Integrated optimization in cloud environment

Hana Teyeb

NNT : 2017SACLL010

Optimisation Intégrée dans un Environnement Cloud

Thèse de doctorat de l'Université Paris-Saclay
Préparée à Télécom SudParis et la Faculté des Sciences de Tunis

École doctorale n°580 : Sciences et technologies de l'information et de la communication (STIC)

Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Tunis, le 18-12-2017, par

Hana Teyeb

Composition du Jury:

Samir BEN AHMED Professeur, FST, Université de Tunis-El Manar, Tunisie	Président
Pierre SENS Professeur, UMPC, France	Rapporteur
Jouhaina CHAOUACHI Professeur, IHEC, Université de Carthage, Tunisie	Rapporteur
Sourour ELLOUMI Maître de conférences, HDR, ENSTA, France	Examinatrice
Samir Tata Professeur, Télécom SudParis, (Samovar)	Directeur de thèse
Nejib Ben Hadj-Alouane Professeur, ENIT, Université de Tunis-El Manar, Tunisie (OASIS)	Co-Directeur de thèse

Thèse de doctorat

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Optimisation intégrée dans un environnement Cloud

Mots clés : Cloud ; machine virtuelle ; communication ; migration ; placement ; ordonnancement ; programmation linéaire.

Résumé : Dans les systèmes cloud géographiquement distribués, un défi majeur auquel sont confrontés les fournisseurs de cloud consiste à optimiser et à configurer leurs infrastructures. En particulier, cela consiste à trouver un emplacement optimal pour les machines virtuelles (VMs) afin de minimiser les coûts tout en garantissant une bonne performance du système. De plus, en raison des fluctuations de la demande et des modèles de trafic, il est essentiel d'ajuster dynamiquement le schéma de placement des VMs en utilisant les techniques de migration des VMs. Cependant, malgré ses avantages apportés, dans le contexte du Cloud géo-distribué, la migration des VMs génère un trafic supplémentaire dans le réseau backbone ce qui engendre la dégradation des performances des applications dans les centres de données (DCs) source et destination. Par conséquent, les décisions de migration doivent être bien étudiés et basées sur des paramètres précis.

Dans ce manuscrit, nous étudions les problèmes d'optimisation liés au placement, à la migration et à l'ordonnancement des VMs qui hébergent des applications hautement corrélées et qui peuvent être placés dans des DCs géo-distribués. Dans ce contexte, nous proposons un outil de gestion de DC autonome basé sur des modèles d'optimisation en ligne et hors ligne pour gérer l'infrastructure distribuée du Cloud. Notre objectif est de minimiser le volume du trafic global circulant entre les différents DCs du système.

Nous proposons également des modèles d'optimisation stochastiques et déterministes pour traiter les différents modèles de trafic de communication. En outre, nous fournissons des algorithmes quasi-optimales qui permettent d'avoir la meilleure séquence de migration inter-DC des machines virtuelles inter-communicantes. En plus, nous étudions l'impact de la durée de vie des VMs sur les décisions de migration afin de maintenir la stabilité du Cloud. Enfin, nous utilisons des environnements de simulation pour évaluer et valider notre approche. Les résultats des expériences menées montrent l'efficacité de notre approche.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Title : Integrated Optimization in Cloud Environment

Keywords: Cloud; virtual machine; communication; migration; placement; scheduling; linear programming.

Abstract: In geo-distributed cloud systems, a key challenge faced by cloud providers is to optimally tune and configure their underlying cloud infrastructure. An important problem in this context, deals with finding an optimal virtual machine (VM) placement, minimizing costs while at the same time ensuring good system performance. Moreover, due to the fluctuations of demand and traffic patterns, it is crucial to dynamically adjust the VM placement scheme over time. Hence, VM migration is used as a tool to cope with this problem. However, despite the benefits brought by VM migration, in geo-distributed cloud context, it generates additional traffic in the backbone links which may affect the application performance in both source and destination DCs. Hence, migration decisions need to be effective and based on accurate parameters. In this work, we study optimization problems related to the placement, migration and scheduling of VMs hosting highly correlated and distributed applications within geo-distributed DCs. In this context, we propose an autonomic DC management tool based on both online and offline optimization models to manage the distributed cloud infrastructure. Our objective is to minimize the overall expected traffic volume circulating between the different DCs of the system.

To deal with different types of communication traffic patterns, we propose both deterministic and stochastic optimization models to solve VM placement and migration problem and to cope with the uncertainty of inter-VM traffic. Furthermore, we propose near-optimal algorithms that provide with the best inter-DCs migration sequence of inter-communicating VMs. Along with that, we study the impact of the VM's lifetime on the migration decisions in order to maintain the stability of the cloud system. Finally, to evaluate and validate our approach, we use experimental tests as well as simulation environments. The results of the conducted experiments show the effectiveness of our proposals.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Dedication

To the memory of my grandmother,
To my parents and lovely sister,
To my husband,

Acknowledgements

Foremost, I would like to thank my advisors, Prof. Nejib BEN HADJ-ALOUANE and Prof. Samir TATA, for their outstanding support without which this thesis would not have been possible. Despite their many responsibilities, they have always found time to provide feedbacks, new ideas and suggestions on my work. I would like to thank them for allowing me to grow as a research scientist and for giving me wonderful opportunities to actively participate in renowned scientific events and conferences.

I would like to extend my greatest gratitude to the members of the thesis committee for accepting to revue my work. I would like to thank the committee president Prof. Samir BEN AHMED for accepting to chair my thesis defense, the reading committee members Prof. Jouhaina CHAOUACHI and Prof. Pierre SENS for their interest and Prof. Sourour ELLOUMI for evaluating my work and being part of the defense committee.

A very special thanks to Dr. Ali Balma for his excellent advising and support during the last three years. His wise guidance, for sure, helped me widen my research from various perspectives. Without his insightful comments, it would not have been possible to conduct this research. I want to express my sincere thanks to the laboratory head, Prof. Atidel BEN HADJ-ALOUANE, who provided me with the opportunity to join her team, who gave me access to research facilities and for her help and her precious advices.

I also want to acknowledge all the people who have contributed to this work. Special thanks go to the members of OASIS research lab whom I have had the pleasure to work with and share many memories. Many thanks to Amina BOUROUIS, Anis ZEMNI and Walid HFAIEDH for all your help.

I express my sincere gratitude to all my family: my parents, my sister and my husband, who have been always a great support and have pushed me toward through hard times. Without you, none of this would have been possible. Thank you for being part of my life.

Notations

$\boldsymbol{D}$ The set of data centers.

$\boldsymbol{V}$ The set of virtual machines.

$\boldsymbol{R}$ The set of hardware resources (CPU, RAM, storage).

$\boldsymbol{d_{ij}}$ The amount of traffic exchanged between the VM i and the VM j ($i \in V$, $j \in V$).

$\boldsymbol{a_i^k}$ Takes 1 if the VM i can be placed in the DC k, 0 otherwise ($i \in V$, $k \in D$).

$\boldsymbol{cap_r^k}$ The capacity of the DC k in terms of resource r ($k \in D$, $r \in R$).

$\boldsymbol{u_{ir}}$ The amount of resource r consumed by the VM i ($r \in R$, $i \in V$).

$\boldsymbol{M_i}$ The migration cost of VM $i \in V$.

List of Acronym

AM Autonomic Manager

AG Aggregated Formulation

CF Classical Formulation

CLT Central Limit Theorem

DC Data Center

DVMP Dynamic Virtual Machine Placement

EC2 Amazon Elastic Computing Cloud

EDFA Erbium-Doped Fiber Amplifier

GIS Geographic Information System

GP Geometric programming

HPC High Performance Computing

HL Hub Location

IaaS Infrastructure as a Service

ILP Integer Linear Program

IPs Infrastructure Providers

IoT Internet of Things

IVMP Initial Virtual Machine Placement

LAN Local Area Network

LP Linear Programming

MILP Mixed Integer Linear Program

MF Multi Flows

NIST National Institute of Standards and Technology

NLP Non linear programming

OM Out of Memory

PaaS Platform as a Service

QoS Quality of Service

QP Quadratic Programming

SaaS Software as a Service

SAA Sample Average Approximation

SIP Stochastic Integer Program

SLA Service Level Agreement

SPs Service Providers

ToR Top-of-Rack

TSP Traveling Salesman Problem

VM Virtual Machine

VMMS Virtual Machine Migration Scheduling

WAN Wide Area Network

WDM Wavelength Division Multiplexing

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 General Context

Cloud computing has emerged during the last years as a new adopted paradigm where tenants can benefit from on-demand computing resources provided in a pay-as-you-go manner [1]. Cloud computing is based on virtualization technologies that enable the resource sharing. In fact, virtualization aims at partitioning physical resources into logical resources that can be allocated to applications in a flexible manner. For instance, server virtualization is a technology that partitions the physical machine into multiple virtual machines (VMs) and allows it to be executed on the same physical host [2].

In such an environment, tenants may benefit from computing resources including processing, memory, storage and networking. The adoption of this paradigm provides many benefits such as cost savings, reliability, and scalability [1]. As a result, cloud computing services are increasingly attracting companies to move their business into the cloud. Consequently, the number of applications that are running on the different VMs has also increased considerably [3, 4].

In order to achieve reliability and serve world-wide users, large-scale cloud providers are relying on a geographically distributed infrastructure where data centers (DCs) are built in different locations and interconnected with a backbone network [5]. In a geo-distributed cloud infrastructure, DCs are hosting different types of distributed

applications including web applications, scientific workflows, parallel processing applications, etc. These applications are generally deployed among a number of VMs that can be placed into distant DCs. In many cases, they may present a high communication level between each other which could potentially produce a huge amount of traffic exchange.

With the rise of cloud services popularity, the number of communication-intensive applications has been growing considerably which has resulted in the increase of the amount of inter-VMs traffic. Moreover, due the heterogeneity of the applications hosted in cloud DCs, there exist wide variety of communication patterns ranging from one-to-one and all-to-all traffic matrices. As a consequence, it is crucial to take into consideration the different characteristics of inter-VMs traffic.

The efficiency of the DCs as well as the performance of the hosted applications depend highly on the resource allocation and the placement scheme of the different VMs [6]. One of the key challenges faced by cloud providers is network-aware VM placement and migration problem which includes online/offline placement decisions, migration scheduling decisions, minimization of inter-DCs traffic, risk management, etc. This problem has attracted much attention in recent years as it aims to optimize the cloud configuration while satisfying different objectives such as efficient resource utilization, reducing energy consumption and minimizing network traffic.

1.2 Motivation and Problem Statement

Managing a geo-distributed infrastructure requires the cloud providers to solve a number of challenges. A key challenge faced by the cloud providers is to optimize the cloud infrastructure, which involves the optimization of the placement scheme of the different VMs in the system. In fact, with the rise of cloud services popularity, cloud computing-based traffic has been rapidly growing in recent years. Indeed, the number of VMs that are hosting applications with critical network requirements (e.g. message-based applications, web applications, video streaming servers, etc.) has also increased. These applications are characterized by their large data volume which will result in a high amount of communication traffic between DCs.

Most cloud Service Providers (SPs) are relying on Infrastructure Providers (IPs) in order to connect their geo-distributed DCs. The backbone network is owned and managed by the IPs. SPs are charged based on the total network Input/Output of data transferred through the backbone links (i.e. from and to cloud servers) [7].

According to many recent studies [8, 9, 10], inter-DCs traffic is usually significantly more expensive than intra-DC traffic. In fact, as shown in [11], communication costs are around 15% of operational expenditure incurred to a cloud provider. Based on runtime measurements, the study presented in [12] shows that inter-DC traffic accounts for up to 45% of the total traffic going through DC edge routers. In addition, inter-VMs communication traffic is considered as one of the dominating costs for communication intensive distributed applications [13]. Thus, it is important to investigate how to maintain the inter-DCs traffic as minimum as possible. Indeed, optimizing the VM placement has also an impact on the DCs energy consumption. Efficient placement decisions will reduce inevitably the amount of energy consumed as well as the amount of traffic transferred between geo-distributed DCs.

However, cloud systems are highly dynamic, the demand is changing constantly making thus, current placement scheme ineffective. To tackle this problem, VM

migration techniques are commonly used in order to re-optimize the configuration of the cloud system. In fact, VM migration is used as a tool to cope with the demand fluctuations and the dynamic aspects of traffic patterns. As a matter of fact, VM migration brings with it many benefits; (1) it provides flexibility in the management of a DC, and (2) it enables moving VMs across DCs in order to adjust and optimize the cloud infrastructure.

However, despite the benefits brought by VM migration technology, it rises also many challenges. During the migration process, an additional traffic is sent through the network links [2]. In addition, the performance of the VMs in source as well as in destination can be affected during this process, especially, if the VM is migrated from a DC to a distant one over a bandwidth-constrained network links. Hence, it is important to ensure reliability and maintain system performance during the migration process.

Efficient cloud DCs management has become a very complex task particularly, for geographically distributed DCs. The main factors for such a complexity are the heterogeneity of the VMs and their critical QoS requirements mentioned in the SLAs. In this context, several crucial decisions need to be taken by the cloud manager:

- Where to place VMs while minimizing the inter-DCs traffic?
- When a system reconfiguration is needed?
- Which VM needs to be migrated and to which DC?
- If a set of inter-communicating VMs needs to be migrated, what is the best migration sequence that minimizes the overall communication traffic?
- How to prevent from network overloading in the future while inter-VMs traffic is uncertain? and how to minimize this risk?

