits of combining the allocation and migration algorithms and

demonstrate their ability to achieve significant energy savings while maintaining feasible runtimes when compared with the best fit heuristic. The next chapter introduces a graph coloring-based approach to deal with the new trends in Cloud.

Chapter 5

Graph coloring based Approach for Energy Efficient Resource Allocation

5.1 Introduction

New hybrid Cloud solutions that combine Iaas and PaaS like OpenStack Heat [13] are evolving over time and being more and more attractive since they enable the joint deployment of infrastructure and applications. However, most of the interest was on the IaaS Clouds. These solutions still lack energy efficient resource scheduling and no attention was paid to solve the problem at this level.

In IaaS clouds, the focus has been mostly on smart placement and optimal packing for efficient resource utilization including in some cases an energy efficiency criterion [72] [105]. The time dimension has received less attention when, in fact, it is expected that users or consumers will acquire virtual resources from providers for a specified time interval to take advantage of the cloud flexibility and cost reduction benefits. Energy efficient advance resource reservation (or scheduling) combined with optimal placement has not been as thoroughly investigated. The starting and ending time of user requested cloud services have to be taken into account and combined with energy consumption minimization criteria in order to avoid resource reservation conflicts and collisions for concurrent or overlapping requests.

This is the focus of our work where we take into account advance resource reservation and time conflicts while simultaneously aiming at improved energy efficiency and resource utilization for providers. We aim to provide a generic model and optimization algorithms of energy efficient resource allocation that could be applied by IaaS-PaaS cloud providers. Another goal is to dynamically optimize placement while avoiding conflicts when allocating resources to users by preventing the assignment of resources to concurrent requests.

We propose a new model based on graph coloring to prevent time conflicts combined with energy consumption minimization and resource utilization maximization criteria to achieve optimal and conflict free allocations. For the purpose we cast the advance resource reservation problem into a graph coloring problem and more specifically make use of graph pre-coloring and re-coloring to handle resource requests and resources releases. Graph coloring is NP-complete and is defined as coloring the vertices of a graph with the minimum number of colors without any two adjacent vertices having the same color. Graph coloring was used in various research areas of computer science such data mining [106], image segmentation [107], register allocation [108], timetabling [109], frequency assignment [110] and aircraft maintenance scheduling [111]. Graph coloring fits well with the problem of advance resource reservation since it can be made to ensure non conflicting resource reservations (when consumers can not use the same resource simultaneously or share the resource in the same time interval). It fits also on-demand resource allocation if the start time is immediate the end time is not fixed.

In our proposed model, improvements in energy efficiency are achieved by privileging reservation of more energy efficient resources in priority. To prioritize the selection of servers we rank them according to their performance per watt PPW, a measure of the energy efficiency of a computer architecture or a computer hardware defined in [112]. The higher the performance per watt, the more energy efficient the computer is. Our model is generic enough to use alternate energy consumption or energy efficiency metrics and will remain relevant if another metric is adopted.

Starting from the graph coloring technic combined with the energy efficiency metric, this work propose a new generic graph coloring model of energy efficient advance resource reservation in IaaS-PaaS cloud data centers. The proposed model provides users with access to a set of resources for a specified time while minimizing resource usage and maximizing data centers' energy efficiency. It could also

deal with on-demand resource allocation if the start time is immediate the end time is not fixed.

For initial resource reservation (after request arrival), we provide an integer linear programming formulation which generalizes the graph coloring problem. Then, we derive an Energy Efficient Graph Pre-coloring EEGP heuristic algorithm to deal with larger graph instances and to slow computation times. To adapt reservations when resources are released, we propose two heuristic algorithms called Energy Efficient Graph Recoloring (EEGR) and Migration Aware Energy Efficient Graph Recoloring (MA-EEGR) that reassign resources after services end. We compare our proposed algorithms with the baseline advance reservation algorithm (AR) used in the Haizea resource manager [77] and show that our heuristics achieve significantly better results in terms of energy efficiency and resource utilization.

5.2 The System Model

Our proposed model is based on graph coloring [113] which is defined as follows: given a graph G = (V, E) and an integer k, a proper k-coloring of a graph G is an assignment of distinct k colors to each vertex such that two adjacent vertices have not the same color. The least k such that G is k-colorable is called the chromatic number and denoted by $\chi(G)$. To derive our proposed model, we model virtual resources as **colors**, we translate VM requests or demands into a **graph** G and we relate them to graph coloring and the energy efficiency metric.

5.2.1 Resource Modeling: Colors

We consider a cloud data center with m heterogeneous virtualized servers. We assume resources are exposed in the form of virtual resource units (VRUs). A **VRU** is an abstraction of resources that is characterized by its computational, memory, and communication capacities and by its availability. VRU could represent a virtual container for hosting one instance of an application or could simply represent a compute unit like ECU Amazon EC2 Compute Unit [5]. Each VRU is modeled as a **color** $c_{j,id}$ where j corresponds to the server providing this VRU and id specifies the id of the VRU. Colors belonging to the same server j form a

cluster of colors C_j. If at least one color of the color cluster C_j is reserved, we consider that C_j is used, else it is free.

To each C_j, we associate a weight w_j which represents PPW (performance per watt) of server j. As already mentioned, we adopt for the servers power efficiency metric which can be defined as the rate of transactions or computations that can be delivered by a computer for every watt of consumed power. This measure is becoming an increasingly important metric for data centers [114]. Manufacturers of servers such as Intel, AMD and Original Equipment Manufacturer (OEM) favor the performance per watt metric over more straightforward performance metrics.

5.2.2 End User Request Modeling : Request Subgraph

The end user request $R_k = \{VM_1, ..., VM_{n_k}\}$ expresses a demand of n_k VMs. Each VM_i requires a specific amount r_i of resource units. VM_i is logically divided into r_i requested resource units (defined as RRUs). Each RRU will be assigned to a unique VRU (see Figure 5.2 and Figure 5.3). The selected VRU is reserved for the associated VM_i for the duration of its specified start to stop interval: $[a_i, b_i]$. Hence, all RRUs of a specific VM are assigned to the same reservation interval.

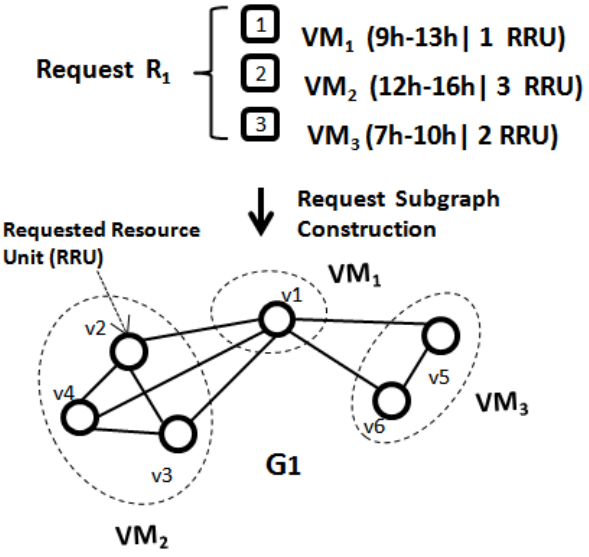

FIGURE 5.1: Request Subgraph Construction

In order to cluster the RRUs of a given VM in the assignment or reservation process, we define the bivalent variable $x_{u,c}$, and set its value to 1 to indicate if an

RRU u is assigned to a VRU or a color c and equal to 0 otherwise. This leads us to associate to each request R_k a subgraph G_k with vertex set V_k (or set of RRUs) and a set of edges E_k. The edges connect vertices (RRUs) that have overlapping reservation time intervals. This will enable conflict free scheduling of resources so virtual resources are assigned to one and only one request in a end user specified interval.

An example of request subgraph construction, after arrival of the first request R_1, is shown in Figure 5.1. VM_1, VM_2 and VM_3 requirements in terms of resource units (RRUs) are respectively 1, 3 and 2. Hence, one RRU v_1 is associated to VM_1, three RRUs (v_2, v_3, v_4) are associated to VM_2 and two RRUs (v_5, v_6) are associated to VM_3. Edges or links between the nodes v_i (or RRUs) indicate that these connected nodes should be assigned strictly different colors.

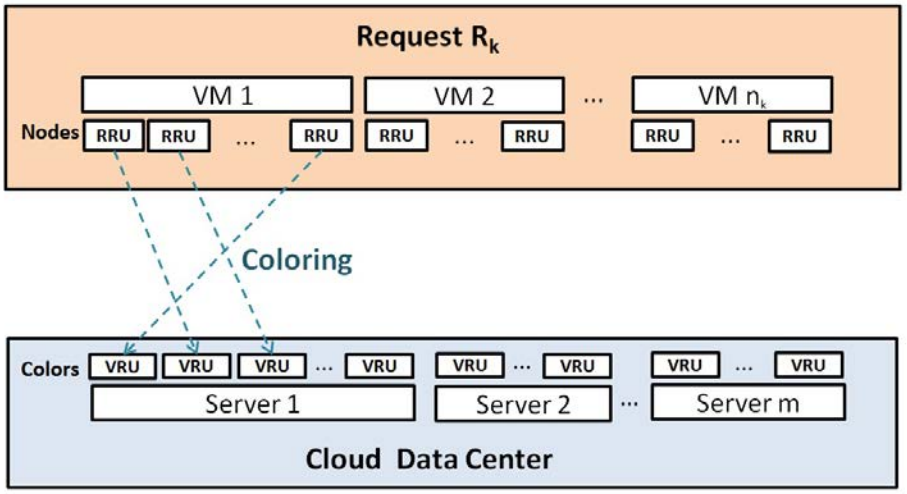

FIGURE 5.2: Graph coloring based model

To give a description of how to apply our generic graph-coloring based model on hybrid IaaS-PaaS Clouds, we present a concrete example of model building when a user wants to deploy a LAMP application on a virtual machine using OpenStack Heat and Docker (see Appendix B for more details).

A LAMP application is a web application based on the composition of Linux, Apache, MySQL, and PHP. In this scenario, we consider a user request for building a LAMP application on an isolated VM that contains a docker container running Apache with PHP and another one running MySQL database (see Figure 5.4). The user request is presented as a template that specifies the different

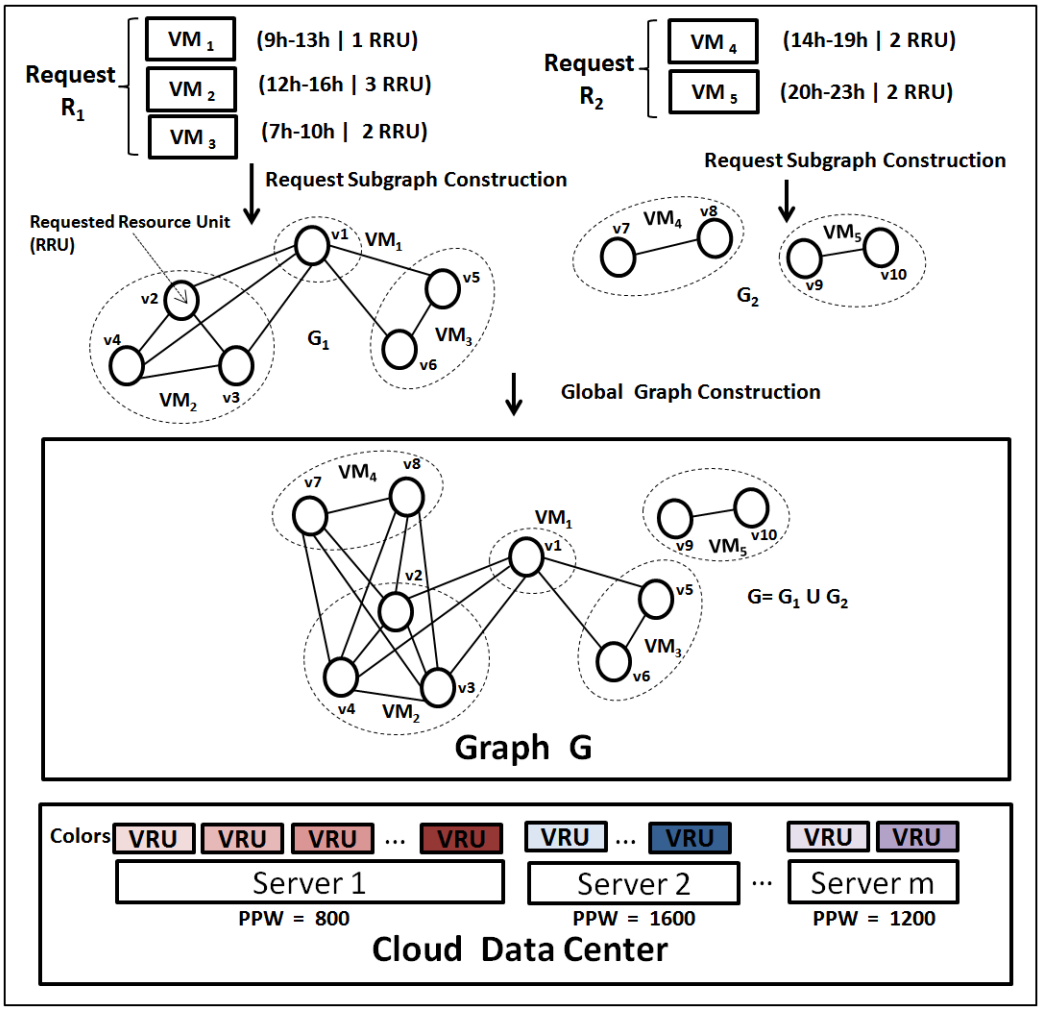

FIGURE 5.3: Graph Coloring Model

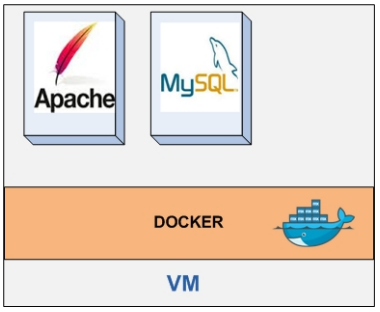

FIGURE 5.4: Lamp application deployment on a VM

resources (VM and containers). This latter is modeled as subgraph where nodes (RRUs) represent requested applications that will be running on containers and links indicate overlapping reservation time intervals. Containers represent colors or VRUs exposed by the provider (see Figure 5.5).

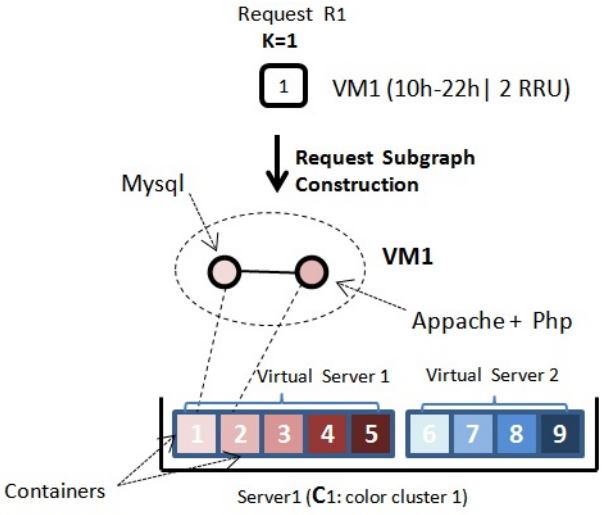

FIGURE 5.5: Model building in case of a LAMP application deployement

5.2.3 Energy efficiency metric

The performance per watt (PPW) is an increasingly used metric to assess the energy efficiency of data centers, supercomputers, servers or hardware [114]. Intel, AMD and Original Equipment Manufacturer (OEM) use the PPW [112] as the energy efficiency metric to measure and rank the energy efficiency of computers. It can be defined as the rate of transactions or computations that can be delivered by a computer for every watt of power consumed. A server PPW tends to vary with its resources usage or solicitation. Traditionally, a server PPW is established and measured at full load (e.g. Intel). This is the metric considered in our model.

The PPW metric is used to select in priority the servers that have the highest reported PPW values. Less efficient servers will be less used or solicited and can hence be shutdown or put to sleep mode to save energy.

5.2.4 Graph coloring for Energy Efficient Resource Reservation

Conflicts between all requested resources are derived from an undirected global dynamic **graph** G. $G = (V, E)$ is dynamically constructed over time and updated after request arrivals and departures. Vertex set V represents RRUs belonging to all requested VMs and E represents the set of all edges in the graph. In case of a new request arrival (request R_k), new nodes and edges will appear on the graph. Thus, the global graph ($G = G \cup G_k$) will represent a conjunction of G and G_k. In case of a request reservation end, G_k will be retrieved and deleted from the global graph G. This is depicted in the first step of Figure 5.6 and 5.8 illustrating how graph G is updated at request arrivals and service departures.

Once graph G is constructed, the next step is to color the graph while maximizing its average power efficiency by privileging the servers with the highest PPW performance when assigning VRUs:

$$\overline{PPW} = \frac{\sum_{h=1}^{|V|} w_j x_{h,c_{j,id}}}{\sum_{h=1}^{|V|} x_{h,c_{j,id}}} \tag{5.1}$$

Less efficient servers will be less used and could be shutdown or put to sleep mode in order to achieve more energy savings.

In addition to finding the chromatic number $\chi(G)$, the number of used color clusters (or servers) should be minimized. This will consolidate VRUs assigned to RRUs in a minimum number of servers. To these objectives, we associate a number of valid conditions and constraints to speed up convergence towards a viable solution. The RRUs of the same VM have to be assigned to VRUs (colors) belonging to the same server or color cluster.

The reservation of resources at new requests arrivals can be seen as a pre-coloring extension of the graph coloring problem, a generalization of graph coloring, since at each new request arrival we have a graph where a subset of the vertices already have a color and we have to extend this pre-coloring to the whole graph (see [113]). To handle the new resource reservation requests, we use the EEGP heuristic based algorithm to pre-color the graph so as to achieve no-conflict scheduling and

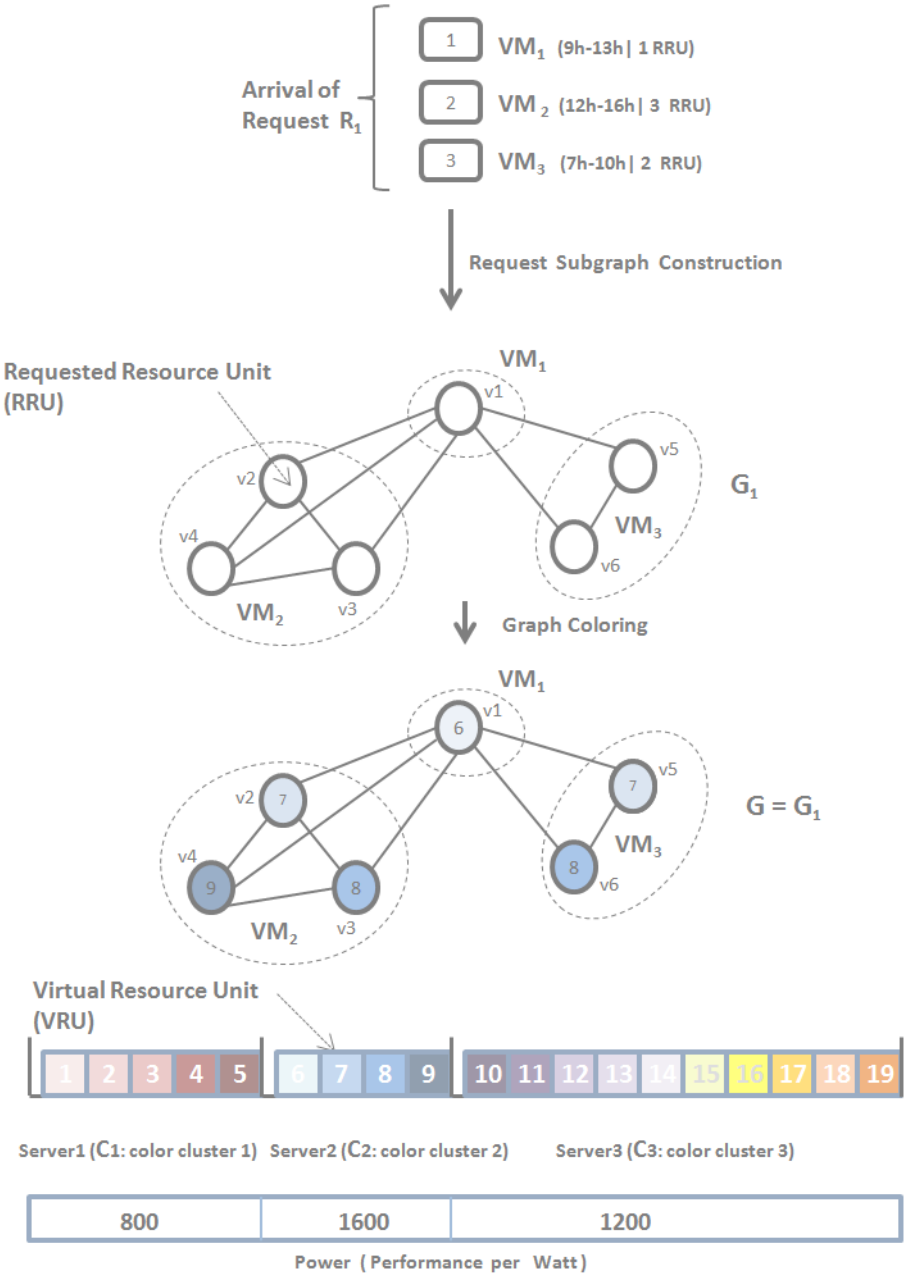

FIGURE 5.6: Graph Coloring (first request)

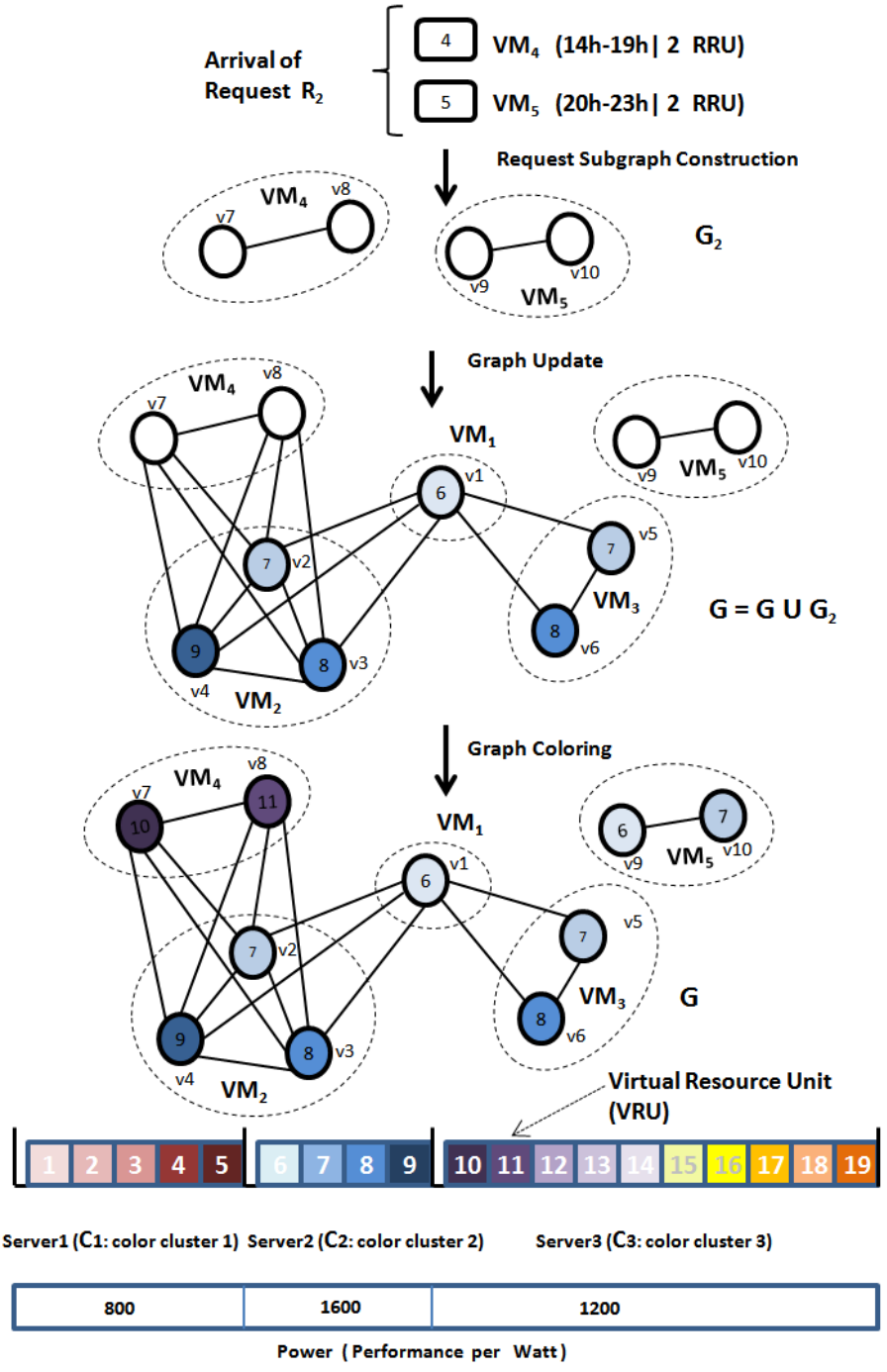

FIGURE 5.7: Graph Coloring (second request)

maximization of the average power efficiency or equivalently maximize the achieved $\overline{PPW}$.

To handle service departures, when VM jobs end and the assigned VRUs are released, we resort to heuristic algorithms, EEGR and MA-EEGR. Service departures are opportunities for reassignments where freed and more power efficient resources will be substituted for less performing ones. The objective is to reduce opportunistically the chromatic number $\chi(G)$ further. This corresponds to a partial graph recoloring problem [113] where some vertices are recolored to maximize $\overline{PPW}$. Recoloring consists in our case of migrating a VM from a server to another more power efficient one (see Figure 5.8 and Figure 5.9).

Since the energy efficient VM reservation problem in cloud data centers is known to be NP-hard, an exact (in our case an ILP-based algorithm) will find the optimal solutions in acceptable convergence times only for small graphs or problem sizes. In order to scale and find solutions in reasonable convergence times, we resort to the EEGP heuristic, that uses the notations and variables listed in Table 5.1. The exact algorithm is useful to check if the performance of the EEGP algorithm is close to optimal in terms of number of used colors and energy efficiency and to assess the performance improvement in convergence time.

5.3 Energy Efficient Initial Advanced Resource Reservation

5.3.1 Exact energy efficient graph precoloring Algorithm

The proposed exact energy efficient graph precoloring algorithm is an extended graph coloring approach with valid conditions expressed in the form of constraints or inequalities. The problem is cast into an ILP whose objective function minimizes the number of used resources (VRUs or colors) and minimizes simultaneously the energy consumption of the data centers (or maximizes the energy efficiency by maximizing $\overline{PPW}$).

We define as key decision variable z_c for each color c that is set to 1 if color c is reserved to a RRU, 0 if it is not reserved. In addition, we define the bivalent

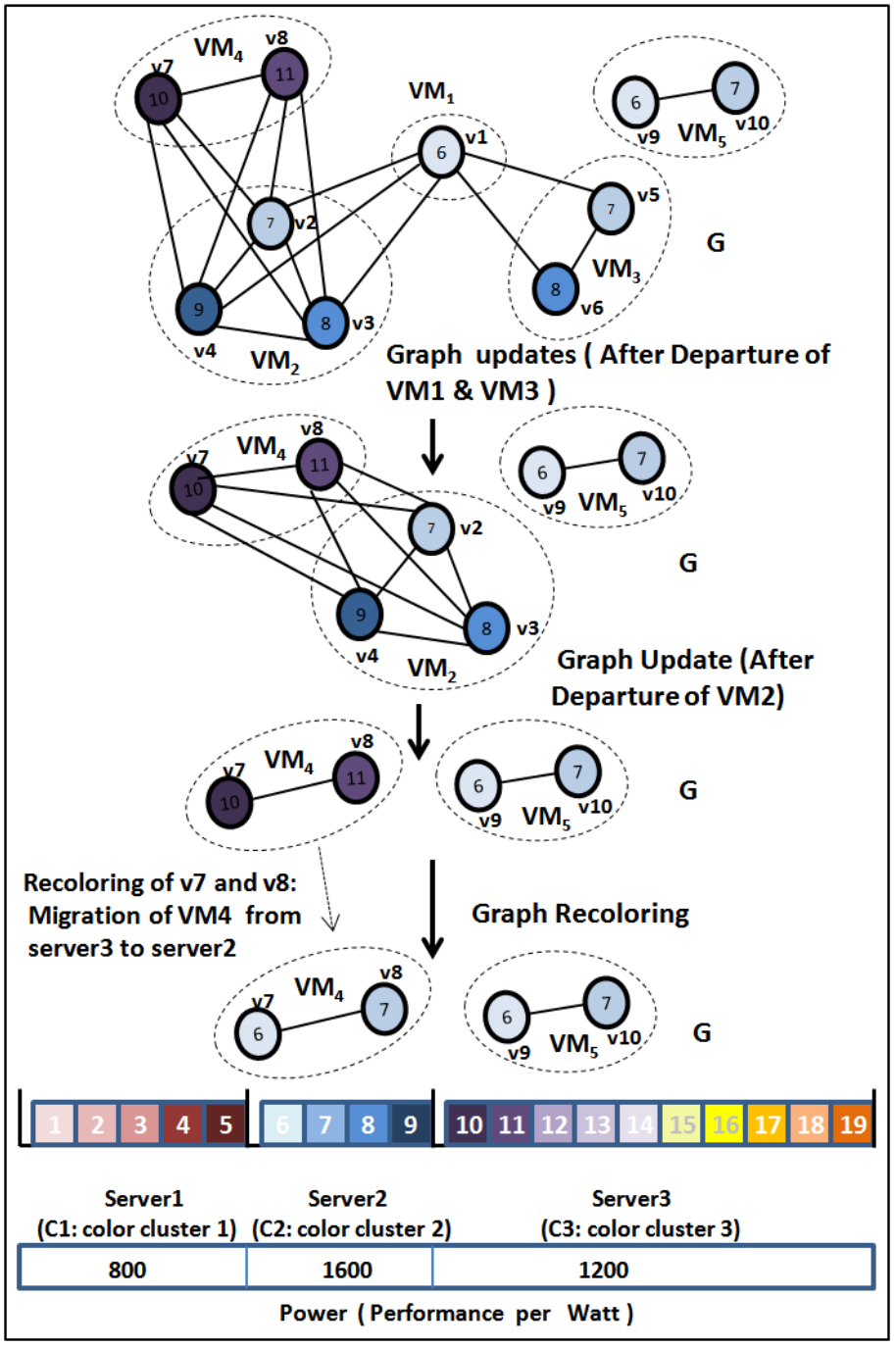

FIGURE 5.8: Energy Efficient Graph Recoloring

Table 5.1: Notations

Symbol	Meaning
VRU	Virtual resource unit : a **color**.
RRU	Abstract representation of VRU at request level : a graph **vertex**.
G	Global graph of requests.
V	Vertex set of G. It represents the RRUs belonging to all requested VMs.
E	The set of all edges in the graph G.
E'	The set of intra VM edges
$\chi(G)$	The chromatic number of a graph (the smallest number of colors needed to color the vertices of G).
G_k	Subgraph associated to request R_k.
V_k	Vertex set of the graph G_k.
E_k	Edge set of the graph G_k.
R_k	A request having a single id k.
n_k	Number of virtual machines of request R_k.
VM_i	A virtual machine which logically divided into r_i RRUs.
r_i	Number of VRUs requested by VM_i.
$[a_i, b_i]$	The time during it a VM_i is reserved.
$c_{j,id}$	A virtual resource unit VRU (or a color), where j is the server to which it belongs and where id is its associated id.
C_j	Cluster of colors containing colors that belong to the same server j.
w_j	The performance per watt (PPW) of the server j.
$P_{j,idle}$	Power consumption of server j when it is idle.
z_c	A binary variable. $z_c = 1$ if color c is used and 0 otherwise.
x_{uc}	A binary variable. $x_{uc} = 1$ if RRU u is reserved to color c and 0 otherwise.
y_j	A binary variable. $y_j = 1$ if at least one color belonging to C_j is used and 0 otherwise.
n	Total number of nodes in the graph G.
m	Number of virtualized servers of the data center.

variable x_{uc} to indicate that *RRU* u has been reserved to color c and set x_{uc} to 1; $x_{uc} = 0$ otherwise.

The objective function to reserve all the demands (or RRUs) to a minimum number of colors while maximizing energy efficiency can be expressed using:

$$\min \sum_{c \in C} z_c - \sum_{j=1}^{m} \sum_{u=1}^{n} \sum_{c \in C_j} w_j x_{uc} \tag{5.2}$$

This optimization is subject to a number of linear constraints and obvious facts such as a color can only be reserved to one RRU or such as RRUs of the same VM are reserved to colors belonging to the same color cluster. These conditions are formally expressed through valid inequalities and constraints that have to be respected when maximizing overall energy efficiency :

1. Each requested resource unit RRU is associated to one and only one VRU. The proposed equality (5.3) ensures that each vertex of the graph is colored with a unique color :

$$\sum_{c \in C} x_{uc} = 1, \forall u \in V \tag{5.3}$$

2. In graph coloring, any two nodes connected by an edge must have different colors. The valid inequality (5.4) ensures that two linked RRUs are reserved to two different VRUs (or colors) :

$$x_{uc} + x_{vc} \leq 1, \forall (u, v) \in E, c \in C \tag{5.4}$$

3. Another valid inequality ensures that z_c is equal to 1 if the color c is assigned to a RRU u :

$$x_{uc} \leq z_c, \forall u \in V, c \in C \tag{5.5}$$

4. The RRUs of the same virtual machine have to be associated to colors of the same color cluster (server):

$$x_{uc} \leq \sum_{c, c' \in C_j, c \neq c'} x_{vc'}, \forall (u, v) \in E', j = 1, \ldots, m \tag{5.6}$$

The exact and extended graph precoloring model can be summarized by lumping the objective function with all the constraints and conditions into the following set of equations:

$$\min \sum_{c \in C} z_c - \sum_{j=1}^{m} \sum_{u=1}^{n} \sum_{c \in C_j} w_j x_{uc} \tag{5.7}$$

Subject To:

$$\sum_{c \in C} x_{uc} = 1, \forall u \in V \tag{5.8}$$

$$x_{uc} + x_{vc} \leq 1, \forall (u, v) \in E, c \in C \tag{5.9}$$

$$x_{uc} \leq z_c, \forall u \in V, c \in C \tag{5.10}$$

$$x_{uc} \leq \sum_{c, c' \in C_j, c \neq c'} x_{vc'}, \forall (u, v) \in E', j = 1, \ldots, m \tag{5.11}$$

$$z_c = \begin{cases} 1, & \text{if color } c \text{ is used;} \\ 0, & \text{otherwise.} \end{cases} \tag{5.12}$$

$$x_{uc} = \begin{cases} 1, & \text{if a RRU u is colored with color } c; \\ 0, & \text{otherwise.} \end{cases} \tag{5.13}$$

The notations used by the proposed graph coloring model for the energy efficient resource reservation are listed in Table 5.1 for easy reference.