Several attempts have been made over the past years to study the VM placement and migration problem. However, most of the existing works were based on heuristics and approximation algorithms which do not provide optimal solutions for the problem. In addition, there are only few works that have considered the problem within a geo-distributed cloud infrastructure where the minimization of the inter-DCs traffic is a rising challenge. Moreover, the optimization of the VM placement includes many other challenges to solve, for instance, offline and online placement decisions, migration scheduling decisions, placement decisions with respect to the uncertainty of inter-VMs traffic, etc.

In order to make the optimal decisions to answer the above questions, we propose a solution based on an autonomic management system. Autonomic computing systems are capable of self-managing themselves by doing self-configuration and self-optimization [14]. Such a system must be able to analyze itself at runtime, determine its state and determine a desired state that maintains the QoS. The proposed tool is based on optimization model providing the optimal solution for the VM placement problem. To tackle the introduced sub-goals this thesis makes the following contributions.

1.3 Objectives and Contributions

The main goal of this thesis is to propose a DC management tool based on network-aware optimization programs that aim to provide, short as well as long-term, optimal placement, migration and scheduling decisions for the different VMs within a geo-distributed cloud infrastructure. The objective of these optimization programs is to minimize the expected traffic volume circulating between the different DCs.

To tackle the introduced objectives, this thesis makes the following contributions:

- Traffic-aware offline/static optimization of the VM placement scheme in geo-distributed DCs.
- Traffic-aware online/dynamic optimization of the VM placement scheme in geo-distributed DCs.
- Traffic-aware inter-DCs VM migration scheduling optimization.
- Proactive optimization of the VM placement scheme for risk management with uncertainty.

1.3.1 Traffic-aware VM Placement and Migration Problem

Finding the optimal placement and migration scheme is a challenging task. An effective VM placement and migration plan can lower the energy consumption and improve the whole system performance [3]. In our work, we have divided the problem into two sub-problems.

First, we study the *Offline* VM placement problem where we consider that the VMs will be placed for the first time in the cloud system. We propose exact offline optimization programs that provide optimal placement scheme for the different VMs while at the same time minimizing the inter-DCs traffic volume. Moreover, we use different formulation strengthening techniques to reduce the computational time of the proposed programs.

Then, we focus on the *Online* version of the problem which involves the VM migration. We propose exact online optimization models that aim to find optimal placement and migration plans while ensuring minimum backbone traffic.

Finally, in order to show the effectiveness of our approach, we use both experimental tests and simulation tools.

1.3.2 Traffic-aware VM Migration Scheduling Problem

The migration of inter-communicating VMs over the backbone network can lead to the increase of the traffic on the network links. Hence, it is important to find the best migration sequence of VMs that minimizes the communication traffic. Hence, we prevent network link congestion and maintain the performance of both VMs in the source and destination as well as the migrating VM.

Because of the challenges risen by VM migration in geo-distributed cloud infrastructure, we propose near-optimal heuristics that provide effective migration scheduling of inter-communicating VMs. Furthermore, extensive migrations may impact the whole system performance, hence, it is crucial to keep the number of migrations as small as possible. Besides, VMs have a finite execution time, thus,

it is interesting to study the impact of this parameter on the migration decisions. Thus, we propose both exact and heuristic solutions to solve this problem.

1.3.3 Proactive VM Placement and Migration Problem for Risk Management

Recent studies [15, 16, 17] have shown that the inter-VMs traffic is highly dynamic and bursty which may cause the existent placement and migration schemes to be inefficient. In addition, most of the existent works [18, 19, 20] make migration decisions based on deterministic demand estimation and workload characterization without considering stochastic properties. Many traffic-intensive applications have highly non-uniform communication traffic patterns. For these reasons, placement and migration decisions must be predictive and considering the different levels of risk that can occur in the future. One of the main issues that we have studied in this thesis is the network overloading problem. Due to the uncertainty of inter-VMs traffic, the risk of overloading DC edge routers is very high. Therefore, placement and migration decisions need to be proactive in order to minimize this risk.

Hence, we propose stochastic exact and heuristic optimization programs to deal with the VM placement and migration problem within a geo-distributed cloud infrastructure. We further consider network overloading probability constraints to minimize the risk of network congestion problem in the future.

1.4 List of Publications

The work presented in this thesis has let to the following publications[1]:

- **H. Teyeb**, N. B. Hadj-Alouane, S. Tata, and A. Balma, "Optimal dynamic placement of virtual machines in geographically distributed cloud data centers," *International Journal of Cooperative Information Systems*, p. 1750001, 2017. SJR 0.269.

- **H. Teyeb**, N. B. Hadj-Alouane, and S. Tata, "Network-aware Stochastic Virtual Machine Placement in Geo-distributed Data Centers," in *25th International Conference on Cooperative Information Systems CoopIs, Rhodes, Greece* 2017, rank A.

- **H. Teyeb**, A. Balma, S. Tata, and N. B. Hadj-Alouane, "Traffic-aware virtual machine migration scheduling problem in geographically distributed data centers," in *IEEE CLOUD, San Francisco, USA, 2016*, 2016, rank B.

- R. Benali, **H. Teyeb**, A. Balma, S. Tata, and N. B. Hadj-Alouane, "Evaluation of traffic-aware VM placement policies in distributed cloud using CloudSim," in *WETICE*. IEEE Computer Society, 2016, pp. 95–100, rank B.

- **H. Teyeb**, A. Balma, N. B. Hadj-Alouane, and S. Tata, "Optimal virtual machine placement in large-scale cloud systems," in *IEEE CLOUD, Anchorage, AK, USA*. IEEE, 2014, pp. 424–431, rank B.

[1] The conferences ranking is based on the CORE 2017 classification available at `http://portal.core.edu.au/conf-ranks/`

- **H. Teyeb**, A. Balma, N. B. Hadj-Alouane, and S. Tata, "Optimal virtual machine placement in a multi-tenant cloud," in *ICSOC Workshops*, ser. Lecture Notes in Computer Science, vol. 8954. Springer, 2014.

- **H. Teyeb**, A. Balma, N. B. Hadj-Alouane, S. Tata, and A. B. Hadj-Alouane, "Traffic-aware virtual machine placement in geographically distributed clouds," in *International Conference on Control, Decision and Information Technologies, CoDIT 2014, Metz, France, November 3-5, 2014*. IEEE, 2014, pp. 24–29.

This work has also been the subject of a patent in collaboration with IBM, San Francisco, USA.

- **H. Teyeb**, A. Balma, N. B. Hadj-Alouane, S. Tata, M. Mohamed, A. Megahed, "Optimal Dynamic Placement of Virtual Machines in Geographically Distributed Cloud Data Centers," U.S. Patent Application Number 15/493,034; Filed on April 20, 2017.

1.5 Outline of the Thesis

This thesis is organized as follows:

- Chapter 2 covers the state of the art as well as some background knowledge necessary for understanding our work. First, it presents the general context of the thesis by giving a brief background on cloud computing, inter-VM communication, VM migration and autonomic computing. Then, existing optimization approaches for the VM placement problem are reviewed and a summary outlining our approach is given. Finally, an overview of the system model used throughout this thesis is presented and explained.

- Chapter 3 presents offline optimization programs proposed to solve the VM placement problem in geo-distributed DCs. The objective of these models is to minimize the inter-DCs network traffic. We give the different mathematical formulations to solve the problem which we have enhanced using formulation strengthening techniques. Then, we present and analyze the results.

- In Chapter 4, we present online optimization programs to solve the VM placement and migration problem within a geo-distributed cloud infrastructure. Our objective is to minimize the inter-DCs network traffic. Simulation-based evaluation is provided as well as experimental results, showing the effectiveness of the proposed approach.

- Chapter 5 presents exact as well as heuristic solutions to solve the VM scheduling problem in geo-distributed DCs. First, we focus on finding the best inter-DCs migration sequence of inter-communicating VMs. Then, we study the impact of VM's lifetime period on the migration decisions. Experiment results are presented illustrating the effectiveness of the proposed solutions.

- As for Chapter 6, it presents online stochastic optimization models to solve the VM placement and migration problem while considering bandwidth capacity

constraints on DC's edge routers. The proposed algorithm aims to proactively optimize placement and migration decisions in order to minimize the risk of network overloading in the future while at the same time minimize the expected inter-DCs traffic.

- Chapter 7 concludes this manuscript by summarizing our contributions and presenting future research directions.

Chapter 2

Background and State of the Art

Contents

2.1 Introduction

In this chapter, we present the general context of the thesis by giving first, a brief background about cloud computing and cloud DCs. Second, a description of the different concepts needed for understanding the rest of the given work . Among these concepts, we define inter-VM communication and VM migration techniques. Third, we introduce the concept of autonomic computing. Then, we review and compare existing VM placement approaches. Finally, we outline our contributions and we present the system model used throughout this thesis.

2.2 Background and Context

In this section, we present some background related to our work. First, we introduce the context of our work which is the cloud computing. Then, we present some

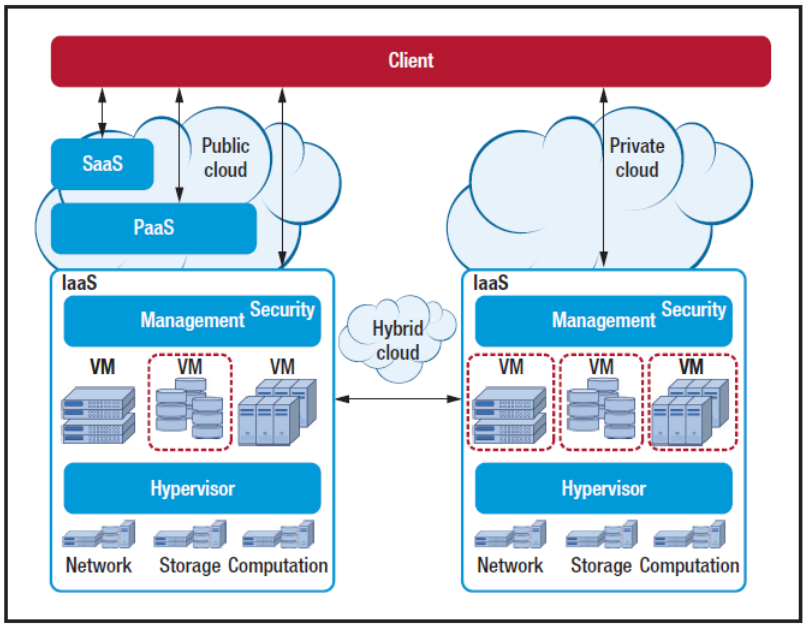


Figure 2.1: Cloud models [21].

characteristics of the cloud DCs that are necessary for the comprehension of our work. Afterwords, we provide a brief description on the inter-VM communication and VM migration techniques in the cloud. Finally, we introduce the concept of autonomic computing and give some background around autonomic managers.

2.2.1 Cloud Computing

According to the National Institute of Standards and Technology (NIST)[1], Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Based on virtualization technologies, cloud computing has gained popularity in recent years.

Virtualization technologies have several benefits. In fact, they enable efficient resource allocation and management, in order to reduce operational costs while improving application performance and reliability [22]. The aim of virtualization is to partition physical resources into logical resources that can be allocated to applications in a flexible manner. For instance, server virtualization enables the resource sharing and allows to multiple VMs to be executed on the same physical host. The isolation of logical resources from the underlying physical resources, server virtualization enables flexible assignment of workloads to physical machines [23].

Cloud computing has five main characteristics. Namely, broad network access, rapid elasticity, on-demand self-service, resource pooling, and measured service [1].

On demand self services: Cloud services such as email, network or service can be provided without requiring human interaction with the service provider.

Broad network access: Cloud services are available via the network and can be accessed from any networked device.

Resource pooling: The provider's computing resources are shared and serve multiple consumers, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.

Rapid elasticity: Cloud services can be rapidly and elastically provisioned. Customers can automatically provision and release resources whenever required.

Measured service: Cloud providers monitor the customers' resource usage and charge customers for the used resources based on a pay-as-you-go manner.

Cloud computing has mainly three service categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS).

IaaS: In an IaaS model, a third-party provider hosts hardware, software, servers, storage and other infrastructure components on behalf of its users. The IaaS providers also host users' applications and handle tasks including system maintenance, backup and resiliency planning.

PaaS: PaaS platforms offer developers the ability to deploy supported applications onto the cloud. The developer does manage the underlying infrastructure however, it has control on the deployed application and the hosting environment configurations.

SaaS: In SaaS model, consumers are able to access and use software applications running on the cloud infrastructure over the internet. Google, Twitter and Facebook are examples of SaaS.

In [1], the authors categorize the cloud into four deployment models as presented below.

Private Cloud: In this category, the cloud infrastructure belongs to a single institution. Private clouds are either managed by the institution itself or by a third-party. It is characterized by its limited access.

Public Cloud: Public clouds are commercial cloud systems operated and managed by public cloud providers. They allow worldwide customers to provision services and charge them in a pay-as-you-go manner.

Community Cloud: Community clouds allow infrastructure sharing among different institutions having common interests (e.g. security requirements, policy and compliance considerations).

Hybrid Cloud: Hybrid clouds are a combination of two or more cloud infrastructures (private, public or community) that are bound together by standardized technology that enables data and application portability.

Figure (2.1) illustrates the introduced concepts of the cloud computing.

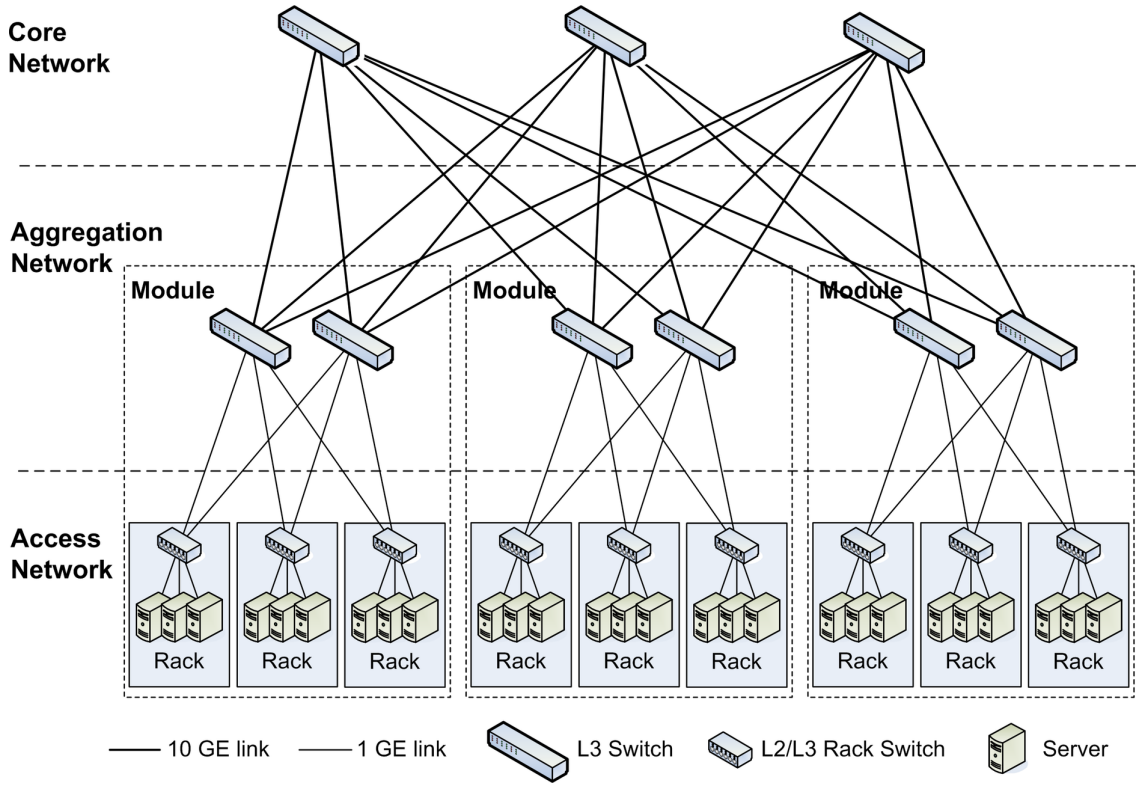


Figure 2.2: Three-tier network architecture [6].

2.2.2 Cloud Data Centers

In this section, we provide background knowledge on modern cloud data centers. In particular, we present common architecture and network topology.

According to [24], DCs are organized in a multi-tiered network hierarchy. Figure (2.2) presents the generic intra-DC network architecture. The network architecture is mainly composed of three tiers where each tier has a specific role in the traffic handling. In the *Access Tier*, every physical server is connected to one or two access switches. In the *Aggregation Tier*, each access switch is connected to the aggregation switches. In the *Core Tier*, each aggregation switch is connected to one or more core switches.

In fact, the access switches are in charge of connecting the servers between each other and to the upper tiers. As for the aggregate switches, they connect the access switches between each other. In addition, they enable the localization of traffic among the servers. Finally, core switches ensure the connection between the aggregation switches in such a way that there exists a connection among each pair of servers. It also includes gateways for the traffic to allow the communication outside the DC.

In modern DCs, servers are generally organized in racks, where each rack has a *Top-of-Rack* (ToR) switch. ToRs are connected to aggregation switches whereas aggregation switches are connected to core switches in the top-tier.

In traditional DCs, Tree network architecture has been widely used because of its simplicity in terms of reducing costs [24]. However, this network architecture may present scaling issues and network congestion problems. To cope with these issues, recent studies have proposed new network architectures, for instance, VL2

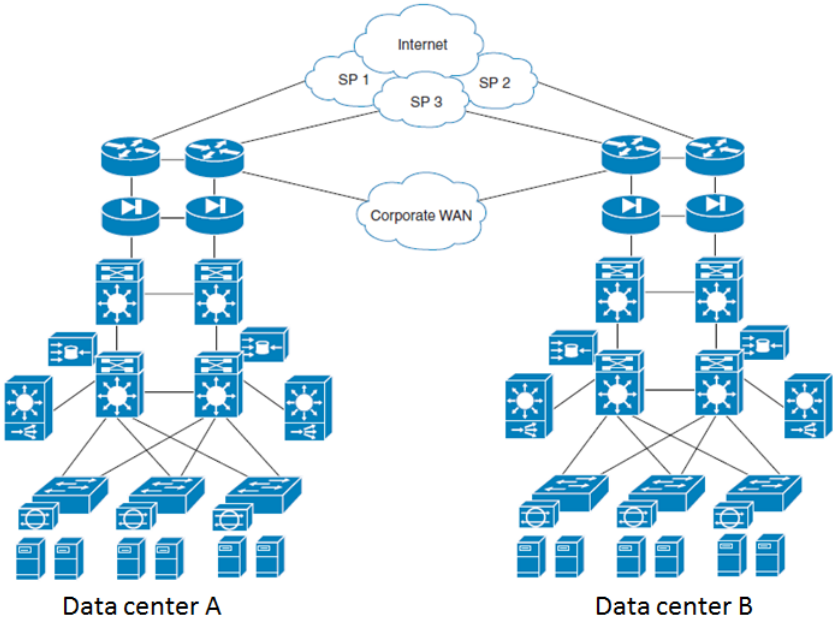


Figure 2.3: Distributed DCs Network Design [25].

[26], BCube [27], PortLand [28].

In order to serve world-wide end users, cloud providers are relying on a geographically distributed infrastructure where DCs are built in different locations. Many public cloud providers have adopted this distributed infrastructure such as Amazon Elastic Computing Cloud (EC2) [29] and Microsoft Azure [30]. This decentralized service delivery architecture provides cost efficiency, ensures adequate Quality of Service (QoS) and avoids potential performance problems [31]. Many studies [32, 33, 31, 34] have shown that significant gains can be obtained from such decentralized approach. According to [25], DC edge routers are responsible for connecting the DC to the backbone network. Figure (2.3) shows an example of distributed DCs network design.

Figure (2.4) shows another example of geo-distributed infrastructure where DCs are connected with an IP over WDM network. The IP over Wavelength Division Multiplexing (WDM) network is composed of two layers: the IP layer, and the optical layer. In the IP layer, each node has an optical switch which is connected to an IP router. The router aggregates data traffic from access networks. The optical layer can provide large capacity and wide bandwidth for data communication between IP routers. Optical switches are connected to optical fiber links. On each fiber, a pair of wavelength multiplexers/demultiplexers is used to multiplex/demultiplex wavelengths. Moreover, the erbium-doped fiber amplifiers (EDFAs) are used to amplify the optical signal in each fiber for long distance transmission [35].

Cloud DCs continue to grow, over the last years, in terms of both hardware resources and traffic volume, thus making cloud operation and management a challenging task. Therefore, the use of management tools is mandatory and very useful in order to help DC administrators managing their infrastructure [36]. Data Center Management tools (DCM) are designed to help organize a company's infrastructure and facilitate the DC management. Most of the existent DCM are used to gather and monitor basic information about energy consumption, cooling etc. Others are used

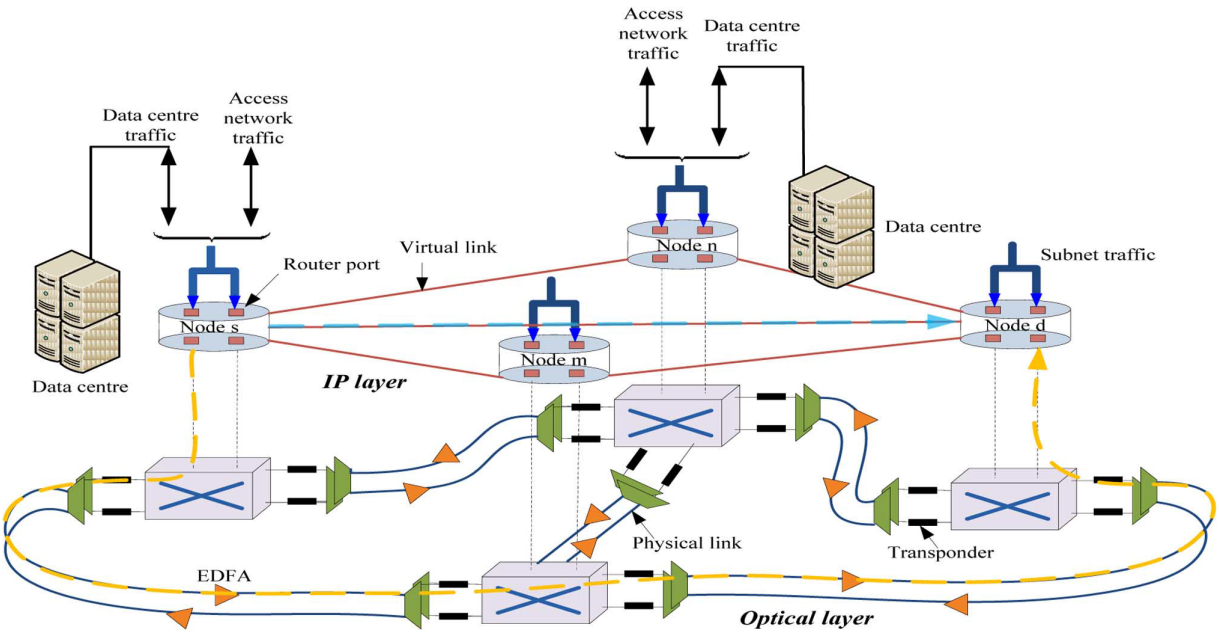


Figure 2.4: IP over WDM network [35].

to help DC administrator planning the capacities of the IT facilities [37]. However, existent commercial DCM tools (e.g CloudWatch [38], LogicMonitor[39], NimSoft [40]) are lacking some important features and present several issues of inefficiency. In fact, to be effective, a DCM should be dynamic and able to adapt itself automatically, on a real-time basis, to the changes in the system. It should be aware of other parameters such as the application workload, virtualization software, network communication, SLAs, QoS, etc [41].

In particular, a DCM tool needs to be able to (re)optimize the DC infrastructure and take automated decisions based on predefined criteria in order to satisfy some business needs [37]. It should have automated and autonomic features that are able to detect, solve and optimize several problems, such as VM placement, migration and scheduling problems which are the focus of this thesis. More details on the proposed DCM tool can be found in Section 2.4.

In the next section, we introduce the concept of inter-VM communication.

2.2.3 Cloud Applications and Communication Models

In geo-distributed cloud environment, cloud providers are deploying their DCs in different locations. The cloud applications deployed in such an infrastructure, such as web applications, scientific workflows and parallel processing applications are composed of several VMs and storage components that present highly correlated communication between them [6]. With the increasing number of traffic-intensive application in the cloud, the inter-VM network bandwidth consumption is increasing considerably. Moreover, the overall application performance depends mainly on the underlying network resources [6].

With the rise of popularity of cloud services, the number of cloud applications has also increased considerably. These applications are generally composed of multiple inter-communicating VMs which may exchange a huge amount of data between each other. Examples of traffic-intensive applications are scientific applications,

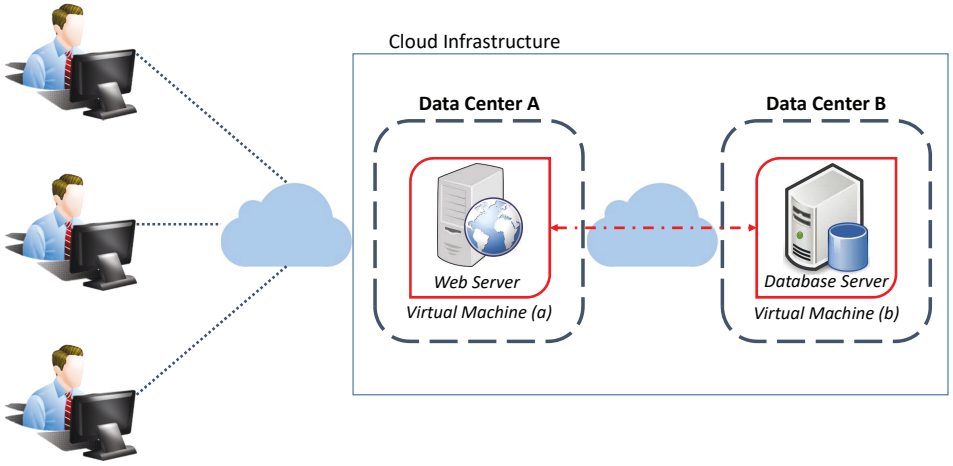


Figure 2.5: Example of distributed application.

video streaming servers, search engines and web browsers. Communication traffic between VMs is considered as one of the dominating costs for communication intensive distributed applications [13]. An example of data-intensive distributed application is shown in Figure (2.5). Web applications are generally distributed among different VMs. Web server running in one VM may need to communicate with a database server running in another distant VM in order to satisfy client's requests. In such a context, it is important to ensure several application performance metrics such as response time, round trip time (RTT), latency, etc. Thus, some DCs need to be placed in proximity of end-users to provide better user's experience [32]. In [42], the authors have classified cloud application workloads into three categories:

Data-Intensive Workload: Such workloads may cause huge data transfer, however, they require less computational resources. As an example, we can think of a video streaming application where each user request generates a new video streaming process. For this type of application, it is important to maintain the application performance and prevent from bottlenecks in the network. Hence, it is crucial to take placement decisions according to network-status and levels of congestion of communication links.