5.3.2 Energy efficient graph precoloring heuristic (EEGP)

Using the same underlying model illustrated in Figures 5.6 and 5.7, to derive the exact ILP-based algorithm (through a graph pre-coloring extension), we propose

the EEGP heuristic as an alternative that converges much faster to near optimal solutions (found by the exact algorithm).

The proposed EEGP algorithm assigns gradually colors to not yet colored vertices (RRUs). For each set of vertices (RRUs) belonging to the same VM, the algorithm uses the steps specified below to achieve coloring which is equivalent to assigning a VRU to each RRU in the set. The EEGP algorithm uses the following steps to find a solution:

1. Find the color cluster C_j with the highest PPW and with free VRUs. The first step of the EEGP algorithm is handled by the function **Find-Color-Cluster**(C,VM_i) described further in this work.

2. Determine the neighboring RRUs (or graph vertices) directly connected to VM_i RRUs. This step in the EEGP algorithm is handled by the function **List-of-Connected-Nodes**(VM_i) that constructs the list L_{VM_i} of RRUs connected to VM_i. In Figure 5.7, $L_{VM_4} = \{v2, v3, v4\}$.

3. Construct the list of colors $\in C_j$ that are not assigned to VM_i neighboring RRUs. This step uses the function **List-of-Unused-colors**(C_j,L_{VM_i}) to construct the list col_{j,VM_i}.

4. Finally, the algorithm can assign to each RRU $\in VM_i$ a different color from the list col_{j,VM_i}.

Algorithm 1 EEGP Algorithm

Input: Graph G and a set of color clusters C
Output: Coloring of G (Not colored nodes)

```
1: for all (Not colored) VM_i ∈ V do
2:    C_j = Find-Color-Cluster(C,VM_i)
3:    L_VMi = List-of-Connected-Nodes(VM_i)
4:    col_j,VMi = List-of-Unused-colors(C_j,L_VMi)
5:    for all RRU ∈ VM_i do
6:       color(RRU, col_j,VMi)
7:    end for
8: end for
```

The function **Find-Color-Cluster**(C,VM_i) selects the color cluster $C_j \in C$ (or server) from which to reserve (partially or totally) VRUs for the VM_i RRUs (or vertices). The selected C_j, is the color cluster with the highest PPW that satisfies the following two conditions:

function: Find-Color-Cluster (C,VM_i) :
Input: A VM_i and a set of color clusters C
Output: A color cluster C_j to color VM_i RRUs

```
1: boolean found = false
2: Sort the list of used color clusters in decreasing order by their highest PPW.
3: for C_j in the list do
4:    free-colors(VM_i, C_j) = |C_j| - deg(VM_i, C_j)
5:    if (free-colors(VM_i, C_j) ≥ r_i) then
6:       found = true
7:       return C_j
8:    end if
9: end for
10: if (found == false) then
11:    Sort the list of unused color clusters in decreasing order by their highest
       PPW.
12:    for C_j in the list do
13:       free-colors(VM_i, C_j) = |C_j| - deg(VM_i, C_j)
14:       if free-colors(VM_i, C_j) ≥ r_i then
15:          return C_j
16:       end if
17:    end for
18: end if
```

1. C_j has enough free colors to assign to the VM_i RRUs. To verify this condition, we compute **free-colors**(VM_i, C_j) that gives the number of free colors of C_j needed to color the VM_i RRUs. Let **deg**(VM_i, C_j) be the number of colors associated to the neighbours of the VM_i RRUs within server C_j. The **free-colors**(VM_i, C_j) is obtained by subtracting **deg**(VM_i, C_j) from the total number of colors of C_j. In the example shown in Figure 5.7, **deg**(VM_4, C_2) is equal to 3, so, **free-colors**(VM_4, C_2) is equal to 1 $(4-3)$. Hence, C_2 does not have enough free resources to assign to the two VM_4 RRUs (namely, $v7$ and $v8$). This compels the algorithm to pursue the search by checking the next best PPW server (or color cluster), that is check C_3. This server has enough free colors because **deg**(VM_4, C_3) is equal to 0 (no neighbours are using the resources) and **free-colors**(VM_4, C_3) is equal to 10 $(10-0)$. The RRUs, $v7$ and $v8$ of VM_4, are assigned colors 10 and 11 from C_3.

2. Using C_j ensures that the minimum number of colors (VRUs) are used to color the graph G. For example, if C_j and C_k have the same power weight

(or PPW) and C_j is used but C_k is not, C_j is chosen because some of its colors are already used in the graph.

To color a subgraph G_k, the worst complexity of the algorithm is $|V_k| * m$, where V_k is the number of nodes of G_k and m is the number of hosts or color clusters. The cost of updating the global graph G by adding or deleting subgraphs depends on the data structure used to represent this graph. We adopt the most commonly used data structure for representing graphs which is the adjacency list representation implemented with linked lists of adjacent nodes because of its simplicity and dynamic aspects. This structure stores the adjacency list of each node as a linked list in a space of $O(|V| + |E|)$, supports optimal and dynamic insertions or deletions of nodes and fast scanning of edges. Inserting or deleting edges and nodes takes $O(1)$ when using adjacency list. To add subgraph G_k to the global graph G, nodes V_k and intra VM and inter VM edges E_k should be inserted. The links that indicate overlapping reservation time intervals (inter VM edges) are determined after consulting the head nodes list which is sorted according to reservation time. Hence, the worst complexity of adding a subgraph G_k to G is $O(|V| + |V_k| + |E_k|)$.

5.4 Energy Efficient Advanced Dynamic Resource Reservation

5.4.1 Energy Efficient Graph Recoloring Heuristic (EEGR)

The partial re-coloring heuristic algorithm, EEGR, is triggered at service departures to find more energy efficient solutions by re-coloring some of the graph G vertices. This corresponds to the migration of some of the VMs to another more power efficient server as described:

1. After the departure of a VM_i, EEGR builds a list $Recol_{VM_i}$ of candidates for re-coloring using the function **VMsToRecolor**(VM_i). This returns the list of VMs connected to VM_i that are associated to color clusters (or servers) whose PPW is lower than that associated to VM_i. Since VM_i departs and frees resources from a given server, these resources become candidates for

hosting VMs following migrations that will improve energy efficiency. Figure 5.8 illustrates these actions. When VM_2 (associated to C_2) departs (actually the entire Request R_1 departs), VM_4 that was connected to VM_2 and that is associated to the less energy efficient color cluster C_3 becomes candidate for migration that is achieved through the action: $Recol_{VM_2} = VM_4$ or colors/VRUs 6 and 7 of C_2 are assigned to VM_4 nodes $v7$ and $v8$ respectively.

2. For each VM_j belonging to list $Recol_{VM_i}$, color VM_j using **EEGP**.

Algorithm 2 EEGR Algorithm

```
1: if (End of VM_i associated to C_k) then
2:    Recol_{VM_i} ← VMsToRecolor (VM_i)
3:    for all VM_j ∈ (Recol_{VM_i}) do
4:       EEGP(VM_j)
5:    end for
6: end if
```

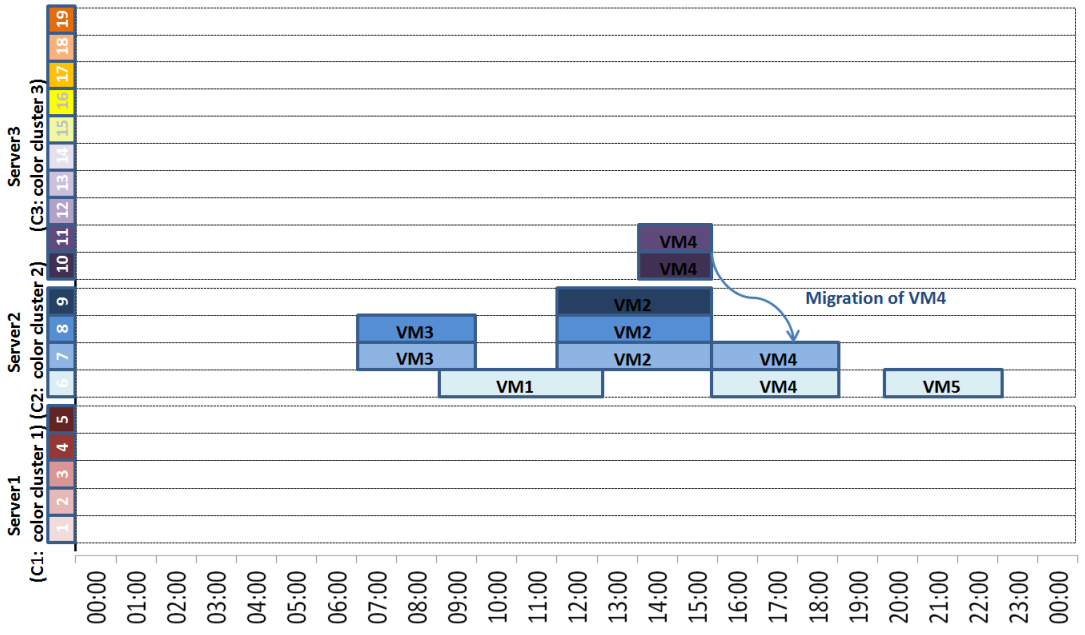

FIGURE 5.9: Reservation over time

5.4.2 Migration-Aware Energy Efficient Graph Recoloring Heuristic (MA-EEGR)

After service departures, we reassign some VMs on different servers for better optimization results. Note that at this stage, we can use the graph recoloring algorithm EEGR introduced in the last section. However, even if EEGR algorithm aims to dynamically reallocate (re-reserve) resources to maximize energy efficiency, it is still based on blind migration and did not take into account the cost of VM migrations. Therefore, we designed a more sophisticated migration aware recoloring algorithm MA-EEGR that dynamically re-reserve resources via VM live migration, while considering the induced energy cost of migrations. A VM migration correspond to an action of vertices recoloring in graph G. Migration decisions are energy aware since a server is not emptied (or freed) if the energy gained by this mechanism, $E_{j,gain}$, is lower than the threshold fixed by the administrator, $E_{threshold}$ (see Eq. 5.14). We define $E_{j,gain}$ as the energy saved after emptying a server j. This gain is calculated by retrieving the energy consumed by VM migrations from server j to a server k ($E_{mig,C_j/C_k}$) and the energy consumed to switch on the server j from the idle energy consumption of server j ($E_{j,idle}$) during the period of its inactivity ($t_{j,inactive}$). For simplicity reasons, we assume powering on or off a server is negligible and $E_{j,OFF/ON}$ is equal to zero. The threshold $E_{threshold}$ is set to the lowest possible value which is zero. The energy consumption of VM migrations $E_{mig,C_j/C_k}$ is estimated using a lightweight mathematical model proposed in [61] (see section 2.3.3.2 for more details).

$$\begin{cases} E_{j,gain} \geq E_{threshold} \\ E_{j,gain} = E_{j,idle} - (E_{mig,C_j/C_k} + E_{j,OFF/ON}) \\ E_{j,idle} = P_{j,idle} * t_{j,inactive} \end{cases} \tag{5.14}$$

Algorithm 3 describes the proposed MA-EEGR algorithm. Once a virtual machine VM_i (associated to C_k) ends, we build a list of color clusters candidate to be freed (line 2). These selected color clusters have colors associated to VMs connected to VM_i and whose PPW is lower than the color cluster associated to VM_i. For example, in Figure 5.8, VM_2 associated to C_2 departs. Color cluster C_3 is associated to VM_4 that was connected to VM_2. As C_3 is less energy efficient than C_2, C_3 is added to the list of candidate color cluster to be freed ($clusterlist = C_3$).

For each color cluster C_j in the list (*clusterlist*), we check first if color cluster C_k have enough free colors to recolor vertices assigned to colors from C_j and connected to VM_i (if **Empty**(C_j,C_k) = true). Then, we check if migration decision will provide power gain (line 5). Once these two conditions are satisfied, the next step is to build the list of VMs to recolor with colors from C_k. This list, *vmlist*, is returned by the function **list-vm-to-migrate**(VM_i, C_j) and contains VMs connected to VM_i and in the same time associated to C_j. The last step is to recolor each VM_j in *vmlist* following the same steps as our EEGP algorithm, but without applying the function **Find-Color-Cluster** (lines 8-11). The selected color cluster from which to reserve (partially or totally) colors (VRUs) for VM_j vertices (RRUs) is fixed to C_k. The worst complexity of this algorithm is $O(v*m)$, where v is the number of nodes to be recolored and m is the number of hosts or color clusters.

Algorithm 3 MA-EEGR Algorithm

```
1: if (End of VM_i associated to C_k) then
2:   clusterlist ← list-clusters-to-empty (VM_i)
3:   for C_j ∈ clusterlist do
4:     if (Empty(C_j,C_k)) then
5:       if (E_{j,gain} >= E_threshold) then
6:         vmlist ← list-vm-to-migrate (VM_i, C_j)
7:         for all VM_j ∈ vmlist do
8:           L_{VM_j} = List-of-Connected-Nodes(VM_j)
9:           col_{k,VM_j} = List-of-Unused-colors(C_k,L_{VM_j})
10:          for all RRU ∈ VM_j do
11:            color(RRU, col_{k,VM_j})
12:          end for
13:        end for
14:      end if
15:    end if
16:  end for
17: end if
```

5.5 Performance evaluation

5.5.1 Evaluation Settings

Heterogeneous data center. To simulate an heterogeneous data center, we refer to SPECpower_ssj2008 benchmark [104] which provides an evaluation of servers

based on the performance per watt metric. We generate an infrastructure composed of thousands of heterogeneous servers chosen randomly from the benchmarking results. For simplicity reasons, we assume that resource units are equivalent to ECUs [5] (e.g an Intel Xeon X5550 has 13 ECU and an Intel Xeon E5430 has about 11 ECU). Thus, information on performance per watt w_j and on the number of colors (VRU) $|C_j|$ of each server j are directly retrieved the SPECpower_ssj2008 benchmark.

Users' Requests. User requests arrive following a poisson process with rate of 10 requests per second. To remove the possibility of very short and very long living VMs, the lifetime of a VM or the interval during which a VM_i is reserved $(b_i - a_i)$ is uniformly distributed between 200s and 1800s. The number of VMs per request is uniformly distributed between 1 and 10. The number of RRUs per VM is uniformly distributed between 1 and 20 (e.g EC2 instance types, namely small, large, xlarge, high-cpu-medium and high-cpu-xlarge have respectively 1, 4, 8, 5 and 20 EC2 compute units (ECUs)).

5.5.2 Evaluation results

5.5.2.1 Energy Efficient Initial Advanced Resource Reservation

In the experiments below, we assess the performance of the EEGP heuristic in comparison with the proposed exact solution and also with the advance reservation (AR) algorithm from the Haizea scheduler. The AR algorithm is a greedy mapping algorithm based on the idea that free physical hosts are selected in priority. The assessment scenarios correspond to a data center with 1000 servers and thousands of virtual resource units VRUs (or colors) for the experiments leading to the results of Figures 5.12 and 5.13 and correspond to data centers with 100 servers and with 1175 VRUs for the rest of the evaluated scenarios. The efficiency of our heuristic EEGP algorithm is shown in Table 5.2. We evaluate the gap between the achieved objective functions values by the exact algorithm and the EEGP heuristic solution. Table 5.2 reports this gap in % between the EEGP heuristic and the exact solutions for small and medium global graph G sizes. This gap is computed using:

$$gap = \frac{(heuristic\_solution - optimal\_solution)}{optimal\_solution} \times 100$$

TABLE 5.2: Gap between EEGP and Exact solutions

Graph size	Objective function Z		Gap
	EEGP	Exact	
4	-9794.28	-9794.28	0
32	-78371.24	-78370.24	0.0012
69	-165171.61	-165170.61	0.0006
112	-267828.16	-267824.16	0.0014
117	-279474.33	-279470.33	0.0014
162	-380412.61	-380410.61	0.0005
208	-466824.37	-466817.37	0.0014
232	-507047.02	-507042.02	0.0009
262	-575017.54	-575012.54	0.0008
299	-654000.31	-653995.30	0.0007
335	-716852.47	-716844.47	0.0011
365	-769421.88	-769415.88	0.0007
399	-823698.82	-823688.82	0.0012
416	-854652.27	-854641.27	0.0012
437	-889098.53	-889089.53	0.0010
Average gap			0.0010

Table 5.2 indicates that the EEGP achieved objective function values are only 0.001 % from the optimal. We next examine the computation time of the proposed algorithms. It can be seen, in Figure 5.10, that the EEGP heuristic is computationally efficient compared to the exact algorithm. For example, the heuristic algorithm finds solutions within 0.0005 % of optimal in about 10 ms while the exact algorithm requires 31 mn to find the optimal solution. In addition to global graph size G, convergence time depends also on the request subgraph sizes. As shown in Figure 5.10, convergence time for the exact algorithm is 13 min for $(|V| = 112, |V_k| = 43)$ and 4.4 s for $(|V| = 117, |V_k| = 5)$. The EEGP heuristic can achieve near optimal performance with significantly reduced computation time (see Figure 5.11) not exceeding 12 s for graph sizes of 3000 $(|V| = 3000)$. The AR algorithm is faster than EEGP for large instances since it does not seek near optimal or optimal solutions. It just needs to find the least loaded servers to host VMs. Looking jointly at Figures 5.11 and 5.12, the EEGP outperforms considerably the AR algorithm in terms of average PPW and uses much fewer servers (a few versus tens of servers or physical hosts). Figure 5.12 shows that the EEGP algorithm is four times better than the AR algorithm in energy efficiency.