Computationally Intensive Workloads. This category represents the High Performance Computing (HPC) applications that are used to solve complex and advanced problems. These applications require high amounts of computing capacity, however, it causes an insignificant data transfer over the communication links. For this category, we can use consolidation techniques in order to reduce the number of active servers which will help in reducing the energy consumption of the DC. VM consolidation technique tries to place VMs within the same host in order to reduce the number of active hosts.

Balanced Workload: This category includes applications that require both computing and data transfer among VMs. An example of such applications is Geographic Information Systems (GISs) which need to transfer huge amounts of graphical data and at the same time, need huge computing resources to process these data.

Cloud DCs host heterogeneous applications which may produce different communication traffic patterns. As shown in [6], there are predominant types of inter-VM network traffic that can be found in the literature [43, 17, 44].

- **Stable Inter-VM communication traffic:** At large timescale, the authors of [43] have demonstrated that for a large proportion of VMs, the traffic rates are stable despite the divergence of the average rate among VMs. Hence, it can be concluded that the communication patterns among VMs can be estimated and can be considered as known a priori to the users. For these types of applications, deterministic optimization models can be applied.
- **Highly non-uniform communication traffic:** In [17], the authors have reported, based on runtime measurements study, that the VMs generate uneven traffic volumes. The study shows that inter-VM traffic rate varies significantly and it is very bursty. Thus, it is hard to have an accurate estimation of the inter-VM traffic. For this category, stochastic optimization models are the most suitable.

In this thesis, we mainly focus on *Data-Intensive* distributed applications. We propose placement and scheduling policies that minimize the communication traffic between DCs. Furthermore, we study both types of traffic patterns (i.e. stable and dynamic) and we propose placement and scheduling policies dealing with each type of traffic pattern.

In the next section, we introduce VM migration techniques in cloud environment.

2.2.4 Virtual Machine Migration

VM migration is the process of dynamically moving a virtual machine from one physical machine to another. The destination host can be within the same DC or in a distant one. The VM migration is a management technique that has many benefits. In particular, it gives DC managers the ability to adapt the placement of the different VMs in order to optimize their infrastructure, to better satisfy performance objectives, improve resource utilization and communication locality, achieve fault tolerance, reduce energy consumption, and facilitate system maintenance activities [2].

Local-Area Network Migration

There are mainly two types of VM migration techniques. The simplest one is called *Non-Live Migration* (cold migration). This technique consists in suspending and resuming the execution of VMs before and after the migration process, respectively. However, this type of migration has not been widely used due to long VM downtime during the migration process.

The second type of VM migration is called *Live-VM migration*. It is the most common type of VM migration where the VM is maintained available during the migration process. The goal of this type of VM migration is to reduce as much as possible the total transfer time. There are mainly two approaches for live migration.

- *Pre-Copy Migration:* Memory contents are copied while the VM is still running. However, the memory content can be changed during the transfer process, thus, the changed contents are iteratively copied to the destination. The

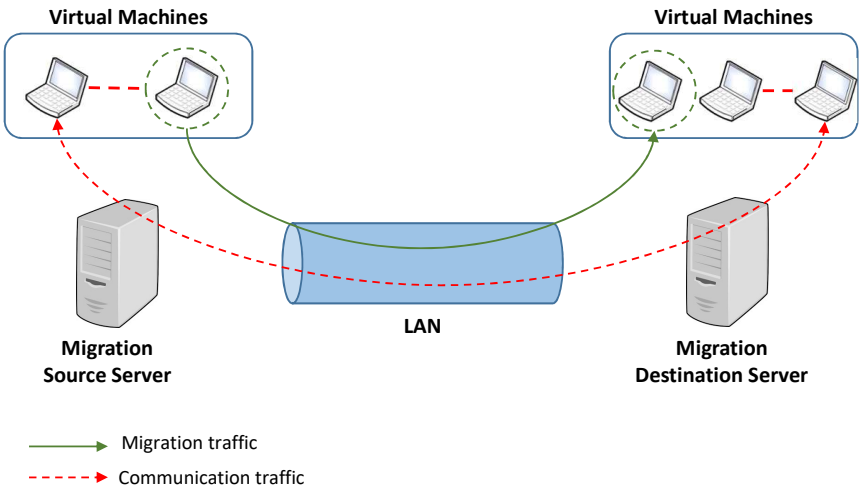


Figure 2.6: Local-Area Network VM migration.

process continues until either the number of remaining pages is small, or a fixed threshold is reached. In such cases, the VM is suspended allowing the remaining pages to be copied. Then, the VM will resume its execution in the destination and it will be destroyed at the source.

- *Post-Copy Migration:* In this approach, the memory content is transferred after transferring the process state. First, the process states are copied to the destination which allows the VM to resume quickly. Then, VM's memory contents are fetched from source to target. All access to memory contents that have yet to be migrated are trapped by memory faults, causing the missing content to be fetched from source machine.

Since the migrated VM remains running during the live migration process, it still communicate with other VMs (in source or/and destination). We refer to this traffic by communication traffic. In addition to communication traffic, there is also traffic generated during the migration process which is composed of the data transferred during the migration of the VM as illustrated by the Figure (2.6).

Wide-Area Network Migration

Most of the existing migration technologies focus on Local-Area Network (LAN) migration. In fact, migration of VMs over a LAN is relatively simple since DC LANs are provisioned using high-speed low-latency links [7].

In contrast to LAN VM migration, Wide-Area Network (WAN) VM migration requires the transfer of the disk image in addition to CPU and memory states [3]. Moreover, WAN links interconnecting DCs are bandwidth-constrained and the network connection are less stable in WANs. In addition, inter-DCs latencies are more important than in LAN environment. In such an environment, it may be impossible to dedicate a certain amount of bandwidth capacity to transfer one VM from one DC to another, especially when the disk is also transferred among WAN links.

Hence, it is important to ensure the reliability during the migration while at the same time minimizing the bandwidth usage and optimizing the data transfer

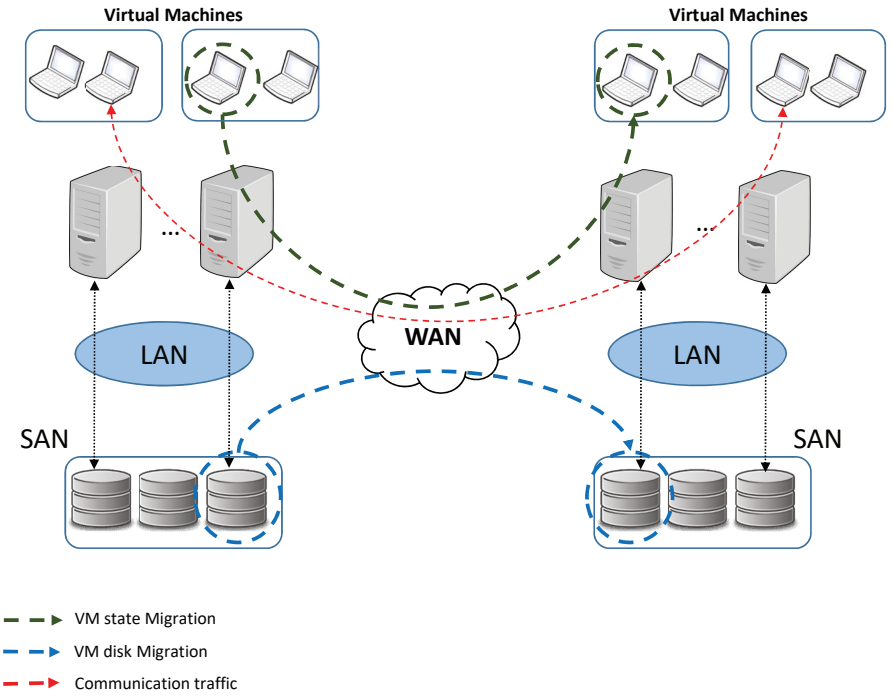


Figure 2.7: Wide-Area Network VM Migration.

in order to reduce the migration costs [45]. Figure (2.7) represents an example of WAN VM migration.

2.2.5 Autonomic Computing

Autonomic computing [46] was first introduced by IBM in 2001 as a vision of computing environments which can automatically observe and adapt themselves according to high-level objectives. The driving motivation behind the autonomic computing initiative was the fact that, the complexity of today's large-scale distributed systems makes it hard to develop, deploy, configure, and maintain them.

The main characteristic of any autonomic system is self-management [47]. Self-management is the ability of a system to automatically adapt to changes that appear in its environment without needing human interaction. In this context, Autonomic managers (AM) [46] are software agents which implement self-management properties of the autonomic computing system. An AM must be able to collect and store monitoring information. Once gathered, monitoring information is stored in a knowledge base. It is then analyzed in order to decide whether actions need to be taken or not. In case actions need to be taken a plan must be created, which will generate a set of desired changes. Finally, the plan must be executed. Figure (2.8) presents an example of an autonomic manager.

In a self-managing autonomic environment, system components are characterized by embedded control loop functionality (or attributes) [48]. These functionalities are divided into four main categories: self-configuration, self-healing, self-optimization and self-protect.

Self-configuration: Self-configuring components are able to dynamically adapt to the changes in the environment that can include the deployment or removal of

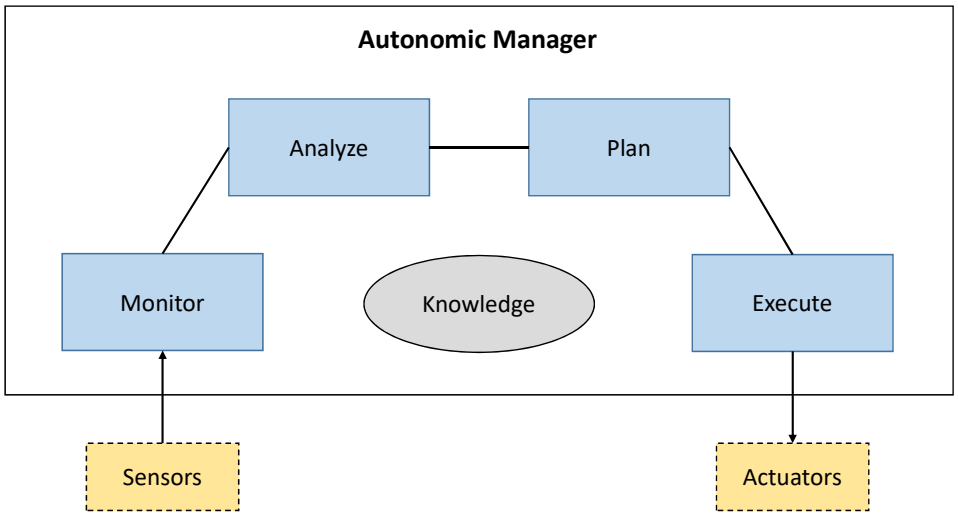


Figure 2.8: Autonomic Manager.

components, changes in the system characteristics, etc. The dynamic adaptation is based on policies that are provided by the IT manager. This property ensures flexibility of the system and allows productivity and business growth.

Self-healing: Self-healing components are able to detect system failure, diagnose the problem and propose a corrective action based on policies without disrupting the IT environment. Hence, the system becomes more resilient and produces less failures.

Self-optimization: Self-optimizing components are able to monitor and tune resources automatically in order to meet end-users or business needs. Some tuning actions could be the migration/reallocation of certain resources in order to improve for example energy utilization or ensure deadline constraints. Self-optimizing components ensure the elasticity of the system and optimize the resource utilization over time.

Self-protection: Self-protecting components are able to anticipate and detect hostile behaviors. As a response to such behavior, they take corrective actions to make themselves less vulnerable. Hence, the system security will be consistently reinforced with new privacy policies.

In this thesis, we propose an autonomic DC management tool based on optimization models aiming at providing periodically optimized plans which include placement, migration and scheduling decisions in order to satisfy the objective of minimizing the traffic volume between DCs.

2.2.6 Optimization Problems

Optimization is a sophisticated tool which is able to help decision makers solve complex problems that arise when having limited resources and under different constraints. As a matter of fact, modeling consists of elaborating a simplified representation that can solve a given problem. Optimization modeling consists of identifying

the objective function, the design (or decision) variables and the constraints for the problem [49]. However, the choice between the different representations has a huge influence on the effectiveness of the obtained solution. Optimization brings several benefits. In fact, it permits to discover unknown approaches and find the best ones under several constraints. Moreover, using optimization, the decisions will be automated and could be validated by exploring more scenarios and testing new alternatives.

Classification of Optimization Problems

Optimization problems can be classified based on different criteria. For example, based on the nature of equations (i.e. objective function and constraints), optimization problems can be categorized into four main categories: linear, nonlinear, geometric and quadratic [50].

- **Linear Programming Problem (LP):** This type of problem is the most used type of constrained optimization model. The objective function and all the constraints must be linear functions of the design variables. LP problems can be also classified as *Integer Linear Programming* (ILP) problems, where all decision variables are integer, and *Mixed Integer Linear Programming* (MILP) problems, where some, but not all decision variables are integer. Integer optimization problems concern mainly problems of efficient allocation of limited resources that need to meet a desired objective, in particular, when the resources can only be divided into discrete parts. To model optimization problems with discrete decisions, a common approach is to formulate the problem as mixed integer optimization [51].

- **Nonlinear Programming Problem (NLP):** This category involves problems that have nonlinear functions among the objective and the constraints.

- **Geometric Programming Problem (GP):** If the objective function and the constraints are expressed as polynomials, the problem is called geometric.

- **Quadratic Programming Problem (QP):** This type of problem has a quadratic objective function and linear constraints. For maximization problems, the objective function is concave. It can be solved by adapting the linear programming techniques.

We can also classify optimization problems based on deterministic nature of the variables. Hence, optimization problems can be categorized as *deterministic* and *stochastic* programming problems [50].

- **Deterministic Programming Problem:** In this class of problems, all the variables are considered as deterministic. In a deterministic system, for the same input, the system will produce the same output.

- **Stochastic Programming Problem:** In this type of problems, some or all of the parameters are considered as random variables (non-deterministic or stochastic) and are expressed probabilistically. A stochastic variable is a random variable that evolves in time. A stochastic model takes into account the element of risk and it is more difficult to formulate and solve efficiently.

Depending on the nature of equations involved in the problem, a stochastic optimization problem is called a stochastic linear, geometric, dynamic, or nonlinear programming problem.

Solving Optimization Problems

In the literature, there are several methods for solving different types of optimization problems efficiently. The optimum methods known as *mathematical programming techniques* provide best or optimal solution to a given problem [50]. Solving Integer optimization problems is a very difficult task. Unlike continuous linear optimization problems, the feasible regions of integer optimization problems consists of a discrete set of points. In particular, for MILP, the feasible region is a set of disjoint polyhedra [51]. Finding global optima for integer optimization problems requires to prove that a particular solution dominates all others. To cope with these difficulties, one approach is to find a valid upper bound, a relaxation or valid inequality (i.e. cutting plane). The simplest approach to solve integer optimization problems is to enumerate all possible outcomes. However, this can lead to a combinatorial explosion due to the exponential number of variables. A more efficient solving approach is to eliminate some solutions using feasibility or domination rules. This methods is called *branch and bound* and is commonly used to solve integer optimization problems efficiently.

In this thesis, we have used some of the above mentioned optimization techniques. In particular, we have used *MILP* to model and solve the deterministic problem and *Stochastic integer programming* (SIP) to solve stochastic problems with uncertainty. Using a linear solver, the MILP models can provide optimal and exact solutions.

As discussed above, MILP are generally known as *NP-hard* problems [52] (i.e. computational complexity), however, some particular models can be solved within a reasonable period of time. There are several formulation enhancement methods used to cope with the complexity of such problems. In this work, we have used some well-known techniques, namely, *valid inequalities* [53] and *variable aggregation* [54].

Valid inequalities are additional constraints to the linear program that improve tightness of relaxation and combine constraints in order to eliminate non-integer solutions. As for the *variable aggregation* technique, it aims at reducing the number of variables in the formulation.

On the other hand, to solve SIP problems, one common method is to use scenario-based approach that consists in the enumeration of all possible outcomes and solve the problem as an ILP. However, when the number of scenario is huge, this method becomes inefficient. An alternative method is to apply sampling-based methods which have been successfully used in many different fields of stochastic optimization [55].

Many of the existing traditional optimization techniques applied to real world problems suffer from many issues preventing them from determining a solution within a reasonable amount of time. This is due to several reasons such as the huge number of variables, difficulties of the constraints, symmetry of the formulation, etc [56].

To cope with these problems, alternative methods were proposed. Among these methods, we cite decomposition methods, such as Dantzig-Wolfe or Benders, heuristic and metaheuristics. In particular, *heuristics* and *metaheuristics*, are able to provide approximate solutions. In contrast to exact methods, (meta)heuristics are

generally simple to design and implement. A heuristic is often used to provide better computational performance. However, the optimality of the obtained solution cannot be guaranteed and has to be validated using experiments and simulations [57]. Metaheuristic is a class of algorithms, which is able to solve complex optimization problems using a number of conventional heuristics. The main advantage of metaheuristic is the fact that it does not require any knowledge about the optimization problem to be used [58].

Complexity of Optimization Problems

According to [59], there are mainly two classes of optimization problems: P and NP. The class P includes all polynomial-time solvable decision problems. As forNP it defines the class of all non-deterministic polynomial-time solvable decision problems.

Definition 1. A decision problem P_i is *NP-Hard* if, every problem in NP is polynomial-time reducible to P_i.

Definition 2. A decision problem P_i is *NP-Complete* if, it is *NP-Hard* and it is also in the class NP.

In fact, *NP-hard* problems are generally very complex and resource consuming. Thus, approximation algorithms are often used to help decision-makers obtain a feasible solution.

Definition 3. Given an optimization problem O, an algorithm A is an approximation algorithm for O if, for any given instance, it returns an approximate feasible solution.

Although approximate algorithms provide feasible solutions, they are not optimal ones. Therefore, it is important to investigate the quality of approximate solution which commonly expressed by the relative error and the optimality gap.

Definition 4. Given an optimization problem O, for any given instance i of O and for any feasible solution s of i, the relative error is defined as follows.

$$E(i,s) = \frac{|m^*(i) - m(i,s)|}{max\{m^*(i), m(i,s)\}} \tag{2.1}$$

Where $m^*(i)$ the optimal solution with respect to the instance i. For both maximization and minimization problems, if the relative error is equal to 0 then, the obtained solution is optimal and it becomes close to 1 when the obtained solution is very poor.

Definition 5. We denote by G, the optimality gap of an approximate solution. It is expressed in % and defined as follows.

$$G = E(i,s) \times 100 \tag{2.2}$$

It is equal to 0% if the approximate solution is optimal. It becomes close to 100% when the obtained solution is very poor.

One of the *NP-hard* problems that we are studying in this thesis is the *Hub Location* problem which is a fundamental building block for the placement problems that arise in Cloud Computing. It is an application of MILP models and network flow models. Therefore, we define in the next section, this problem and we provide some existent works dealing with it.

The Hub Location Problem

The main functionality of a traffic network is to establish the flow from a set of source nodes to a set of destination nodes with minimum costs [60]. Let us consider the complete graph $G = (N, V)$ where N is the set of all nodes and E is the set of edges. Suppose that the flow d_{ij}, $(i \in N, j \in N)$ needs to be sent from the source node i to the destination j. One solution is to connect the node i and j directly. However, this solution is costly and inefficient as it requires that each pair of nodes will be connected together. Another solution for the problem is to select intermediary nodes, called *hub nodes*, which will consolidate the traffic and redistribute it providing thus, an efficient routing of the flow within the network.

The *hubs* are nodes that receive the traffic from different sources and redirect it to destination nodes or to other *hub* nodes. Using intermediary nodes will help to consolidate the traffic in the hubs and minimize the total cost.

Hub location problem have many applications. It is mostly used in the telecommunication and transport fields. There are two version of the Hub location problem namely, single and multiple allocation. In the single allocation problem, a simple node can be connected to only one hub node. On the other hand, for the multiple allocation version, one simple node can be connected to several hub nodes.

Several works dealing with the *Hub location* problem were undertake. The most known formulation of the problem was proposed by Campbel in [61]. However, the proposed formulation is hard to solve for medium and large size problems. The number of variable of the proposed formulation is $O|N^4|$. The main reason for the complexity of the formulation is its symmetry. A formulation is called symmetric if it generates several solutions having the same objective function. Therefore, the *Branch and Bound* algorithm becomes inefficient. In this case, the formulation will induce to a resource saturation before reaching the optimal solution [62].

Besides the classical formulation of the hub location problem, it can be combined with the multi-commodity flow problem. A multi-commodity problem is the generalization of the single-commodity problem where multiple demand flows exist between different source and destination nodes within the same network. There are many representation of the multi-commodity flow problem namely, (1) *Path* formulation, which have an exponential number of variables, (2) *Node-Arc* formulation, which is very hard to solve due to its symmetry, (3) *Overflow variable* formulation, is a very compact formulation that is not suitable for large problem, and (4) *Flow aggregation* formulation which we will use in the next chapters as it has been shown as the most effective formulation for multi-commodity problem [63].

In the next section, we will review relevant work dealing with the VM placement problem.

2.3 State of the Art

Virtual Machine Placement is the process of selecting the most suitable host for a VM. The host can be a physical host within the same DC or in a distant one. The VM placement problem has been studied from different perspectives. As shown in Figure (2.9), the authors in [64] have classified the different VM placement approaches into two categories: power based and application QoS based approach. Each approach is divided into dynamic/online and static/offline placement.

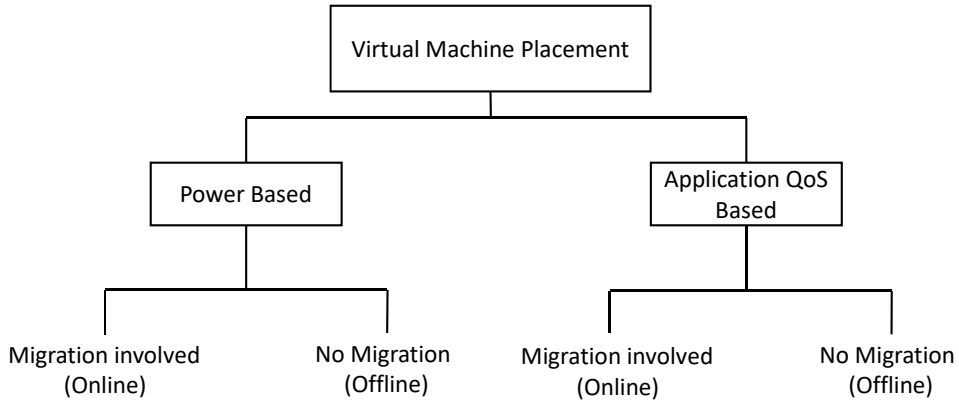


Figure 2.9: Classification of VM Placement Approaches.

In online VM placement, VM migration is considered. In this category, the placement and migration decisions are made during the runtime of the DCs where there are new coming of consumer requests. On the other hand, offline placement approach indicates generally the initial VM placement plan that will be running on the different DCs.

The main difference between these two placement approaches is the fact that online placement will require potential VM live migrations which will produce an additional network traffic. This may affect the performance of the hosted applications as well as the migrated ones.

In the next sections, we review works related to offline, online and stochastic VM placement problem in cloud systems.

2.3.1 Offline VM Placement Problem

In a static/offline VM placement [65], no system reconfiguration is considered and all future demands are supposed to be known in advance. This problem can be seen as the *Initial* VM placement problem, where we consider that all VMs will be placed for the first time in the cloud system.

In [66], the authors presented a green resource management framework for embedding virtual data centers across geographically distributed data centers. Their aim was to maximize the cloud provider's profit. A MILP formulation was proposed in [67]. It aims at placing VMs in large-scale DCs while minimizing the power consumption. The authors considered both inter- and intra-DC VM placement. In [68], the authors propose algorithms to solve the coupled placement of application storage and computation in modern DCs. In [69], the authors considered the placement problem of VMs that host applications with intense bandwidth requirements. In [70], the authors propose a network-aware VM placement approach that satisfies traffic demands of the VMs in addition to hardware requirements. For that purpose, they present different heuristics to solve this problem.

However, most of the aforementioned works proposed heuristic methods to solve the static VM placement problem. In addition, most of them are not suitable for geographically distributed cloud infrastructure.

2.3.2 Online VM Placement Problem

In a dynamic/online placement [71, 72], VM migration is used in order to cope with the demand fluctuation and the dynamic aspect of traffic patterns.

The problem of VM placement in a Cloud environment has received particular attention in recent years. The VM placement problem within a single DC has been extensively studied in the literature. In particular, managing communication traffic within a DC as it becomes a crucial issue.

In [73], the authors addressed the problem of VM placement while minimizing the communication traffic. They proposed a heuristic algorithm to solve the offline problem and a greedy algorithm to solve the online version of the problem. However, they did not consider the migration cost for the online algorithm. In [43], the authors proposed an approximate algorithm that solves the problem of VM placement with traffic-awareness. However, the online version of the proposed solution consider the re-solving of the offline problem. The VM migration cost is not considered. In [74], the authors proposed an online VM placement algorithm based on the traffic matrix. They aim to aggregate and allocate inter-communicating VMs to close servers in order to reduce the traffic congestion. However, no migration or reallocation cost is considered. In [75], the authors propose an approach for VM placement and migration in order to minimize the data transfer time consumption. The main limitation of the previous works is the fact they do not consider the migration cost of VMs in the proposed dynamic methods.

There are only a few researches that have considered the migration cost in the VM placement decisions with traffic awareness. In [71], the authors proposed an algorithm that aims to improve communication performance by reducing the traffic cost of VMs while decreasing the energy consumption of DCs. In [76], the authors proposed Remedy as a cost estimation model that optimizes the VM placement according to the associated cost of migration modeled by the network traffic generated during migration. Most of the recent works consider the VM placement problem and the migration problem as two separate problems. However, it would be interesting to study the interaction between the initial placement and the migration decisions as well as the impact of each on the other. In fact, an effective VMs initial placement can improve the system performance. In [77], the authors investigated the problem of joint VM placement and migration in DC via a multi-objective function. Their aim was to reduce the energy consumption and the cross network traffic among platforms.