In Figure 5.13, the EEGP heuristic performance is identical to that of the exact algorithm in terms of energy efficiency. Both algorithms provide the same average performance per watt when reserving VRUs and have the same behaviour since they select the most energy-efficient resources.

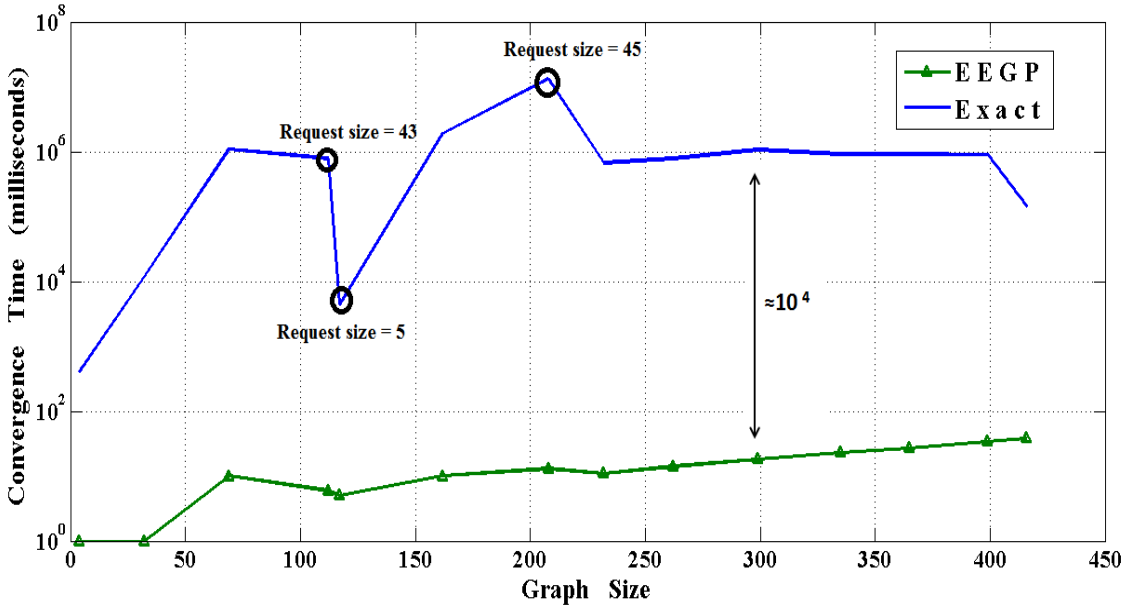

FIGURE 5.10: Convergence Time

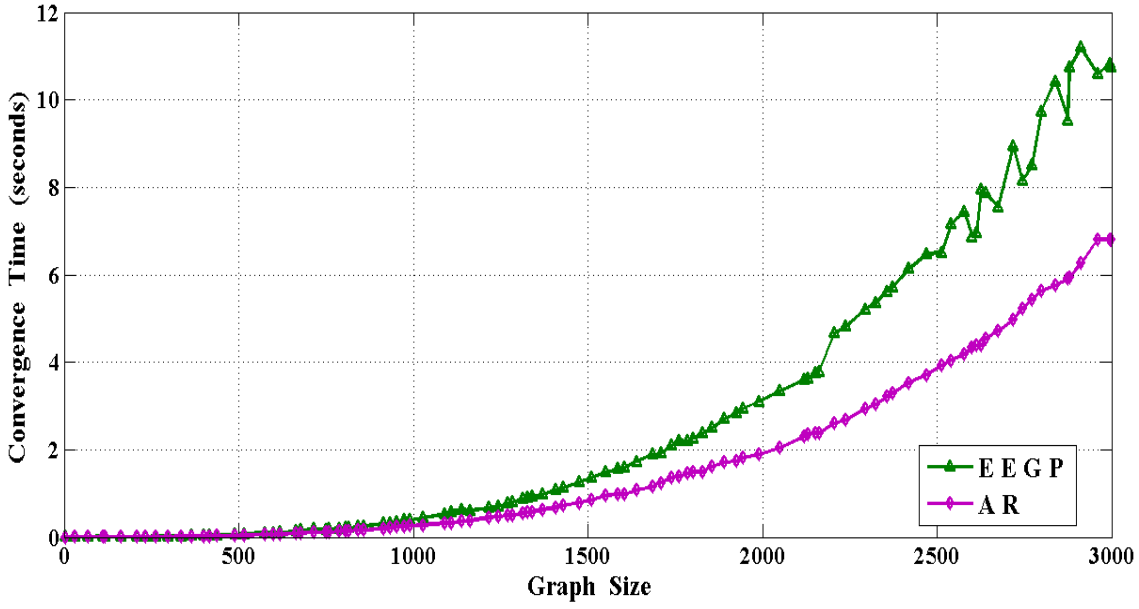

FIGURE 5.11: Convergence Time

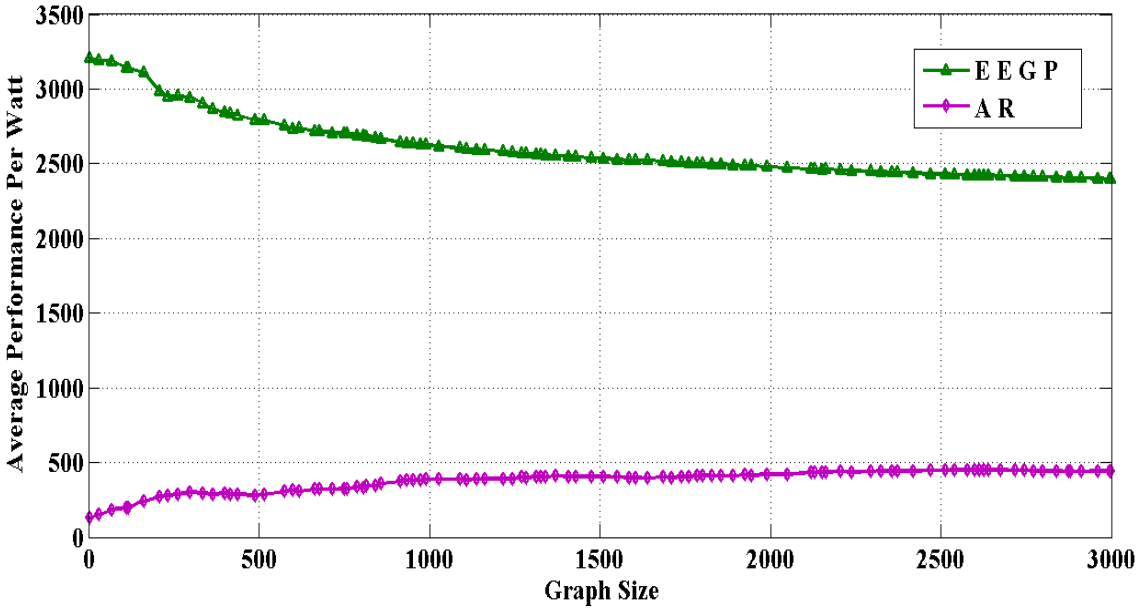

FIGURE 5.12: Average Performance Per Watt

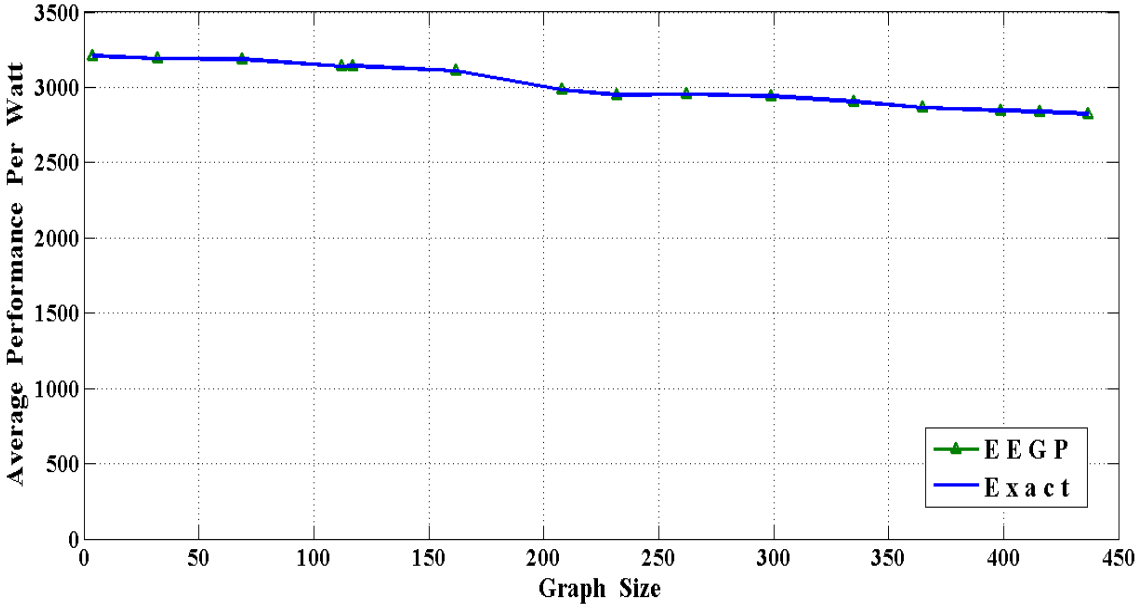

FIGURE 5.13: Average Performance Per Watt

The EEGP heuristic consumes a little bit more VRUs (or colors) than the exact solution as depicted in Figure 5.14. The chromatic number (number of used colors) is slightly higher when the heuristic algorithm is used. The EEGP heuristic uses nevertheless as many servers as the exact solution to achieve the resource reservation as depicted in Figure 5.15. The AR algorithm selecting the less loaded servers consumes more servers (a factor of 10). The effectiveness of EEGP heuristic is highlighted by the very small (0.0014%) relative gap with the exact solution. The EEGP heuristic overall performance is quite good in terms of energy, resource usage and is as remarkable in terms of convergence time that is found close to the simple and basic AR algorithm.

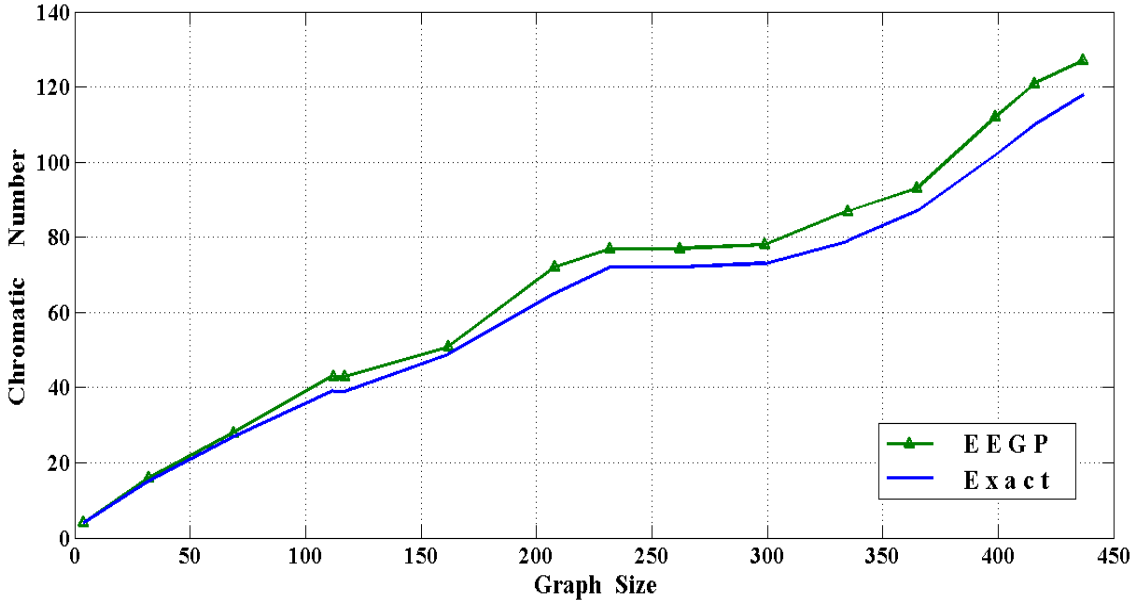

FIGURE 5.14: Chromatic Number

Table 5.3 depicts the convergence time of the EEGP algorithm running in a server with 24GB of RAM and a pair of 2.53GHz Intel Xeon E5540 quad-core processors. In this experiment, the size of the data center m was fixed to 10000 servers. The time required for EEGP to converge to a solution was evaluated for requests of size 30 RRUs per VM and for colored graph, G, sizes of 1000, 2000, 3000, 4000 and 5000. Results show that EEGP solves the problem for large graphs in reasonable time ; not exceeding a minute for the largest graph size of 5000.

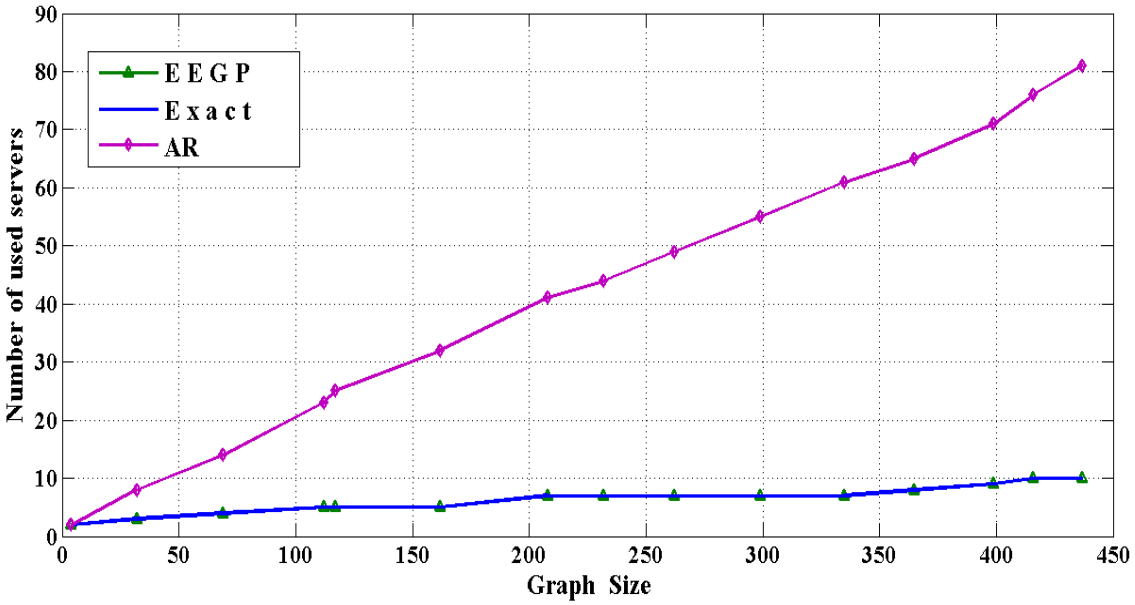

FIGURE 5.15: Number of used servers

TABLE 5.3: convergence Time of EEGP (m=10000)

Request Size \ Graph Size	1000	2000	3000	4000	5000
30	500(ms)	4(s)	12(s)	35(s)	1(min)

5.5.2.2 Energy Efficient Advanced Dynamic Resource Reservation

In the experiments below, we assess firstly the performance of the EEGP heuristic in comparison with the proposed EEGP+EEGR algorithm that combines the initial reservation process (EEGP) with the dynamic adaptation process(EEGR). Recall that EEGR algorithm aims at adapting dynamically the resources placement at service departures using the technique of resources recoloring. Secondly, we assess the impact of the migration cost on the recoloring algorithms. We compare the gain, in term of energy, obtained when comparing the Energy Efficient Graph Recoloring and the Migration Aware Energy Efficient Graph Recoloring algorithms.

The performance evaluation is conducted for low and high load conditions and for sustained arrivals of resource reservation requests as well as discontinuous requests to highlight when the use of EEGR is relevant.

(1) Algorithms' behaviour when demand is high and continuous:

Figure 5.16 shows the average performance per watt of the data center when applying AR, EEGP and EEGP+EEGR algorithms. In this experiment, the size of data center m is set to 1000. Request arrivals follow a poisson process with rate of 10 requests per second. The average performance per Watt of the data center improves significantly when the EEGP and EEGP+EEGR algorithms are used. As expected, the EEGP is more than five times better than AR in terms of average energy efficiency. For this high load conditions, EEGP and EEGR, achieve slightly better results (on average) than EEGP alone (two per cent). EEGR finds no opportunities to improve the overall energy efficiency through migration since resources are always used. No resources can be freed for long enough to be exploited for improvements.