However, the main limitation of the aforementioned works is the fact that the proposed solutions only apply to intra-DC environment while the impact of geographical location of DCs was not considered. Hence, these approaches cannot be applied in a geodistributed cloud infrastructure as in this context, the dynamic placement of VMs involves the migration of both the memory and the disk state of the VM.

Some recent researches have studied the problem of VM placement within geographically distributed DCs. A number of these works tried to reduce power consumption or service delay of geographically distributed DCs by optimizing the location of DCs [78], [79]. In [80], on the other hand, the authors proposed both offline and online solutions based on scheduling techniques to solve the problem of energy efficiency and load balancing for a geographically distributed Cloud infrastructure. Whereas, in [81], the authors presented a framework for dynamic service placement

in geographically distributed clouds. Their approach is based on control and game-theoretic models. They aimed to optimize the hosting cost dynamically according to both demand and resource price fluctuations.

However, most of the existing works focus on minimizing the power consumption, or maximizing resource usage. Our work can be considered as complementary to the existing works as we aim to minimize the amount of traffic on the backbone connecting different DCs. Thus, we prevent from possible congestion problems and we reduce the data transport costs including energy consumption costs.

2.3.3 Stochastic VM Placement Problem

In [82], the authors have proposed a survey of the different optimization techniques used to solve this problem. Among these techniques, there are deterministic integer programming and Stochastic Integer Programming (SIP) [83].

In contrast to deterministic approach, the SIP technique considers uncertain parameters (e.g. future demand). It makes use of estimation models using probability distributions. In a realistic Cloud environment, future demands are unknown. The basic idea used in stochastic programming is to convert the stochastic problem into an equivalent deterministic problem. The resulting deterministic problem is then solved by using familiar techniques such as linear programming [50].

VMs workload is considered bursty according to recent studies [15, 16, 17]. SIP has been used to solve load balancing and capacity planning problems in Cloud. In [84], the authors propose an optimal placement algorithm to provision resources of multiple cloud providers. Their objective was to reduce the cost of hosting the VMs while considering future demand and cost. The proposed algorithm is based on SIP to rent resources from providers. They used two-stage formulation. The first stage defines the number of reserved VMs while the second defines the number of VMs that are allocated in the utilization and on-demand phases. In [15], the authors have proposed a stochastic load balancing scheme which aims to provide probabilistic guarantee against the resource overloading with VMs migration, while minimizing the total migration overhead. However, they address the problem within a single DC and did not consider inter-VM communication traffic while making migration decisions. In [85], the authors have studied the VM consolidation problem with dynamic bandwidth demand. They have formulated the problem as a variant of the stochastic bin-packing problem and they have proposed an approximate algorithm to solve it.

In [86], the authors have considered a stochastic model of a cloud computing cluster, where jobs arrive according to a stochastic process. They have focused only on resource allocation problems, such as the design of algorithms for load balancing among servers, and algorithms for scheduling VM configurations. In [87], the authors have studied the VM placement in DCs with multiple deterministic and stochastic resources. First, they have formulated the Multidimensional Stochastic VM Placement problem, with the objective to minimize the number of required servers and at the same time to satisfy a predefined resource availability guarantee. They have shown that the problem is NP-hard, and have proposed a polynomial time algorithm called Max-Min Multidimensional Stochastic Bin Packing. In [88], the authors have studied two cost minimization problems to address the capacity planning in an IaaS Cloud. They have used simulated annealing, a well-known

Table 2.1: Comparison of related works.

Approaches	Objective	Placement Type	Optimization Technique	Migration cost	DCs topology
Cohen and al. [69]	Communication traffic	Offline	Deterministic	N/A	Centralized
Zhang et al. [73], Meng and al. [43]	Communication traffic	Offline and Online	Deterministic	N/A	Centralized
Dias et al. [74]	Traffic congestion	Online	Deterministic	N/A	Centralized
Vu et al. [71]	Traffic and power	Online	Deterministic	✓	Centralized
Mann et al. [76]	Migration traffic	Online	Deterministic	✓	Centralized
Duong-Ba et al. [77]	Traffic and power	Online	Deterministic	✓	Centralized
Goudarzi et al. [80]	Energy efficiency, load balancing	Offline and Online	Deterministic	✓	Distributed
Amokrane et al. [66]	Provider's profit	Offline	Deterministic	N/A	Distributed
Kantarci et al. [67]	Energy consumption	Offline	Deterministic	N/A	Distributed
Zhang et al. [81]	Hosting costs	Online	Deterministic	✓	Distributed
Chaisiri et al. [84]	Hosting costs	Offline	Stochastic	N/A	Distributed
Yu et al. [15]	Load balancing	Online	Stochastic	✓	Centralized
Chase et al. [89]	Cost of resource provisioning	Offline	Stochastic	N/A	Distributed
Our approach	Inter-VMs Communication traffic	Offline and Online	Deterministic and Stochastic	✓	Distributed

randomized search algorithm, to solve these optimization problems.

In [90], the authors studied joint delay sensitive jobs (SENs) and delay tolerant (TOLs) jobs. Their goal was to minimize total costs while guaranteeing QoS for delay sensitive jobs and achieving a desirable delay performance to delay tolerant jobs. They have proposed queue-based scheme for joint server provisioning, SEN dispatching, TOL load shifting and capacity allocation in geo-distributed internet DCs.

In [89], the authors have proposed a joint approach that combines VMs and bandwidth allocation. They have used stochastic programming to take into account the uncertainty of demand. They proposed multi-stage SIP formulation to solve the problem. To improve the efficiency of the stochastic optimization formulation, they have reduced the problem space with scenario tree reduction. However, the authors did not consider VM migration problem. Furthermore, they did not consider inter-VMs communication while making the placement decisions.

To the best of our knowledge, this work is the first effort addressing joint network-aware VM placement and migration problem within geographically distributed DCs with the objective of minimizing the backbone traffic (i.e. inter-DCs traffic). Table (2.1) highlights and summarizes our contributions compared to the most relevant existent works.

The present work is different from traditional VM placement proposals since it considers exact methods that provide optimal placement and migration scheme. In addition, it uses both offline and online approaches to solve the VM placement problem.

Furthermore, we study the VM scheduling problem within a geo-distributed cloud infrastructure. In particular, we propose heuristic methods that provide the best inter-DCs migration sequence of inter-communicating VMs with the objective of reducing the overall traffic during the migration process. We also study the effect

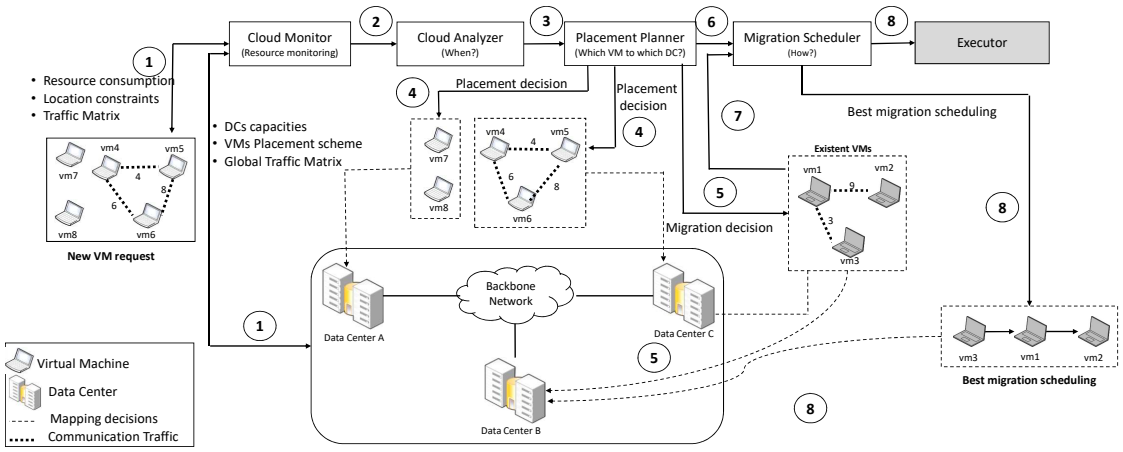


Figure 2.10: System Model.

of VM's execution period on the migration decisions and we show its impact on the stability of the cloud system.

Finally, in order to evaluate the proposed optimization approaches, we have used simulation-based environment as well as experimental tests. We have used Amazon EC2 hardware metrics values and we have generated the traffic matrix, representing the data exchanged between each pair of VMs, randomly. In fact, as it was argued by [91], [17], it is typically hard to obtain such data from real DCs because of the required server level instruments.

In the next section, we present an overview of the system model used throughout this thesis. We enumerate also the different assumptions that we have considered. Then, we illustrate the different components of the autonomic DC management tool proposed.

2.4 Overview

Throughout this thesis, we consider an IaaS environment represented by geographically distributed DCs that are interconnected through a backbone network as shown in Figure (2.10). The different DCs are under the management of the same Service Provider (SP). The backbone network is owned and managed by the Infrastructure Providers (IPs). SPs are charged based on the total network Input/Output of data transferred through the backbone links (i.e. from and to cloud servers) [7].

In this work, we focus only on placement problems. Routing and network design are out of our scope. In such an environment, the cloud provider has a priori, no knowledge about the VMs' demand and the fluctuation of the traffic matrix. The traffic matrix represents communication traffic or bandwidth requirements between each pair of VMs.

In this work, we make the following assumptions:

- The entire infrastructure is owned and managed by the same IaaS provider.
- Each VM is characterized by its hardware configuration in terms of CPU, RAM and Storage.

- Each DC is characterized by its capacity in terms of hardware resources CPU, RAM, and Storage.
- Time is divided into slots [1..T].
- The metrics characterizing the DCs are assumed to be constant during each time slot and are measured at the beginning of each time slot.
- Each VM may have a location constraint. Thus, it can only be placed in a defined set of DCs.
- There are multiple independent clients submitting requests to provision VMs that may be heterogeneous and may have both dynamic traffic and location matrices.

Efficient cloud DCs management has become a very complex task, especially for geographically distributed DCs. In this context, the cloud manager needs to take several crucial decisions: (1) Where to place each VM while ensuring the proximity location constraint and minimizing the backbone traffic? (2)When a system reconfiguration is needed? (3) Which VM needs to be migrated and to which DC? (4) If a set of inter-communicating VMs need to be migrated, what is the best VMs migration scheduling that minimizes the overall traffic circulating in the backbone network? (5) How to make placement and migration decisions such that the risk of network overloading in the future is minimized?

In order to make the optimal decisions to answer the above questions, we propose a DC management tool based on an autonomic system. Autonomic computing systems are capable of self-managing themselves by doing self-configuration and self-optimization [14]. Such a system must be able to analyze itself at runtime, determine its state and determine a desired state that maintains the QoS.

As shown in the Figure (2.10), the system is divided into the following modules:

- *The Monitor Module:* It is responsible for (1) the collection of information relative to the resource consumption of different VMs, the traffic matrix, historical data collect, the location constraint matrix and the DCs' hardware capacities.
- *The Analyzer Module:* It is responsible for (2) analyzing the data collected by *The Monitor Module*. This includes the analyze of the historical data collected by the *Monitor* and the identification of the traffic distribution patterns.
- *The Placement Planner Module:* It is responsible for making placement and migration decisions based on the information sent by *The Analyzer Module* (3). This module is based on optimization models that aim to find the optimal placement (4) and/or migration (5) plan for VMs while minimizing the expected backbone traffic. It is composed of two sub-components: *Initial Placement Planner*, which is based on offline optimization programs that aim to place for the first time the different VMs in the system, and the *Dynamic Placement Planner*, which is responsible for making both deterministic and proactive placement and migration decisions.

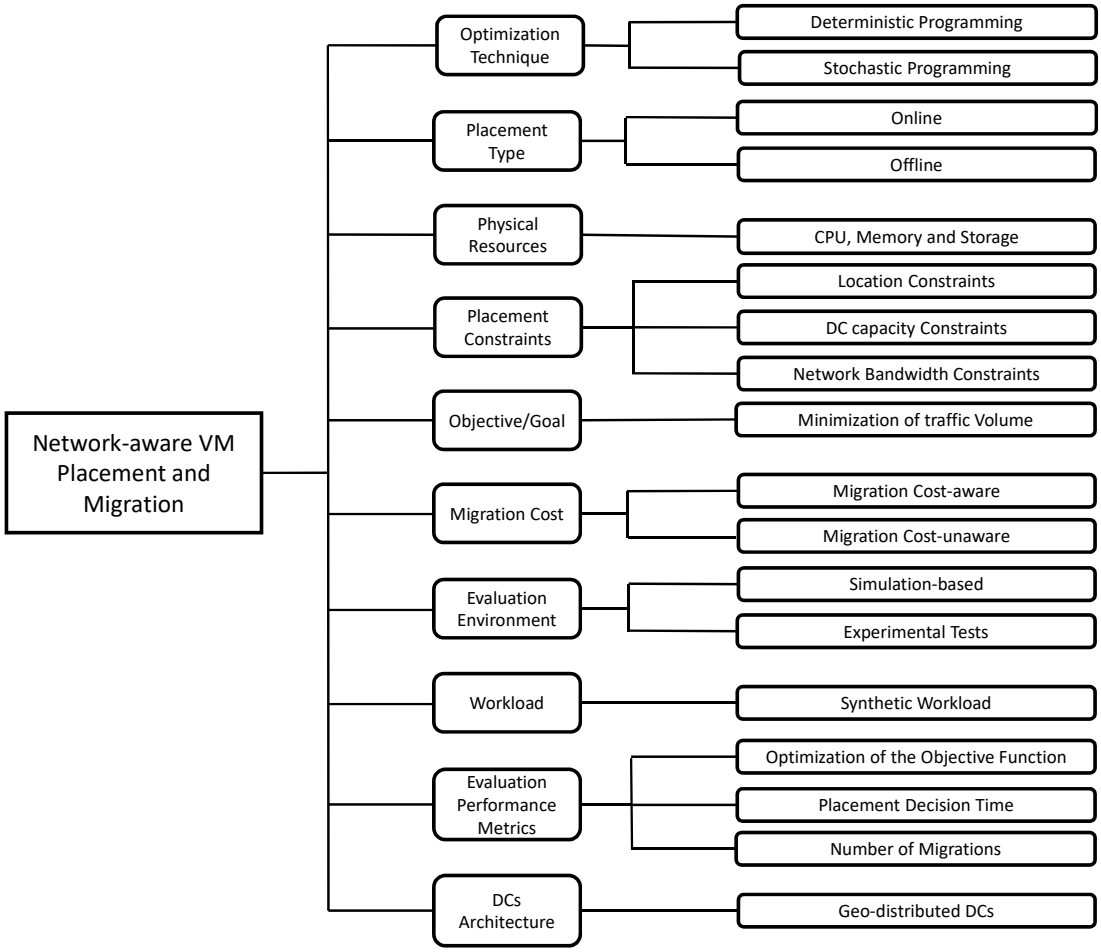


Figure 2.11: Thesis Overview.

- *The Migration Scheduler Module:* This module is based on heuristics. The aim is to find the best inter-DCs migration scheduling for inter-communicating VMs (7). The migration decision is provided by the *Planner Module* (6) and validated/scheduled by the *Migration Scheduler*. In particular, this module considers VMs having deadline and finite lifetime constraints.

- *The Executor Module:* It is responsible for (8) the execution of the placement and/or migration decisions made by the *Placement Planner Module* and the *Migration Scheduler Module*.

In this work, we focus only on the *Placement Planner* and the *Migration Scheduler* modules. The reconfiguration of the system is triggered periodically. At the beginning of each period, the planner decides if the reconfiguration of the system will generate a profit or not. If a reconfiguration is needed, the planner provides the new placement scheme of the different VMs and the list of the VMs that will be migrated. Then, the migration list is sent to the migration scheduler. The latest provides the optimal migration sequence and sends it to the executor that will perform the actual migration.

Figure (2.11) illustrates an overview of the work presented in this thesis. As discussed in Section 2.2.3, we consider two types of inter-VM communication traffic

patterns: *stable* and *Highly non-uniform* traffic. To deal with these two types of traffic, we propose deterministic programming models for applications with stable communication traffic and stochastic programming models to deal with applications having non-uniform and uncertain inter-VM traffic.

In addition, we consider that each VM has a location constraint. This constraint restricts the placement of VMs in a certain set of DC known a priori. It aims at maintaining service performance, reducing time-delay by placing high communicating VMs in proximity of end-users, or ensuring availability.

In contrast to existent works, we take into consideration the migration cost of VMs. As we have mentioned in previous section, live WAN migration generates a huge amount of traffic. Hence, this parameter cannot be neglected when making migration decisions. The goal of this work is to provide optimized placement and migration scheme in order to maintain the inter-DCs traffic volume as minimum as possible.

For the experimental study, we have used the commercial solver CPLEX [92] to solve the proposed optimization programs. Because of the required server level instruments and the lack of benchmarks, it is typically hard to obtain data from real DCs [91, 17]. Hence, to validate our approach under realistic conditions and inputs, we have used the well-known simulation toolkit CloudSim [93].

In fact, CPLEX experiments and CloudSim simulations are complementary and necessary for the evaluation of placement and migration policies in Cloud systems. As we propose optimization based algorithms, it is important to evaluate its effectiveness and the quality of the provided solutions using the solver CPLEX. However, cloud systems are very complex and there are many aspects that may affect the placement decisions such as DC network topology, energy consumption, etc. These aspects are not taken into account in the proposed algorithms. Therefore, it is important to verify that our placement and migration policies still give effective plans under different conditions and scenarios. To show the effectiveness of the proposed approach, we have used different evaluation metrics such as, placement decision time, number of migrations, quality of the objective function, etc.

2.5 Conclusion

In this chapter, we have introduced first, the general context of the thesis by giving some background knowledge about cloud computing, cloud DCs architecture, inter-VM communication, VM migration and autonomic computing. Then, we presented a literature review on the VM placement problem. Finally, we gave a summary of the relevant related works and presented the system model used throughout this thesis.

Chapter 3

Offline VM Placement Optimization in Geo-Distributed DCs

Contents

3.1 Introduction

In this chapter, we present the different optimization models that we have proposed to deal with the *Initial* (i.e. static/offline) VM placement problem in geo-distributed DCs. The work presented in this chapter has been published in [94] and [65]. First, we present the first formulation based on a well-known *Hub Location* formulation. Then, we derive a more efficient formulation and enhance it with variable aggregation technique. Finally, we present the results of experiments conducted on the different formulations.

3.2 Problem Description

In the previous Chapter 2, we have presented the context of this work and we have described the system model used throughout this thesis. The DC management tool

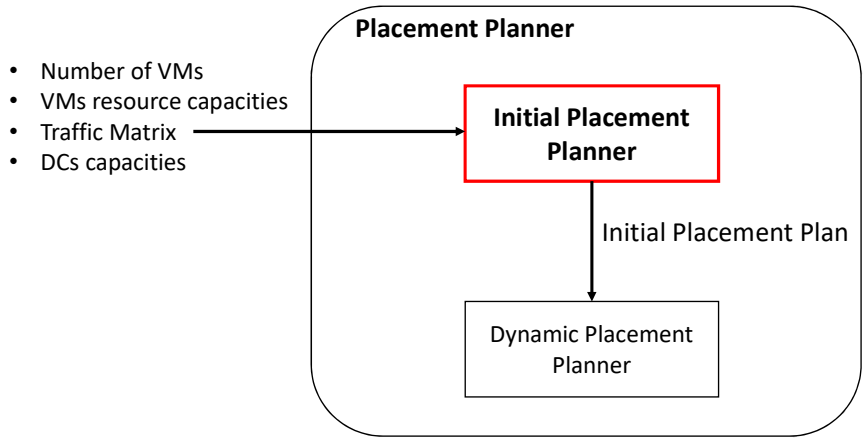


Figure 3.1: Placement Planner Overview.

that we have proposed is based on optimization models that are able to solve the problem efficiently and provide optimal placement scheme.

In this chapter, we focus in particular, on the problem of *initial* VM placement in geo-distributed DCs. As shown in Figure (2.10), the studied system is composed of different modules. Among these modules, there is the *Placement Planner Module* which is responsible for making placement and migration decisions. Due to the complexity of the problem, the *Placement Planner* is divided into two sub-components as shown in Figure (4.1). The *Initial Placement Planner* will be executed when the VMs will be placed for the first time in the system. It will be invoked once and it will provide static placement scheme for different inter-communicating VMs. The placement scheme will be considered as input for the *Dynamic Placement Planner* in order to make migration decisions.

This chapter presents formal optimization models that will be implemented in the *Initial Placement Planner*. We consider that the placement plan will remain the same and there are no VM migration. We refer to this problem by the *Initial VM placement problem* (IVMP). Our objective is to minimize the amount of traffic between the different DCs.

The IVMP can be seen as a variant of the Hub Location problem[95], where DCs are considered as Hub nodes. The problem of *Hub Location* is a class of optimization problem that have been extensively studied in the literature. It has been used to solve many problems such as network design planning in transportation and telecommunication systems [95].

Proposition 1. The IVMP problem is *NP-Hard*.

Proof. The proof is based upon reduction of IVMP to a capacitated multicommodity flow problem by considering DCs as hubs and where the flows are unsplittable, since each demand node must be assigned to a single DC (hub) then to a single path. The capacities on the DC can be considered as capacities on virtual links by splitting the DCs into two connected virtual nodes. The capacity of this virtual link is the same capacity of the DC. This problem is well known as being NP-hard [96]. □

Despite the NP-hardness of the IVMP model, we show using extensive experiments that it can be solved for large problem sizes within a reasonable computational time.

In the next section we present our first attempt to solve the *Initial VM placement problem* (i.e static or offline) in geo-distributed DCs.

3.3 First Proposal

In this section, we present MILP formulations used to solve the IVMP problem in geo-distributed DCs. First, we adapt a well-known *Hub Location* formulation to fit our problem. Then, in order to enhance the execution time of the linear program, we propose a new formulation and add new constraints called *Valid inequalities*. Experiment results show the effectiveness of the strengthening techniques that we have used.

3.3.1 Hub Location Formulation

The VM placement problem in geo-distributed cloud systems can be seen as a variant of the well-known *Hub Location* problem [95], where the DCs are considered as hub nodes. In this work, we try to optimally place VMs among different DCs in order to minimize communication traffic within the backbone traffic. However, most existent *Hub Location* formulations are not suitable for our problem since they consider that all nodes can be hub nodes that is not the case in our problem. In contrast to hub nodes, DCs are considered as intermediary nodes which cannot generate traffic.

The problem is considered as a graph $G = (N, E)$, where N designates the set of all the nodes and E the set of the edges of the graph. We are given a traffic matrix that indicates the amount of communication traffic between each pair of VMs.

In this formulation, we consider the following decision variables:

- y_i^k, takes 1 if the VM $i \in V$ is placed in the DC $k \in D$, 0 otherwise.
- v_{kh}, designates the amount of traffic exchanged between DCs $k \in D$ and $h \in D$.

We denote by *HL* the model described as follows:

$$\min \sum_{k \in D} \sum_{\substack{h \in D \\ h \neq k}} v_{kh} \tag{3.1}$$

Subject to:

$$\sum_{k \in D} a_i^k . y_i^k = 1 \quad \forall i \in V \tag{3.2}$$

$$\sum_{i \in V} \sum_{j \in V} y_i^k . y_j^h . d_{ij} = v_{kh} \qquad \forall (k,h) \in D^2, k \neq h \tag{3.3}$$

$$\sum_{i \in V} y_i^k . u_{ir} \leqslant \sum_{k \in D} cap_r^k \qquad \forall r \in R \forall k \in D \tag{3.4}$$

$$y_i^k \in \{0,1\} \qquad \forall i \in V, \forall k \in D$$

$$v_{kh} \geqslant 0 \qquad \forall k, h \in D$$

The objective function (3.1) aims to minimize the amount of traffic between DCs. This traffic is generated mainly due to the communication between the different VMs. The constraint (3.2) ensures that each VM is running on only one DC

while considering the location matrix which indicates if a VM can be assigned to a certain DC. The constraint (3.3) is a demand satisfaction constraint, it ensures the satisfaction of all traffic demand between different VMs. The constraint (3.4) ensures that the amount of resources (CPU, RAM and storage) consumed by the set of VMs assigned to a DC does not exceed its capacity.

The above model is not linear due to the constraint (3.3). In order to linearize it, we introduce a new binary decision variable x_{ij}^{kh} that takes 1 if the VM $i \in V$ is placed in the DC $k \in D$ and the VM $j \in V, j \neq i$ is placed in the DC $h \in D, h \neq k$.

Hence, the constraint (3.5) must be added to the model.

$$a_i^k.y_i^k + a_j^h y_j^h \leqslant x_{ij}^{kh} + 1 \quad \forall (i,j) \in V^2, \forall (k,h) \in D^2, i \neq j, h \neq k \tag{3.5}$$

The constraint (3.3) becomes:

$$\sum_{i \in V} \sum_{j \in V} x_{ij}^{kh}.\tau_{ij} = v_{kh} \quad \forall (k,h) \in D^2, k \neq h \tag{3.6}$$

This first formulation is proved to be inefficient for medium and large problem sizes. In fact, it is a symmetric formulation and it has a weak lower bound which impacts the quality of the optimal solution and the execution time of the program. Indeed, the (HL) formulation has a huge number of variables $O|N|^4$. The results of different experiments are presented in details in Section 3.3.4.

To cope with these problems, we reformulate in the next section, the problem by considering a multicommodity formulation and by applying aggregation methods [54, 97].

3.3.2 Multicommodity Reformulation

In this section, we present the reformulation of the VM placement problem into a Multicommodity problem [98]. Our formulation is based on flow aggregation. In fact, we aggregate all flows generated by a single source node [99]. In [63] and [100], the authors outlined the computational advantages of this technique. In addition, compared to the first formulation (HL), the number of variables has been reduced to $O|N|^3$ variables.

We consider the problem as a graph denoted by $G = (N, E)$, where N designates the set of all the nodes and E the set of the edges of the graph. We consider that $N = V \cup D$ and $E = L \cup C$, where L designates the set of virtual links that connects VMs and DCs. We denote by C the set of links that connects different DCs with a complete graph. Each link is defined by a pair of source-destination nodes (i, j) where $i, j \in N$.

- If $(i, h) \in L$, then i is settled as a VM node and h as a DC node.
- If $(h, k) \in C$, then h and k are both DCs nodes.

In practice, there are no physical links connecting VMs to DCs. We have used the concept of virtual links in order to adapt our model to a multicommodity problem. Virtual links translate the assignment of each VM to a particular DC.

In this formulation, we introduce the following decision variables:

- f_{hk}^i, designates the amount of traffic originated from the VM $i \in V$ and circulating on the directed link between the DCs $h \in D$ and $k \in D$.