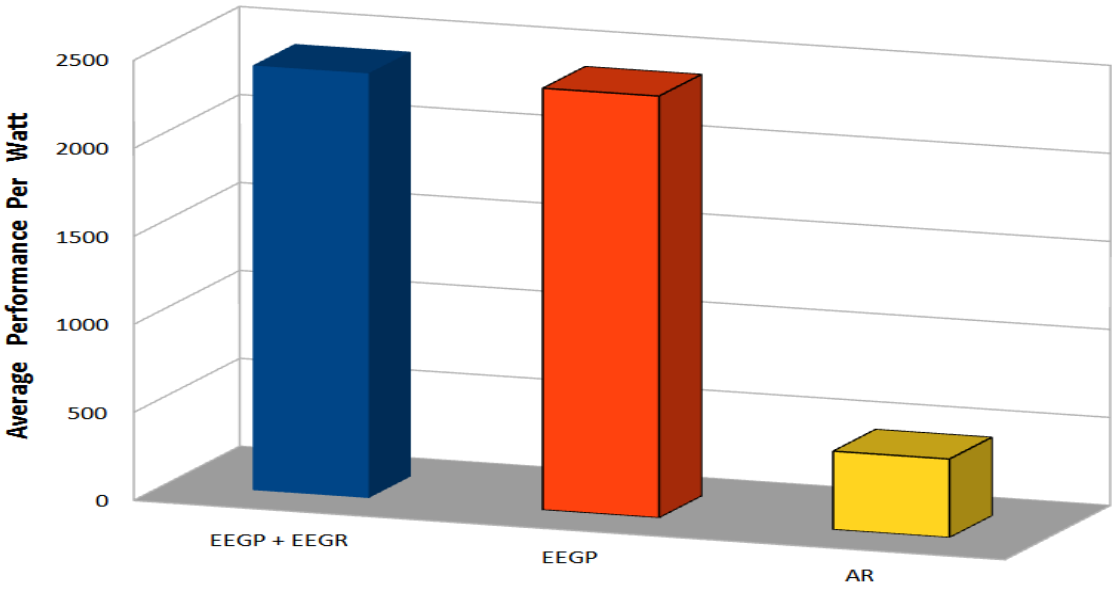

FIGURE 5.16: Average Performance Per Watt(high load conditions)

(2) Algorithms' behaviour when request arrivals is low:

Figure 5.17 shows the average performance per watt of the data center at low load for the three algorithms. In this simulation, the size of data center m is set to 1000.

The load is generated in a an Interrupted Poisson Process arrival pattern. EEGP is more than twice better than AR in terms of average performance per Watt. Results achieved by EEGP and EEGR both running together are roughly three times better than AR and nearly twenty-five percent better than EEGP. The quiet periods, in the requests arrivals, allow the EEGR algorithm to improve energy efficiency through migration of active VMs into less power consuming servers. The use of EEGR is only useful when there are enough and long enough quiet periods in user demand.

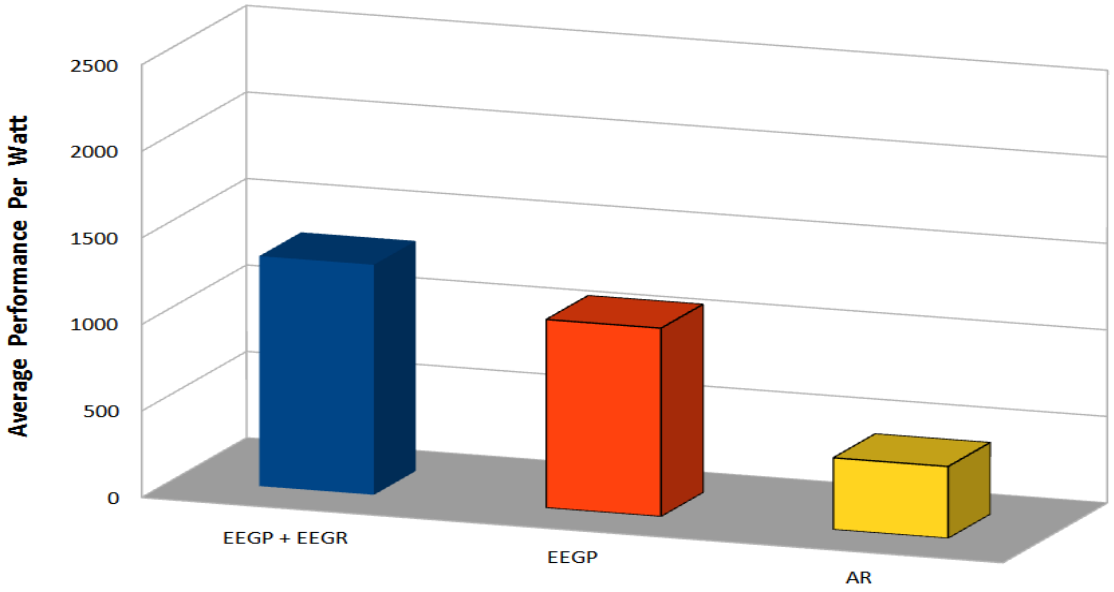

FIGURE 5.17: Average Performance Per Watt (low load conditions)

(3) Resource utilization:

Figure 5.18, depicts the evolution of the number of used servers by the proposed heuristics and AR to assess their performance as a function of load. To reveal the relative performance of the algorithms, we use a specific scenario where we increase gradually the system load (through additional arrivals) until event 50 (see abcissa) and then uniformly decrease this load at constant rate from event 50 to 300 through service departures. The results show that for high load both EEGP and EEGP combined with EEGR use the same number of servers (curves up to event 50 when load reaches its maximum in the simulated scenario) while AR uses many more. Our heuristic algorithms use only 135 servers (at event 50 or

peak load) while AR requires 248. After event 50, in the 50 to 300 range, where the load decreases continuously, EEGP combined with EEGR uses the smallest number of servers because this heuristic consolidates resources via migration of VMs, at service departures, and thus can pack virtual resources in fewer physical servers or resources.

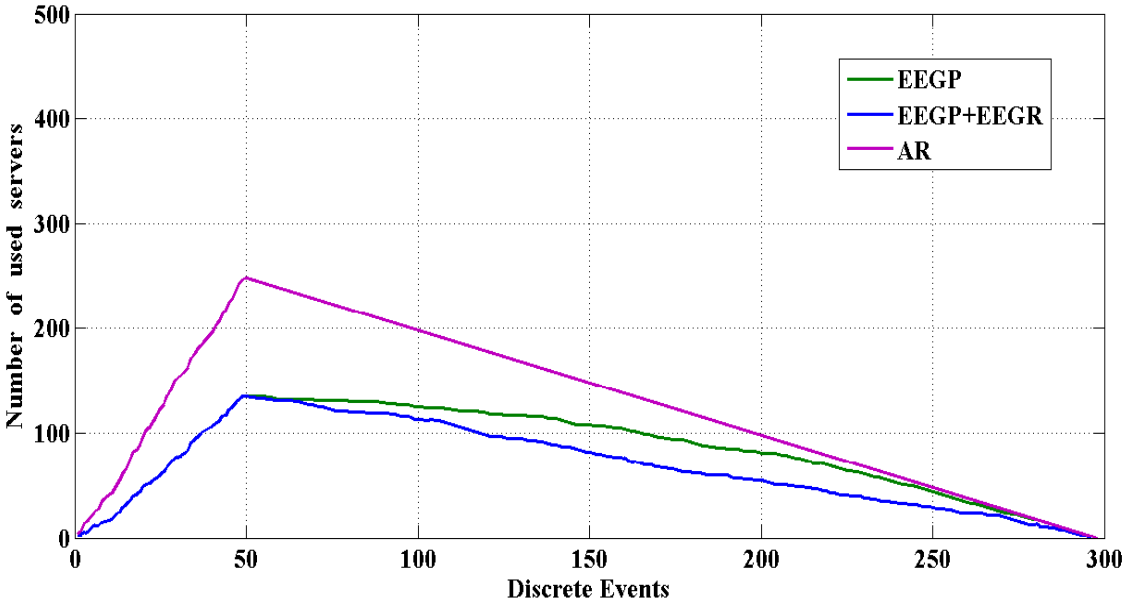

FIGURE 5.18: Number of used servers

(4) Migration cost impact:

In this subsection, we conduct a comparison between the Energy Efficient Graph Recoloring EEGR and the Migration Aware Energy Efficient Graph Recoloring MA-EEGR algorithms. We compare the performance of the proposed recoloring algorithms in terms of percentage of migrated VMs, the number of shutdown and the migration cost (energy). The size of data center m is set to 100. Request arrivals follow a poisson process with rate of 10 requests per second. Figure 5.19 shows that the blind recoloring algorithm EEGR migrates VMs **7 times** more than the migration aware recoloring algorithm. As shown in Figure 5.20, this excessive VM migration leads to a large number of servers shutdown when using the EEGR recoloring algorithm. For a colored graph, G, of size 5892, the recoloring algorithm EEGR shutdown 35 servers while the migration aware recoloring algorithm MA-EEGR shutdown 7 servers. Figure 5.21 shows that MA-EEGR algorithm reduces the energy introduced by the migration of VM in comparison with the EEGR

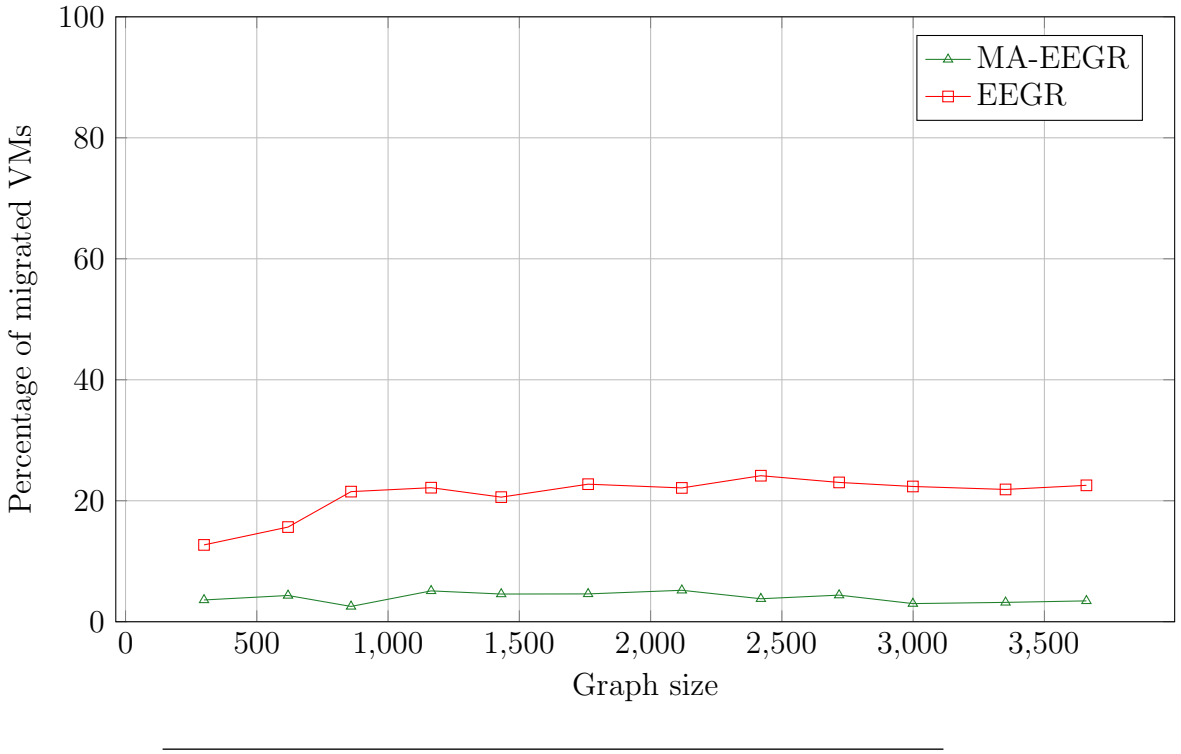

FIGURE 5.19: Migration percentage comparison between MA-EEGR and EEGR

recoloring algorithm. For the colored graph G size (5892), the EEGR algorithm wastes energy **8 times** more than the migration aware recoloring algorithm MA-EEGR. MA-EEGR is more efficient and mature compared to EEGR as it provides a trade-off between energy efficiency and stability.

An example of how algorithms MA-EEGR and EEGR perform is presented via Figure 5.22 and Figure 5.23. In case of applying the EEGR algorithm, after the departure of VM_2 hosted in server 2, VM_4 is migrated to server 2 and server 3 is emptied without checking if this decision will bring energy gain or not. If the energy cost $E_{mig,C_3/C_2}$ induced by migrating VM_4 to server 2 is expensive or exceeds $E_{3,idle}$, the energy gained by shutting down server 3 in two hours (or until the next VM is hosted, VM_6 in the example), the MA-EEGR algorithm do not take the decision of emptying server 3 and hence do not migrate its hosted VMs (see Figure 5.23). MA-EEGR is based on useful migration decisions, hence only efficient and gainful VM migrations are allowed.

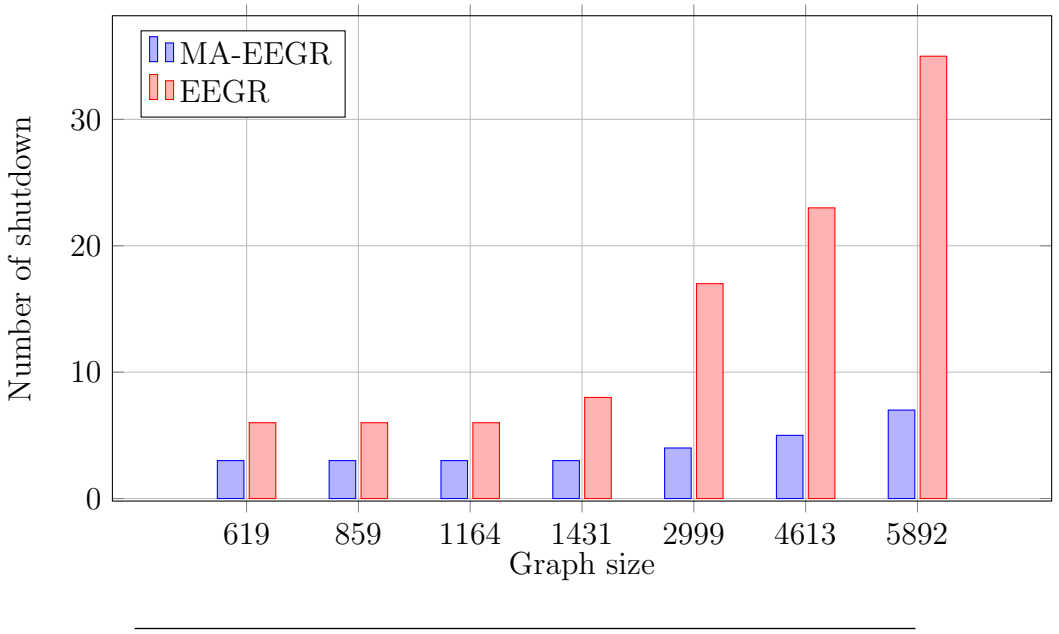

FIGURE 5.20: Comparison between MA-EEGR and EEGR in terms of number of server shutdown

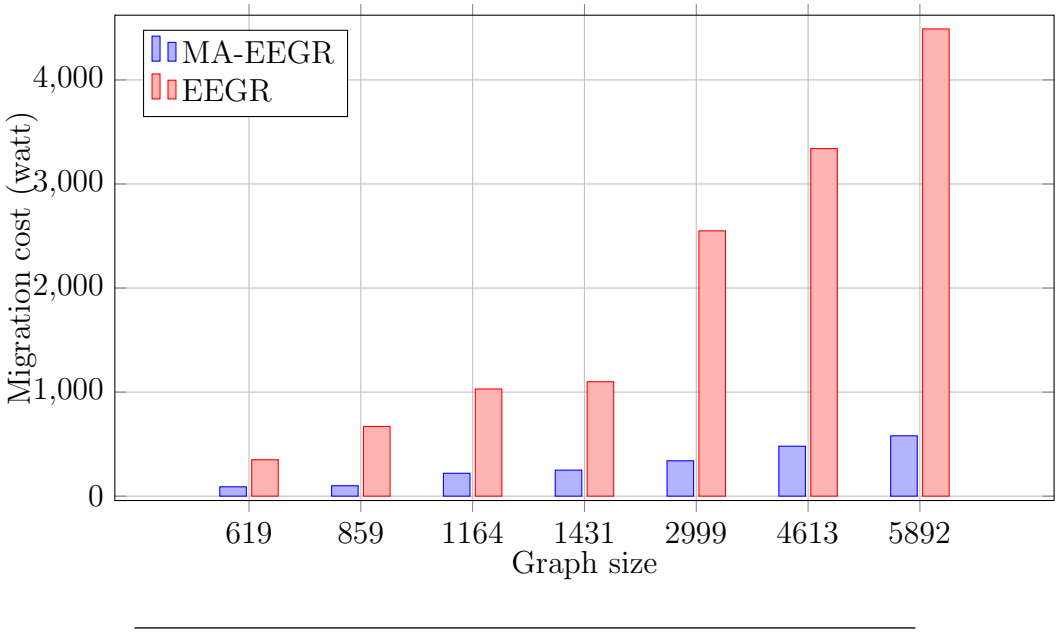

FIGURE 5.21: Comparison between MA-EEGR and EEGR in terms of migration cost

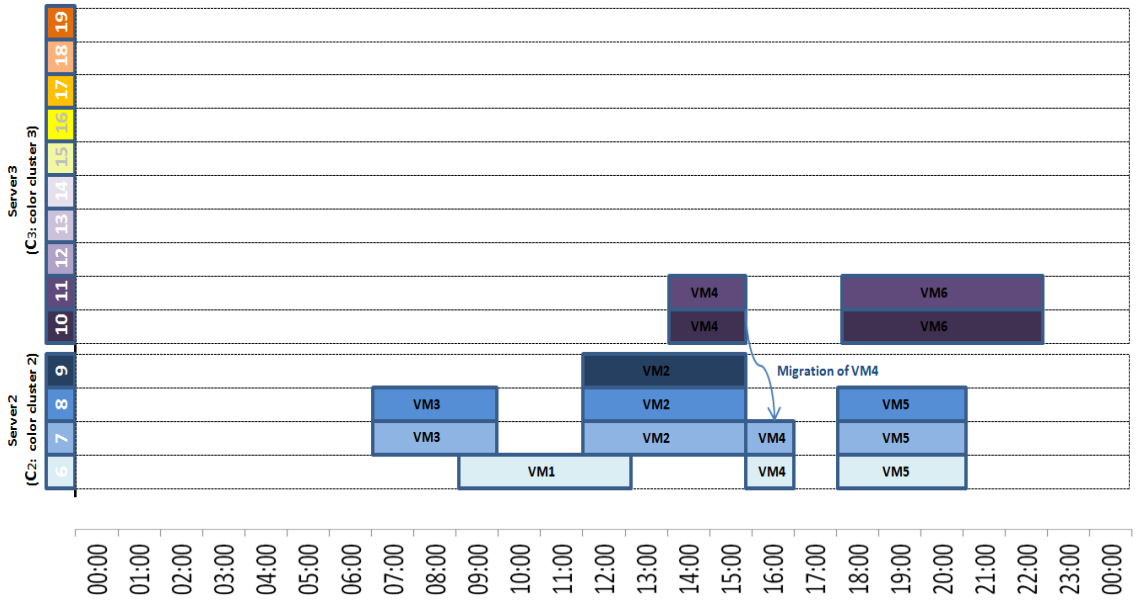

FIGURE 5.22: Recoloring using EEGR algorithm

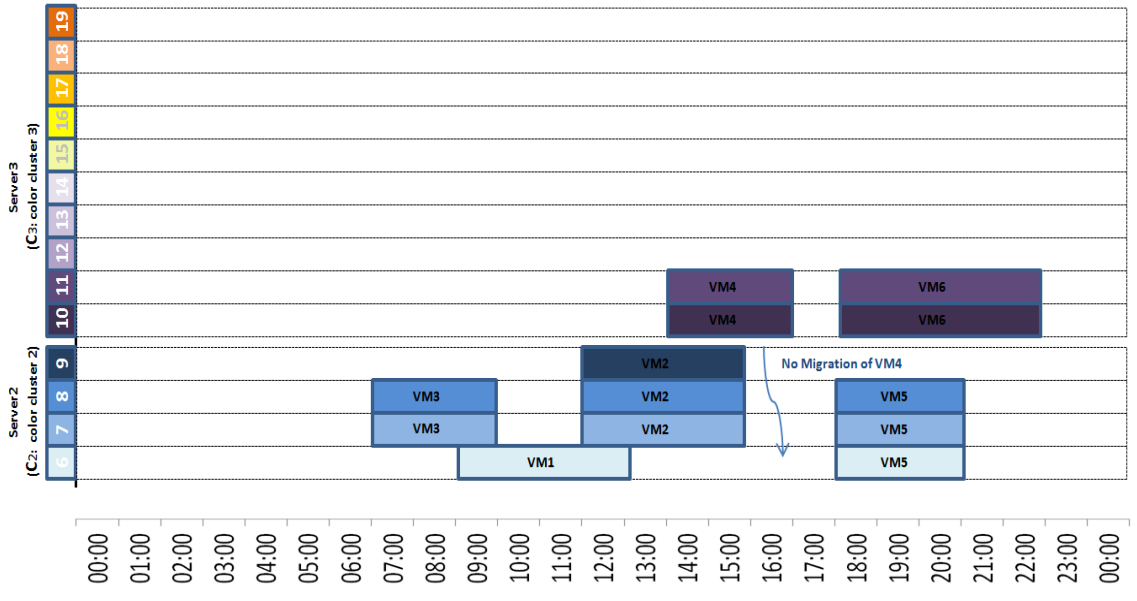

FIGURE 5.23: Recoloring using MA-EEGR alorithm

5.6 conclusions

This chapter presents a graph-coloring based approach for energy efficient resource allocation in hybrid IaaS-PaaS providers. This approach relies on a new graph coloring based model that supports both VM and container virtualization and provides on-demand and advanced reservation resource provisioning. We propose and develop an exact Pre-coloring algorithm for initial/static resource allocation while maximizing energy efficiency. A heuristic Pre-coloring algorithm for initial/static resource allocation was also proposed to scale with problem size. To adapt reservations over time and to improve further energy efficiency, we introduce two heuristic Re-coloring algorithms for dynamic resource reallocation. Evaluation and comparison of the exact and heuristic solutions in terms of energy efficiency, resource usage and convergence time are conducted to demonstrate the efficiency of our proposed algorithms. Our heuristic Pre-coloring algorithm for initial resource allocation is shown to perform very close to optimal, to scale well with problem size and to achieve fast convergence times. Both heuristic Re-coloring algorithms that dynamically adapt the resource allocations at service departures gain significantly energy efficiency and exhibit fast convergence.

Chapter 6

Conclusions and Future Research Directions

This thesis focuses on the design and development of models and algorithms for energy efficient resource allocation in Cloud data centers. In this final chapter we present conclusions about the work presented in this thesis and propose future directions for extending it. The first part summarizes the main contributions and draws conclusions. Then, we present potential directions for future investigations in the second part.

6.1 Conclusions and Discussion

Energy efficiency is becoming increasingly important for Cloud data centers. Their growing scale and their wide use have made a great issue of power consumption. The overall goal of this work is to design and develop models and algorithms for energy efficient resource allocation while considering different dimensions of the problem. These key dimensions are the resource provisioning plan, the dynamicity of the solution, the type of the virtualization and the Cloud service model.

Solving the problem of resource allocation in Cloud while maximizing energy efficiency and adopting the previously cited dimensions, is a very challenging issue. In this thesis, we address the problem with its multiple facets and levels to provide not only a specific solution, but also a generic and complete approach. The major contributions which have been made by this thesis are summarized as follows:

- Chapter 2 introduces the concepts Cloud computing and of virtualization that serves as its enabling technology. We further investigate the problem of energy efficiency in Cloud data centers by studying the major causes of energy waste, presenting the different power saving techniques and introducing energy measurement and modeling in Cloud environments.

- Chapter 3 presents a survey of the state of the art on energy efficient resource allocation in cloud environments. We describe in more details the problem of energy efficient resource allocation in Cloud data centers then we provide an overview on the state of the art of energy efficient Cloud resource allocation at different dimensions and levels. Our goal in this survey has been to get a deeper understanding of the problem, to position the thesis in relation to existing research and to identify the key challenges and issues.

- In chapter 4, we propose a bin packing based Approach for Energy Efficient Resource Allocation for Classical IaaS clouds. We formulate the problem of energy efficient resource allocation as a bin-packing model. This model is VM based and provides on-demand resource allocation. We propose an exact energy aware algorithm based on integer linear program (ILP) for initial resource allocation. To deal with dynamic resource consolidation, an exact ILP algorithm for dynamic VM reallocation was also proposed. It is based on VM migration and aims to optimize constantly the energy efficiency after service departures. A heuristic method based on best-fit algorithm was also adapted to the problem. Experimental results show benefits of combining the allocation and migration algorithms and demonstrate their ability to achieve significant energy savings while maintaining feasible runtimes when compared with the best fit heuristic.

- Chapter 5 presents a graph-coloring based approach for energy efficient resource allocation in hybrid IaaS-PaaS providers. This approach relies on a new graph coloring based model that supports both VM and container virtualization and provides on-demand and advanced reservation resource provisioning. We propose and develop an exact Pre-coloring algorithm for initial/static resource allocation while maximizing energy efficiency. A heuristic Pre-coloring algorithm for initial/static resource allocation was also proposed to scale with problem size. To adapt reservations over time and to improve further energy efficiency, we introduce two heuristic Re-coloring algorithms for dynamic resource reallocation. Evaluation and comparison of the exact

and heuristic solutions in terms of energy efficiency, resource usage and convergence time are conducted to demonstrate the efficiency of our proposed algorithms. Our heuristic Pre-coloring algorithm for initial resource allocation is shown to perform very close to optimal, to scale well with problem size and to achieve fast convergence times. Both heuristic Re-coloring algorithms that dynamically adapt the resource allocations at service departures gain significantly energy efficiency and exhibit fast convergence.

6.2 Future Research Directions

Some issues related to the energy efficient resource allocation problem in Cloud environments have not been addressed in this thesis, these limitations will be addressed as future work. The potential future directions of this research include the following:

- Admission control mechanisms that use different strategies are important to decide which user requests to accept. In fact, advanced reservation technique enables users to get guaranteed services in private Clouds where the capacity is limited since advanced reservation requests have strict starting and ending time and resources must be available at the specified time. If the system is extensively flooded with advance reservation requests, this will lead to starvation of on-demand requests. We aim to integrate an admission control mechanism with our solution to improve the optimality our of resource scheduling. This mechanism will be based on a negotiation process to propose alternative time slots if advance reservation requests are not accepted.

- Load prediction techniques play important role to predict the overall load in the system. As future work, we will enhance our solutions with prediction algorithms to further improve the stability and performance of our proposed resource allocation algorithms.

- Most research on resources scheduling in Cloud environments focus on computational resources. Scheduling network and storage with computational resources is not well investigated. In addition, the network connection between Cloud data centers is a important aspect to consider when scheduling

resources in geographically distributed Cloud environments. Future work must be done to extend our proposed work in order to deal with the above mentioned aspects.

- An important goal of this thesis is to integrate our proposed solutions for energy efficient resource allocation with OpenStack. We aim to provide the missing scheduling policies and to bring energy efficiency to this Cloud environment. We are actively progressing in the achievement of this goal and different manuals and documents were developed and published.

Thesis Publications

International Journal

- C. Ghribi and D. Zeghlache, "Graph Coloring for Energy Efficient Advance Resource Reservation in Green Clouds," Submitted to to the International Journal of Cloud Computing (IJCC), 2014.

International Conferences

- C. Ghribi, M. Hadji and D. Zeghlache, "Energy Efficient VM Scheduling for Cloud Data Centers: Exact Allocation and Migration Algorithms," Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on Cluster , pp. 671-678, doi:10.1109/CCGrid.2013.89.
- C. Ghribi and D. Zeghlache, "Exact and Heuristic Graph-Coloring for Energy Efficient Advance Cloud Resource Reservation," Cloud Computing (CLOUD), 2014, CLOUD '14 Proceedings of the 2014 IEEE International Conference on Cloud Computing, pp. 112-119, doi:10.1109/CLOUD.2014.25.

Posters

- C. Ghribi, M. Hadji and D. Zeghlache, "Dynamic Virtual Machine scheduling for Energy Efficient Cloud data centers," The Institut Mines-Télécom symposium, "Energy Tomorrow", 2013, http://www.mines-telecom.fr/wpcontent/uploads/2014/04/201305_Posters_efficacite_energetique.pdf.

Appendix A

VM instance creation in Openstack-nova IaaS providers

A.1 OpenStack Nova

OpenStack Nova is the OpenStack compute project. It is a compute controller that pools computing resources like CPU, memory, etc... Nova provides API's to control on-demand scheduling of compute instances like virtual machines on multiple virtualization technologies, bare metal, or container technologies. Nova uses images to launch instances or VMs. In this chapter, we will provide a description of the steps followed to create an instance with Nova.

A.2 Image creation

Create a simple credential file:

```
vi creds
#Paste the following:
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin_pass
export OS_AUTH_URL="http://192.168.100.11:5000/v2.0/"
```

Upload the cirros cloud image:

```
source creds
glance image-create --name "cirros-0.3.2-x86_64" --is-public true \
```

```
--container-format bare --disk-format qcow2 \
--location http://cdn.download.cirros-cloud.net/0.3.2/cirros-0.3.2-x86_64-disk.img
```

List Images:

```
glance image-list
```

A.3 Initial network creation

After creating the image, the next step is to create the virtual network infrastructure to which the instance will connect.

Create an external network:

```
source creds

#Create the external network:
neutron net-create ext-net --shared --router:external=True

#Create the subnet for the external network:
neutron subnet-create ext-net --name ext-subnet \
--allocation-pool start=192.168.100.101,end=192.168.100.200 \
--disable-dhcp --gateway 192.168.100.1 192.168.100.0/24
```

Create an internal (tenant) network:

```
source creds

#Create the internal network:
neutron net-create int-net

#Create the subnet for the internal network:
neutron subnet-create int-net --name int-subnet \
--dns-nameserver 8.8.8.8 --gateway 172.16.1.1 172.16.1.0/24
```

Create a router on the internal network and attach it to the external network:

```
source creds

#Create the router:
neutron router-create router1

#Attach the router to the internal subnet:
neutron router-interface-add router1 int-subnet

#Attach the router to the external network by setting it as the gateway:
neutron router-gateway-set router1 ext-net
```

Verify network connectivity:

```
#Ping the router gateway:
ping 192.168.100.101
```

A.4 Instance launching

Generate a key pair:

```
ssh-keygen
```

Add the public key:

```
source creds
nova keypair-add --pub-key ~/.ssh/id_rsa.pub key1
```

Verify the public key is added:

```
nova keypair-list
```

Add rules to the default security group to access your instance remotely:

```
# Permit ICMP (ping):
nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0

# Permit secure shell (SSH) access:
nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
```

Launch your instance:

```
NET_ID=$(neutron net-list | awk '/ int-net / { print $2 }')
nova boot --flavor m1.tiny --image cirros-0.3.2-x86_64 --nic net-id=$NET_ID \
--security-group default --key-name key1 instance1
```

Note: To choose the instance parameters these commands could be used:

```
nova flavor-list   : --flavor m1.tiny
nova image-list    : --image cirros-0.3.2-x86_64
neutron net-list   : --nic net-id=$NET_ID
nova secgroup-list : --security-group default
nova keypair-list  : --key-name key1
```

Check the status of your instance:

```
nova list
```

Create a floating IP address on the external network to enable the instance to access to the internet and also to make it reachable from external networks:

```
neutron floatingip-create ext-net
```

Associate the floating IP address with your instance:

```
nova floating-ip-associate instance1 192.168.100.102
```

Check the status of your floating IP address:

```
nova list
```

Verify network connectivity using ping and ssh:

```
ping 192.168.100.102
# ssh into your vm using its ip address:
ssh cirros@192.168.100.102
```

Here is a snapshot of the Horizon dashboard interface after instance launching:

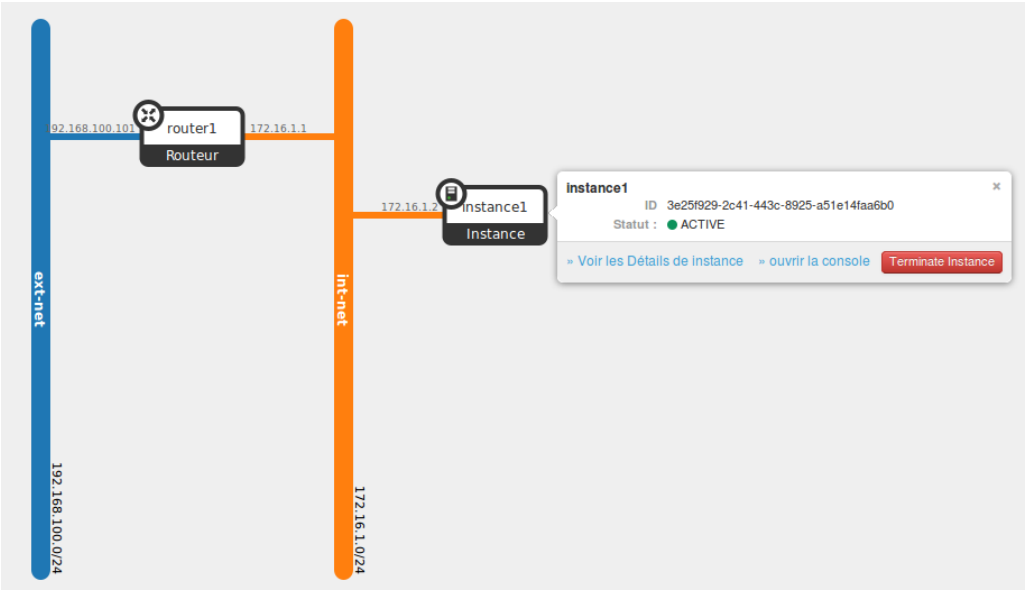

FIGURE A.1: VM instance creation in Openstack-nova IaaS providers

Appendix B

Hybrid IaaS-PaaS service with Docker and OpenStack Heat

B.1 OpenStack Heat

OpenStack Heat is an openstack service that handles the orchestration of complex deployments on top of OpenStack clouds. Orchestration basically manages the infrastructure but it supports also the software configuration management. Heat provides users the ability to define their applications in terms of simple templates. This component has also enabled OpenStack to provide a combined IaaS-PaaS service. Orchestrating Docker containers in OpenStack is via Heat provides orchestration of composite cloud applications and accelerates application delivery by making it easy to package them along with their dependencies. To deploy a stack, a Heat template that describes the infrastructure resources (instances, networks, database, images...) should be written and sent to Heat. It will talk to all the other OpenStack APIs to deploy the stack.

B.2 What is Docker?

Docker is an open source project to automatically deploy applications into containers. It commoditizes the well known LXC (Linux Container) solution that provides operating system level virtualization and allows to run multiple containers on the same server.

To make a simple analogy, a Docker is like an hypervisor. But unlike traditional Vms, docker containers are lightweight as they don't run OSes but share the host's operating system (see the Figure B.1).

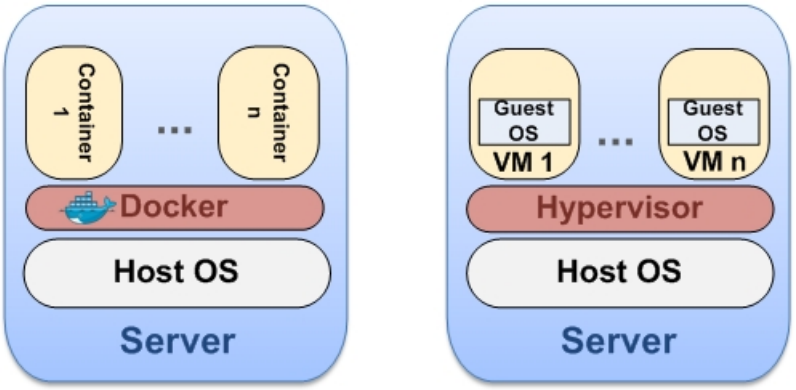

FIGURE B.1: Container based virtualization vs hypervisor based virtualization

A docker container is also portable, it hosts the application and its dependencies and it is able to be deployed or relocated on any Linux server. The Docker element that manages containers and deploys applications on them is called Docker Engine.

The second component of Docker is Docker Hub. It's the Docker's repository of application that allow users to share their applications with their team members and they can ship and run it anywhere

B.3 OpenStack and Docker

Openstack can be easily enhanced by docker plugins. Docker can be integrated into OpenStack Nova as a form of hypervisor (Containers used as VMs). But there is a better way to use Docker with OpenStack. It to orchestrate containers with OpenStack Heat.

To be able to create containers and deploy applications on the top of them, the docker plugin should be installed on Heat. To create a stack, the required resources should be identified, the template should be edited and deployed on Heat. For detailed information on Heat and template creation, see our Heat usage guide.

To test Docker with Heat, we recommand to deploy OpenStack using a flat networking model (see our guide). One issue we encountered when we considered a

multi-node architecture with isolated neutron networks is that instances were not able to signal to Heat. So, stack creation fails because it depends on connectivity from VMs to the Heat engine.

The Figure B.2 shows the communication between Heat, Nova, Docker and the instance when creating a stack. In this example we have deployed apache and mysql on two Docker containers. Stack deployment fails if the signal (3) can not reach Heat API.

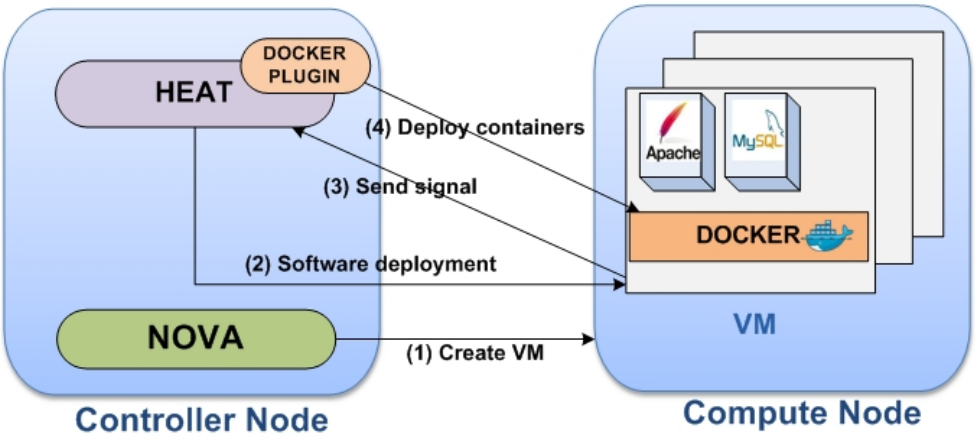

FIGURE B.2: Components interaction to create a stack

This limitation will be overcome in the next versions to allow users using isolated neutron networks to deploy and test Docker.

B.4 Deploy Docker containers with OpenStack Heat

In this section, we will show how to install the Docker plugin, how to write a template and how to deploy it with Heat.

B.4.1 Install the Docker Plugin

To get the Docker plugin, download the Heat folder available on GitHub:

```
download heat (the ZIP folder) from here
https://github.com/openstack/heat/tree/stable/icehouse
```

Unzip it:

```
unzip heat-stable-icehouse.zip
```

Remove the tests folder to avoid conflicts:

```
cd heat-stable-icehouse/contrib/
rm -rf docker/docker/tests
```

Create a new directory under /usr/lib/heat/:

```
mkdir /usr/lib/heat
mkdir /usr/lib/heat/docker-plugin
```

Copy the docker plugin under your new directory:

```
cp -r docker/* /usr/lib/heat/docker-plugin
```

Now, install the docker plugin:

```
cd /usr/lib/heat/docker-plugin
apt-get install python-pip
pip install -r requirements.txt
```

Edit /etc/heat/heat.conf file:

```
vi /etc/heat/heat.conf
(add)
plugin_dirs=/usr/lib/heat/docker-plugin/docker
```

Restart services:

```
service heat-api restart
service heat-api-cfn restart
service heat-engine restart
```

Check that the DockerInc::Docker::Container resource was successfully added and appears in your resource list:

```
heat resource-type-list | grep Docker
```

B.4.2 Create the Heat template

Before editing the template, let's discuss about the content and the resources we will define.

In this example, we want to dockerize and deploy a lamp application. So, we will create a docker container running apache with php and another one running mysql database.

We define an OS::Heat::SoftwareConfig resource that describes the configuration and an OS::Heat::SoftwareDeployment resource to deploy configs on OS::Nova::Server (the Virtual machine or the Docker server). We associate a floating IP to the Docker server to be able to connect to Internet (using OS::Nova::FloatingIP and OS::Nova::FloatingIPAssociation resources). Then, we create two docker containers of type DockerInc::Docker::Container on the Docker host.

Note: here we provide a simple template, many other interseting parameters (port_bindings, name, links...) can enhance the template and enable more sophisticated use of Docker. These parameters are not supported by the current Docker plugin. We will provide more complex templates with the next plugin version.

Create template in the docker-stack.yml file with the following content:

```
vi docker-stack.yml

heat_template_version: 2013-05-23

description: >
  Dockerize a multi-node application with OpenStack Heat.
  This template defines two docker containers running
  apache with php and mysql database.

parameters:
  key:
    type: string
    description: >
      Name of a KeyPair to enable SSH access to the instance. Note that the
      default user is ec2-user.
    default: key1

  flavor:
    type: string
    description: Instance type for the docker server.
    default: m1.medium

  image:
    type: string
    description: >
      Name or ID of the image to use for the Docker server.  This needs to be
      built with os-collect-config tools from a fedora base image.
    default: fedora-software-config

  public_net:
    type: string
```

```
    description: name of public network for which floating IP addresses will be allocated.
    default: nova

resources:
  configuration:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config: |
        #!/bin/bash -v
        setenforce 0
        yum -y install docker-io
        cp /usr/lib/systemd/system/docker.service /etc/systemd/system/
        sed -i -e '/ExecStart/ { s,fd://,tcp://0.0.0.0:2375, }'
        /etc/systemd/system/docker.service
        systemctl start docker.service
        docker -H :2375 pull marouen/mysql
        docker -H :2375 pull marouen/apache

  deployment:
    type: OS::Heat::SoftwareDeployment
    properties:
      config: {get_resource: configuration}
      server: {get_resource: docker_server}

  docker_server:
    type: OS::Nova::Server
    properties:
      key_name: {get_param: key}
      image: { get_param: image }
      flavor: { get_param: flavor}
      user_data_format: SOFTWARE_CONFIG

  server_floating_ip:
    type: OS::Nova::FloatingIP
    properties:
      pool: { get_param: public_net}

  associate_floating_ip:
    type: OS::Nova::FloatingIPAssociation
    properties:
      floating_ip: { get_resource: server_floating_ip}
      server_id: { get_resource: docker_server}

  mysql:
    type: DockerInc::Docker::Container
    depends_on: [deployment]
    properties:
      image: marouen/mysql
      port_specs:
        - 3306
      docker_endpoint:
        str_replace:
          template: http://host:2375
          params:
```

```
            host: {get_attr: [docker_server, networks, private, 0]}

  apache:
    type: DockerInc::Docker::Container
    depends_on: [mysql]
    properties:
      image: marouen/apache
      port_specs:
        - 80
      docker_endpoint:
        str_replace:
          template: http://host:2375
          params:
            host: {get_attr: [docker_server, networks, private, 0]}

outputs:
  url:
    description: Public address of apache
    value:
      str_replace:
        template: http://host
        params:
          host: {get_attr: [docker_server, networks, private, 0]}
```

B.4.3 Deploy the stack

Pre-deployment:

Create a simple credential file:

```
vi creds
#Paste the following:
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin_pass
export OS_AUTH_URL="http://controller:5000/v2.0/"
```

To create a fedora based image, we followed the steps bellow:

```
git clone https://git.openstack.org/openstack/diskimage-builder.git
git clone https://git.openstack.org/openstack/tripleo-image-elements.git
git clone https://git.openstack.org/openstack/heat-templates.git
export ELEMENTS_PATH=tripleo-image-elements/
elements:heat-templates/hot/software-config/elements
diskimage-builder/bin/disk-image-create vm \
fedora selinux-permissive \
heat-config \
os-collect-config \
os-refresh-config \
```

```
os-apply-config \
heat-config-cfn-init \
heat-config-puppet \
heat-config-salt \
heat-config-script \
-o fedora-software-config.qcow2
glance image-create --disk-format qcow2 --container-format bare
--name fedora-software-config < \
fedora-software-config.qcow2
```

If the key was not created yet, use these commands:

```
ssh-keygen
nova keypair-add --pub-key ~/.ssh/id_rsa.pub key1
```

Add rules to the default security group to enable the access to the docker server:

```
# Permit ICMP (ping):
nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0

# Permit secure shell (SSH) access:
nova secgroup-add-rule default tcp 22 22 0.0.0.0/0

# Permit 2375 port access (Docker endpoint):
nova secgroup-add-rule default tcp 2375 2375 0.0.0.0/0
```

If you need to create a new private network, use these commands:

```
source creds

#Create a private network:
nova network-create private --bridge br100 --multi-host T  --dns1 8.8.8.8  \
--gateway 172.16.0.1 --fixed-range-v4 172.16.0.0/24
```

Create a floating IP pool to connect instances to Internet:

```
nova-manage floating create --pool=nova --ip_range=192.168.100.100/28
```

Create the stack:

Create a stack from the template (file available here):

```
source creds
heat stack-create -f docker-stack.yml docker-stack
```

Verify that the stack was created:

```
heat stack-list
```

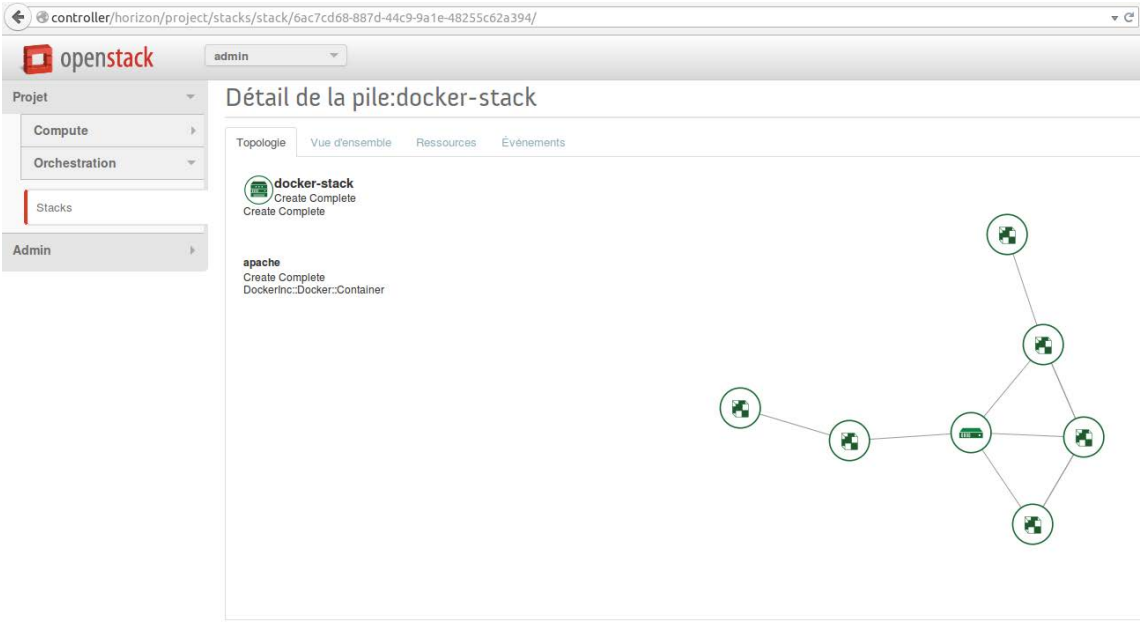

FIGURE B.3: Horizon dashboard interface after stack creation

Here is a snapshot of the Horizon dashboard interface after stack launching:

To check that the containers are created:

```
ssh ec2-user@192.168.100.97
sudo docker -H :2375 ps
```

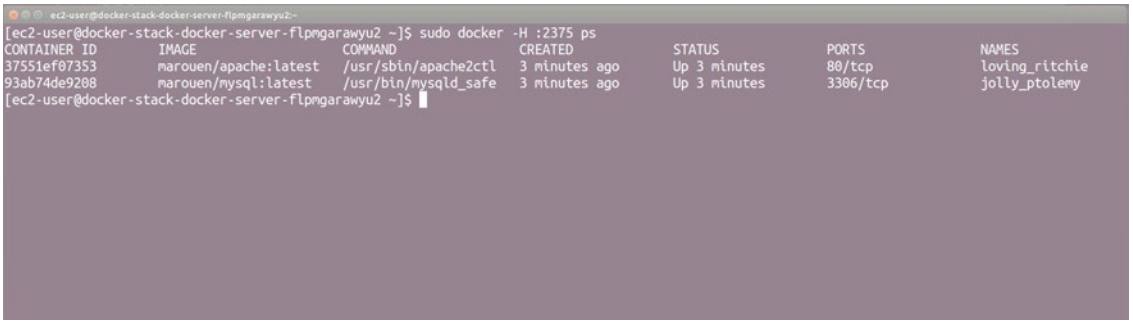

FIGURE B.4: Containers creation

Bibliography

[1] Power Management in the Cisco Unified Computing System: An Integrated Approach, Cisco Systems, April 2013, http://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/unified-computing/white_paper_c11-627731.pdf.

[2] World energy outlook 2009 fact sheet. http://www.iea.org/weo/docs/weo2009/factsheetsWEO2009.pdf.

[3] . Gartner report, financial times, 2007.

[4] James M. Kaplan, William Forrest, and Noah Kindle. Revolutionizing Data Center Energy Efficiency. Technical report, McKinsey & Company, July 2008.

[5] Amazon ec2. http://aws.amazon.com/ec2/.

[6] Microsoft azure services. www.azure.microsoft.com/.

[7] Google app engine. https://cloud.google.com/appengine/, .

[8] Google documents. https://docs.google.com/, .

[9] Google apps. http://www.google.com/intx/fr/enterprise/apps/business/, .

[10] Microsoft. www.microsoft.com/.

[11] Google. www.google.com/.

[12] kubernetes. https://github.com/GoogleCloudPlatform/kubernetes.

[13] Openstack. http://www.openstack.org/.

[14] Docker. https://www.docker.com/, .

[15] Is paas becoming just a feature of iaas?, 451 research group, january 2014. https://451research.com/report-short?entityId=79800.

[16] Kvm. http://www.linux-kvm.org/.

[17] Vmware. http://www.vmware.com/.

[18] Microsoft hyper-v. http://www.microsoft.com/Hyper-V. Accessed September 20, 2014.

[19] Xen. http://www.xenproject.org/.

[20] Virtual box. http://www.virtualbox.org/, .

[21] Shekhar Srikantaiah, Aman Kansal, and Feng Zhao. Energy aware consolidation for cloud computing. In *Proceedings of the 2008 conference on Power aware computing and systems*, HotPower'08, pages 10–10, Berkeley, CA, USA, 2008. USENIX Association.

[22] Franco Travostino, Paul Daspit, Leon Gommans, Chetan Jog, Cees de Laat, Joe Mambretti, Inder Monga, Bas van Oudenaarde, Satish Raghunath, and Phil Yonghui Wang. Seamless live migration of virtual machines over the man/wan. *Future Gener. Comput. Syst.*, 22(8):901–907, October 2006.

[23] Stephen Soltesz, Herbert Pötzl, Marc E. Fiuczynski, Andy Bavier, and Larry Peterson. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. In *In: Proceedings of EuroSys 2007*, pages 275–288, 2007.

[24] Miguel G. Xavier, Marcelo V. Neves, Fabio D. Rossi, Tiago C. Ferreto, Timoteo Lange, and Cesar A. F. De Rose. Performance evaluation of container-based virtualization for high performance computing environments. In *Proceedings of the 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 233–240, Washington, DC, USA, 2013. IEEE Computer Society.

[25] Linux containers. http://www.linuxcontainers.org/.

[26] Solaris containers. http://www.oracle.com/technetwork/server-storage/solaris/containers-169727.html.

[27] Virtuozzo containers. http://sp.parallels.com/fr/products/pvc/, .

[28] Openvz. www.openvz.org/, .

[29] *Virtualization and Containerization of Application Infrastructure: A Comparison*, volume 21, 2014. University of Twente.

[30] Proxmox. http://www.proxmox.com/.

[31] Report to congress on server and data center energy efficiency, environmental protection agency, 2007. www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf.

[32] Natural resources defense council. http://www.nrdc.org/energy, .

[33] Scaling up energy efficiency a cross the data center industry: evaluating key drivers and barriers, nrdc, august 2014. http://www.nrdc.org/energy/files/data-center-efficiency-assessment-IP.pdf, .

[34] Erica Naone. Conjuring clouds. *Technology Review*, 112(4):54–56, 2009.

[35] . The Green Grid. The Green Grid Data Center Compute Efficiency Metric, DCcE, 2010.

[36] Anne-Cécile Orgerie and Laurent Lefèvre. When Clouds become Green: the Green Open Cloud Architecture. *Parallel Computing*, 19:228 – 237, 2010.

[37] Jan Stoess, Christian Lang, and Frank Bellosa. Energy management for hypervisor-based virtual machines. In *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*, ATC'07, pages 1:1–1:14, Berkeley, CA, USA, 2007. USENIX Association. ISBN 999-8888-77-6.

[38] Ludmila Cherkasova and Rob Gardner. Measuring cpu overhead for i/o processing in the xen virtual machine monitor. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '05, pages 24–24, Berkeley, CA, USA, 2005. USENIX Association.

[39] Suzanne Rivoire, Parthasarathy Ranganathan, and Christos Kozyrakis. A comparison of high-level full-system power models. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower'08, pages 3–3, Berkeley, CA, USA, 2008. USENIX Association.

[40] Preeti Ranjan Panda, Aviral Shrivastava, B.V.N. Silpa, and Krishnaiah Gummidipudi. *Power-Efficient System Design*. Springer, 1st edition edition, 2010.

[41] Seung-Hwan Lim, B. Sharma, Gunwoo Nam, Eun K. Kim, and C. R. Das. MDCSim: A multi-tier data center simulation, platform. In *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, pages 1–9. IEEE, August 2009.

[42] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ISCA '07, pages 13–23, New York, NY, USA, 2007. ACM.

[43] Massoud Pedram and Inkwon Hwang. Power and performance modeling in a virtualized server system. *2012 41st International Conference on Parallel Processing Workshops*, 0:520–526, 2010.

[44] Waltenegus Dargie. A stochastic model for estimating the power consumption of a processor. *IEEE Transactions on Computers*, 99(PrePrints):1, 2014. ISSN 0018-9340.

[45] Robert Basmadjian, Nasir Ali, Florian Niedermeier, Hermann de Meer, and Giovanni Giuliani. A methodology to predict the power consumption of servers in data centres. In *Proceedings of the 2Nd International Conference on Energy-Efficient Computing and Networking*, e-Energy '11, pages 1–10, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-1313-1.

[46] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka A. Bhattacharya. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM symposium on Cloud computing*, SoCC '10, pages 39–50, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0036-0. doi: 10.1145/1807128.1807136.

[47] Dimitris Economou, Suzanne Rivoire, and Christos Kozyrakis. Full-system power analysis and modeling for server environments. In *In Workshop on Modeling Benchmarking and Simulation (MOBS*, 2006.

[48] Taliver Heath, Bruno Diniz, Enrique V. Carrera, Wagner Meira, Jr., and Ricardo Bianchini. Energy conservation in heterogeneous server clusters.

In *Proceedings of the Tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '05, pages 186–195, New York, NY, USA, 2005. ACM.

[49] Ramya Raghavendra, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang, and Xiaoyun Zhu. No "power" struggles: Coordinated multi-level power management for the data center. *SIGARCH Comput. Archit. News*, 36(1):48–59, March 2008. ISSN 0163-5964.

[50] Anton Beloglazov and Rajkumar Buyya. Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, MGC '10, pages 4:1–4:6, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0453-5. doi: 10.1145/1890799.1890803.

[51] Gaurav Dhiman, Kresimir Mihic, and Tajana Rosing. A system for online power prediction in virtualized environments using gaussian mixture models. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 807–812, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0002-5.

[52] Ata E. H. Bohra and Vipin Chaudhary. Vmeter: Power modelling for virtualized clouds. In *IPDPS Workshops*, pages 1–8. IEEE, 2010.

[53] Xiao Peng and Zhao Sai. A low-cost power measuring technique for virtual machine in cloud environments, 2013.

[54] Bhavani Krishnan, Hrishikesh Amur, Ada Gavrilovska, and Karsten Schwan. Vm power metering: Feasibility and challenges. *SIGMETRICS Perform. Eval. Rev.*, 38(3):56–60, January 2011.

[55] Anja Strunk and Waltenegus Dargie. Does live migration of virtual machines cost energy? In *27th IEEE International Conference on Advanced Information Networking and Applications, AINA 2013, Barcelona, Spain, March 25-28, 2013*, pages 514–521, 2013.

[56] William Voorsluys, James Broberg, Srikumar Venugopal, and Rajkumar Buyya. Cost of virtual machine live migration in clouds: A performance evaluation. In *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, pages 254–265, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-10664-4. doi: 10.1007/978-3-642-10665-1_23.

[57] Kateryna Rybina, Waltenegus Dargie, Anja Strunk, and Alexander Schill. Investigation into the energy cost of live migration of virtual machines. In *Sustainable Internet and ICT for Sustainability, SustainIT 2013, Palermo, Italy, 30-31 October, 2013, Sponsored by the IFIP TC6 WG 6.3 "Performance of Communication Systems"*, pages 1–8, 2013.

[58] Wenjin Hu, Andrew Hicks, Long Zhang, Eli M. Dow, Vinay Soni, Hao Jiang, Ronny Bull, and Jeanna N. Matthews. A quantitative study of virtual machine live migration. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, CAC '13, pages 11:1–11:10, New York, NY, USA, 2013. ACM.

[59] Haikun Liu, Cheng-Zhong Xu, Hai Jin, Jiayu Gong, and Xiaofei Liao. Performance and energy modeling for live migration of virtual machines. In *Proceedings of the 20th International Symposium on High Performance Distributed Computing*, HPDC '11, pages 171–182, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0552-5.

[60] Qiang Huang, Fengqian Gao, Rui Wang, and Zhengwei Qi. Power consumption of virtual machine live migration in clouds. In *Proceedings of the 2011 Third International Conference on Communications and Mobile Computing*, CMC '11, pages 122–125, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4357-4.

[61] Anja Strunk. A lightweight model for estimating energy cost of live migration of virtual machines. In *2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, June 28 - July 3, 2013*, pages 510–517, 2013.

[62] The G. Grid. The Green Grid Data Center Power Efficiency Metrics: PUE and DCiE. Technical report, 2007.

[63] Intel's cloud computing 2015 vision. http://www.intel.com/Assets/PDF/whitepaper/cloud_vision.pdf.

[64] Padmanabhan Pillai and Kang G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. pages 89–102, 2001.

[65] Tom Guerout, Thierry Monteil, Georges Da Costa, Rodrigo N. Calheiros, Rajkumar Buyya, and Mihai Alexandru. Energy-aware simulation with dvfs. *Simulation Modelling Practice and Theory*, 39:76–91, 2013.

URL http://dblp.uni-trier.de/db/journals/simpra/simpra39.html#GueroutMCCBA13.

[66] Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, and Albert Y. Zomaya. A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in Computers*, 82:47–111, 2011.

[67] Qi Zhang, Mohamed Faten Zhani, Shuo Zhang, Quanyan Zhu, Raouf Boutaba, and Joseph L. Hellerstein. Dynamic energy-aware capacity provisioning for cloud computing environments. In *Proceedings of the 9th International Conference on Autonomic Computing*, ICAC '12, pages 145–154, New York, NY, USA, 2012. ACM.

[68] Gong Chen, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao, and Feng Zhao. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 337–350, Berkeley, CA, USA, 2008. USENIX Association. ISBN 111-999-5555-22-1. URL http://dl.acm.org/citation.cfm?id=1387589.1387613.

[69] Brian Guenter, Navendu Jain, and Charles Williams. Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. In *in Proc. of the 30st Annual IEEE Intl. Conf. on Computer Communications (INFOCOM*, pages 1332–1340.

[70] Dara Kusic, Jeffrey O. Kephart, James E. Hanson, Nagarajan Kandasamy, and Guofei Jiang. Power and performance management of virtualized computing environments via lookahead control. In *Proceedings of the 2008 International Conference on Autonomic Computing*, ICAC '08, pages 3–12, Washington, DC, USA, 2008. IEEE Computer Society.

[71] Anne-Cécile Orgerie and Laurent Lefèvre. ERIDIS: Energy-efficient Reservation Infrastructure for large-scale DIstributed Systems. *Parallel Processing Letters*, 21(2):133–154, June 2011.

[72] Anton Beloglazov and Rajkumar Buyya. Energy efficient resource management in virtualized cloud data centers. In *Proceedings of the 2010 10th*

IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGRID '10, pages 826–831, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4039-9. doi: 10.1109/CCGRID.2010.46.

[73] Aziz Murtazaev and Sangyoon Oh. Sercon : Server consolidation algorithm using live migration of virtual machines for green computing. *Iete Technical Review*, 28(3):212–231, 2011.

[74] 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 *Proceedings of the 2009 IEEE International Conference on Cloud Computing*, CLOUD '09, pages 17–24, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3840-2. doi: 10.1109/CLOUD.2009.72.

[75] Lskrao Chimakurthi and Madhu Kumar SD. Power efficient resource allocation for clouds using ant colony framework. *CoRR*, abs/1102.2608, 2011.

[76] Jonathan G Koomey. Estimating total power consumption by servers in the u.s. ad the world. Technical report, Lawrence Berkeley National Laboratory and Consulting Professor, Stanford University, 2007.

[77] Haizea. http://haizea.cs.uchicago.edu/.

[78] Borja Sotomayor, Kate Keahey, and Ian Foster. Combining batch execution and leasing using virtual machines. In *Proceedings of the 17th International Symposium on High Performance Distributed Computing*, HPDC '08, pages 87–96, 2008. ISBN 978-1-59593-997-5.

[79] Amit Nathani, Sanjay Chaudhary, and Gaurav Somani. Policy based resource allocation in iaas cloud. *Future Gener. Comput. Syst.*, 28(1):94–103, January 2012. ISSN 0167-739X. doi: 10.1016/j.future.2011.05.016.

[80] Shyamala Loganathan and Saswati Mukherjee. Differentiated policy based job scheduling with queue model and advanced reservation technique in a private cloud environment. In *GPC*, pages 32–39, 2013.

[81] Open nebula. http://www.opennebula.org/, .

[82] Eucalyptus. http://open.eucalyptus.com/.

[83] Tien Van Do. Comparison of allocation schemes for virtual machines in energy-aware server farms. *Comput. J.*, 54(11):1790–1797, 2011.

[84] Michele Mazzucco, Dmytro Dyachuk, and Ralph Deters. Maximizing cloud providers' revenues via energy aware allocation policies. In *IEEE International Conference on Cloud Computing, CLOUD 2010, Miami, FL, USA, 5-10 July, 2010*, pages 131–138, 2010.

[85] Tien Van Do and Udo R. Krieger. A performance model for maintenance tasks in an environment of virtualized servers. In *NETWORKING 2009, 8th International IFIP-TC 6 Networking Conference, Aachen, Germany, May 11-15, 2009. Proceedings*, pages 931–942, 2009.

[86] Isi Mitrani. Service center trade-offs between customer impatience and power consumption. *Perform. Eval.*, 68(11):1222–1231, 2011.

[87] Isi Mitrani. Managing performance and power consumption in a server farm. *Annals OR*, 202(1):121–134, 2013.

[88] Tien Van Do and Csaba Rotter. Comparison of scheduling schemes for on-demand iaas requests. *Journal of Systems and Software*, 85(6):1400–1408, 2012.

[89] Gregor von Laszewski, Lizhe Wang, Andrew J. Younge, and Xi He. Power-aware scheduling of virtual machines in dvfs-enabled clusters. In *Proceedings of the 2009 IEEE International Conference on Cluster Computing, August 31 - September 4, 2009, New Orleans, Louisiana, USA*, pages 1–10, 2009.

[90] Dang Minh Quan, Robert Basmadjian, Hermann de Meer, Ricardo Lent, Toktam Mahmoodi, Domenico Sannelli, Federico Mezza, Luigi Telesca, and Corentin Dupont. Energy efficient resource allocation strategy for cloud data centres. In *Computer and Information Sciences II - 26th International Symposium on Computer and Information Sciences, London, UK, 26-28 September 2011*, pages 133–141, 2011.

[91] Ying Song, Hui Wang, Yaqiong Li, Binquan Feng, and Yuzhong Sun. Multi-tiered on-demand resource scheduling for vm-based data center. In *9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGrid 2009, Shanghai, China, 18-21 May 2009*, pages 148–155, 2009.

[92] Akshat Verma, Puneet Ahuja, and Anindya Neogi. pmapper: Power and migration cost aware application placement in virtualized systems. In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '08, pages 243–264, New York, NY, USA, 2008. Springer-Verlag New York, Inc. ISBN 3-540-89855-7.

[93] 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*, VEE '09, pages 41–50, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-375-4.

[94] Tiago C Ferreto, Marco A S Netto, Rodrigo N Calheiros, and César A F De Rose. Server consolidation with migration control for virtualized data centers. *Future Generation Computer Systems*, 27(8):1027–1034, 2011.

[95] Edward G Coffman, Janos Csirik, and Gerhard Woeginger. *Approximate Solutions to Bin Packing Problems*. Oxford University Press, 2002.

[96] Ching-Chi Lin, Pangfeng Liu, and Jan-Jan Wu. Energy-efficient virtual machine provision algorithms for cloud systems. In *IEEE 4th International Conference on Utility and Cloud Computing, UCC 2011, Melbourne, Australia, December 5-8, 2011*, pages 81–88, 2011.

[97] Shingo Takeda and Toshinori Takemura. A rank-based vm consolidation method for power saving in datacenters. *Information and Media Technologies*, 5(3):994–1002, 2010.

[98] Cloud application platform. https://www.heroku.com/.

[99] Cloudfoundry. https://www.cloudfoundry.org/.

[100] Openshift. https://www.openshift.com/.

[101] IBM ILOG CPLEX (August 2012). http://www.aimms.com/cplex-solver-for-linear-programming2?gclid=CLKI-56wrbACFVMetAodGRPLVA.

[102] . Green IT: Making the Business Case, Cognizant Report, January 2011.

[103] Ramon Bertran Yol, Xavier Martorell, Jordi Torres, and Eduard Ayguade. Technical report upc-dac-rr-cap-18 accurate energy accounting for shared virtualized environments using pmc-based power modeling techniques.

[104] http://www.spec.org/power_ssj2008/.

[105] Chaima Ghribi, Makhlouf Hadji, and Djamal Zeghlache. Energy efficient vm scheduling for cloud data centers: Exact allocation and migration algorithms. In *CCGRID*, pages 671–678. IEEE Computer Society, 2013. ISBN 978-1-4673-6465-2.

[106] Kate Smith-Miles, Brendan Wreford, Leo Lopes, and Nur Insani. Predicting metaheuristic performance on graph coloring problems using data mining. In El-Ghazali Talbi, editor, *Hybrid Metaheuristics*, volume 434 of *Studies in Computational Intelligence*, pages 417–432. Springer, 2013.

[107] D. Gómez, J. Montero, J. Yáñez, and C. Poidomani. A graph coloring approach for image segmentation. *Omega*, 35(2):173 – 183, 2007.

[108] G. J. Chaitin. Register allocation & spilling via graph coloring. *SIGPLAN Not.*, 17(6):98–101, June 1982. ISSN 0362-1340. doi: 10.1145/872726.806984.

[109] G. A. Neufeld and J. Tartar. Graph coloring conditions for the existence of solutions to the timetable problem. *Commun. ACM*, 17(8):450–453, August 1974. ISSN 0001-0782.

[110] Robert James Waters. Graph colouring and frequency assignment. In *Doctoral thesis, University of Nottingham*, 2005.

[111] Miklos Biro, Mihaly Hujter, and Zsolt Tuza. Cross fertilisation of graph theory and aircraft maintenance scheduling. In *AGIFORS*, pages 307–318, 1992.

[112] https://www.sgi.com/pdfs/4301.pdf/.

[113] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, 2 edition, September 2000. ISBN 0130144002.

[114] Lizhe Wang and Samee U. Khan. Review of performance metrics for green data centers: a taxonomy study. *J. Supercomput.*, 63(3):639–656, March 2013. ISSN 0920-8542. doi: 10.1007/s11227-011-0704-3.