- φ^i_{jh}, designates the amount of traffic originated from the VM $i \in V$ and circulating on the virtual link between VM $j \in V$ and DC $h \in D$.
- σ_{ik}, is a binary variable that takes 1 if a VM $i \in V$ is assigned to a DC $k \in D$, 0 otherwise.
- α_{kh}, designates the amount of traffic exchanged between two nodes k and h. If $k \in D$ and $h \in D$, then $\alpha_{kh} = 0$. Otherwise, if $i \in V$ and $k \in D$, then $\alpha_{ik} = 0$.

The new linear model denoted by MF_w is described as follows:

$$\min \sum_{i \in V} \sum_{k \in D} \sum_{\substack{h \in D \\ h \neq k}} f^i_{kh} \tag{3.7}$$

Subject to:

$$\sum_{h \in D} \varphi^i_{ih} = \sum_{j \in V} \alpha_{ij} \qquad \forall i \in V \tag{3.8}$$

$$\sum_{h \in D} \varphi^i_{jh} - \sum_{h \in D} \varphi^i_{hj} = -\alpha_{ij} \qquad \forall j \neq i \in V^2 \tag{3.9}$$

$$\sum_{k \in D} f^i_{hk} - \sum_{k \in D} f^i_{kh} + \sum_{j \in V} \varphi^h_{ij} - \sum_{j \in V} \varphi^j_{ih} = 0 \qquad \forall i \in V, \forall h \in D \tag{3.10}$$

$$\sum_{k \in D} \sigma_{ik} = 1 \qquad \forall i \in V \tag{3.11}$$

$$\sigma_{ih} \leqslant a^h_i \qquad \forall i \in V, \forall h \in D \tag{3.12}$$

$$\sum_{i \in V} \varphi^i_{hj} = \sigma_{jh} . \sum_{i \in V} \alpha_{ij} \qquad \forall j \in V, \forall h \in D \tag{3.13}$$

$$\varphi^i_{ih} = \sigma_{ih} . \sum_{j \in V} \alpha_{ij} \qquad \forall i \in V, \forall h \in D \tag{3.14}$$

$$\sum_{i \in V} u_{ir} . \sigma_{ih} \leqslant cap^h_r \qquad \forall r \in R, \forall h \in D \tag{3.15}$$

$$f^i_{kh} \geqslant 0 \qquad \forall k \in D, \forall h \in D, \forall i \in V$$

$$\varphi^i_{jh} \geqslant 0 \qquad \forall i \in V, j \in V, h \in D$$

$$\alpha_{hk} \geqslant 0 \qquad \forall h \in N, k \in N$$

$$\sigma_{ik} \in \{0, 1\} \qquad \forall k \in N, \forall i \in N$$

The objective function (3.7) aims to minimize the amount of traffic between different DCs. The constraint (3.8), ensures that all the traffic originated from a VM $i \in V$ and circulating between all DCs is equal to the amount of traffic exchanged between i and other VMs. As for the constraints (3.9) and (3.10), it ensure the flow conservation; these constraints are only applied for the aggregated flows generated by a source node regardless of the destination. The constraint (3.11) ensures that each VM is running on only one DC. The constraint (3.12) is a location constraint, it restricts the placement of a VM in a particular set of DCs defined in the matrix a^h_i. The constraint (3.13) imposes that the amount of traffic, circulating on the virtual link between the VM $j \in V$ and the DC $h \in D$ which is originated from the VM $i \in V$, is equal to the amount of traffic exchanged between these two VMs (i and

j) if VM j is assigned to the DC h. The constraint (3.14) ensures that the amount of traffic originated from VM i and circulating between i and DC h is equal to the amount of traffic exchanged between i and other VMs if i is assigned to the DC h. The constraint (3.15) ensures that the set of VMs placed in a given DC does not exceed its resource capacities in terms of CPU, RAM and Storage.

Although the MF_w formulation is better than the (HL) formulation regarding the number of variables, it presents also many difficulties while trying to solve it for large problem sizes. Thus, we propose to introduce additional constraints that will tighten the linear relaxation of the previous formulation and reduce the search space of the feasible region. This technique, the so called *formulation strengthening*, is widely used in order to speed up the execution time of compact formulations [101].

3.3.3 Valid Inequalities

In this section, we present different valid inequalities, that we have added to strengthen the MF formulation. Valid inequalities are redundant logical constraints that have shown their efficiency during the testing phase [53]. The idea stems from adding additional constraints to the linear program to improve tightness of relaxation and to combine constraints to eliminate non-integer solutions. The guiding line for devising these inequalities is to bind the values of the decision variables of the objective function f^i_{kh} with supplementary valid constraints. This will provide better bounds for the Branch-and-Bound algorithm applied on the binary σ_{ik} variables. We denote by MF the multicommodity formulation enhanced with the following valid inequalities.

Proposition 2. For any given VM $i \in V$ and a DC $k \in D$, the inequality (3.16) is valid for *(MF)*.

$$\sum_{h \in D} \varphi^i_{ih} \geqslant \sum_{h \in D} f^i_{kh} \qquad \forall k \in D, \forall i \in V \tag{3.16}$$

Proof. This inequality stems from the flow conservation constraints. In fact, all the traffic issued by the VM $i \in V$ is bifurcated at a DC $k \in D$: one part goes to other VMs at the same DC k, and the other part traverses the inter-DC links $(k, h) \in D^2$ to reach VMs at other DCs. This can be written as follows:

$$\sum_{h \in D} \varphi^i_{ih} = \sum_{j \in V} \alpha_{ij}.\sigma_{jh} + \sum_{h \in D} f^i_{kh} \quad \forall i \in V \tag{3.17}$$

Thus, (3.16) follows immediately. □

Proposition 3. Let $i \in V$ and $j \in V$ such as $i \neq j$, the following inequality is valid for *(MF)*.

$$\sum_{j \in V} \varphi^i_{kj} \geqslant \varphi^i_{ik} - \sum_{h \in D} f^i_{kh} \quad \forall k \in D, \forall i \in V \tag{3.18}$$

Proof. By writing the flow conservation constraints on the DC $k \in D$, while considering the flow emanating from a VM $i \in V$ as a commodity, we obtain:

$$\varphi^i_{ik} - \sum_{j \in V} \varphi^i_{kj} = \sum_{h \in D} f^i_{hk} - \sum_{h \in D} f^i_{kh} \tag{3.19}$$

Then we get:

$$\varphi^i_{ik} - \sum_{j \in V} \varphi^i_{kj} \leqslant \sum_{h \in D} f^i_{hk} \tag{3.20}$$

After rearranging the terms of the inequality (3.20), we obtain immediately (3.18). □

Proposition 4. Let $l \in L$, the following equation is valid for (MF)

$$\sum_{i \in V} \sum_{k \in D} \varphi_{kj}^{i} = \sum_{i} \alpha_{ij} \quad \forall j \in V \tag{3.21}$$

Proof. This equation stipulates that the total traffic arriving at a VM $j \in V$ is equal to the total demand required by j. Assume that j is connected to a given DC $k \in D$ then. By virtue of the definition of σ_{jk}, we have:

$$\sum_{i \in V} \varphi_{kj}^{i} = \sigma_{jk}. \sum_{i \in V} \alpha_{ij} \tag{3.22}$$

By summing up the two members of (3.22) on all the DCs, we obtain:

$$\sum_{k \in D} \sum_{i \in V} \varphi_{kj}^{i} = \sum_{k \in D} \sigma_{jk}. \sum_{i \in V} \alpha_{ij} \tag{3.23}$$

Since $\sum_{k \in D} \sigma_{jk} = 1$ by virtue of the constraint (3.11), we obtain immediately (3.21). This equality ensures that the amount of traffic arriving to $j \in V$ through the DC $k \in D$ is equal to the total demand of j. In fact, it is another formulation of the demand satisfaction constraint. □

In the next section, we present the different experiment results conducted on the proposed formulations (HL) and (MF).

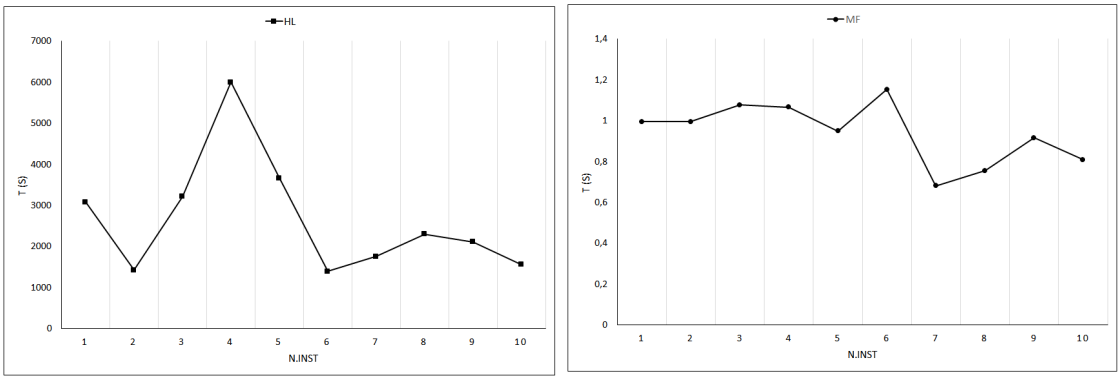
3.3.4 Performance Evaluation

In this section, we present the results of experiments conducted on the optimization models (HL) and (MF) proposed for solving the offline VM placement problem in geo-disributed DCs. We generated instances of different sizes in order to evaluate the execution time of the linear programs and the amount of traffic reduced within the backbone network. We also compared the (HL) and (MF) formulations presented in Sections 3.3.1 and 3.3.2.

The different experiments were carried out on a machine that has an *Intel Xeon* $3,3$ *GHz* CPU and $8GB$ of RAM. We have used the commercial solver CPLEX 12.2 [92] to solve, evaluate and compare different ILP formulations. In all tests, we have considered a general network topology modeling a geographically distributed Cloud architecture.

We assume that VMs have three different instance types (Small, Medium and Large). We have considered the different values of hardware metrics provided by Amazon Elastic Computing Cloud (EC2) [29]. Without loss of generality, we assume that all DCs have the same resource capacities. Each VM can be placed on two possible DCs, known a priori, but effectively, each VM is assigned to only one DC. The traffic matrix represents traffic bandwidth requirements between each pair of VMs. The traffic matrix has been generated randomly. We assume that all VMs can possibly communicate and exchange data with each others. The traffic matrix values vary from 1 to 10 Mbps. All the results are averages in 10 groups of instances generated randomly.

We denote by:



(a) The (HL) Formulation. (b) The (MF) Formulation.

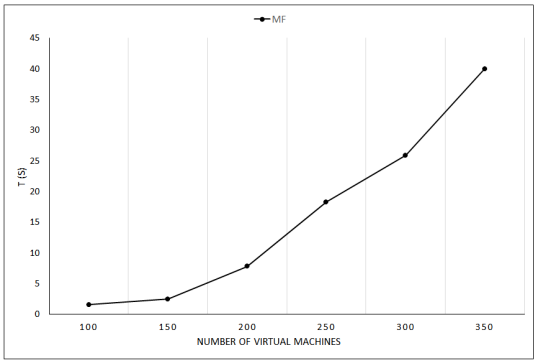
Figure 3.2: Variation of the execution time between MF and HL for $|V| = 60$ and $|D| = 6$.

- HL, the hub location formulation. 3.3.1.
- MF, the enhanced multicommodity formulation presented in Section 3.3.2.
- MF_w, the multicommodity formulation without the valid inequalities.
- S, is the value of the optimal solution provided by CPLEX for (MF) expressed in Mbps.
- S_w, is the value of the best solution provided by CPLEX for (MF_w) expressed in Mbps.
- T, is the execution time in seconds for (MF).
- T_w, is the execution time in seconds for (MF_w).
- G, is the gap (%) between S and the lower bound provided by CPLEX for MF.
- G_w, is the gap (%) between S_w and the lower bound provided by CPLEX for MF_w.

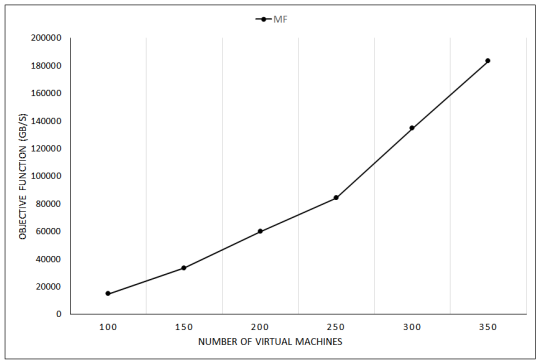
Note that if the gap is equal to zero, it means that the optimal solution is reached. The OM acronym designates an out-of-memory problem of the CPLEX solver.

Comparison of MF and HL

In order to show the effectiveness of the (MF) formulation, we have compared it with the (HL) formulation. The tests were performed using the same set of instances and data inputs for $|V| = 60$ and $|D| = 6$. We plot the execution time for 10 instances generated randomly for both formulations. The experiment results are shown in Figure (3.2a) and (3.2b). Although these two formulations are equivalent and provide the same optimal solution, we observe a huge difference of the execution time between the two formulations. We can conclude that the (HL) formulation is not efficient for practical cases and for large problem sizes compared to the (MF) formulation.



(a) Variation of the execution time with respect to the number of VMs.



(b) Variation of the objective function with respect to the number of VMs.

Figure 3.3: Experiment results performed on MF.

$(\|V\|, \|D\|)$	(MF)			(MF_w)		
	S	G	T	S_w	G_w	T_w
(100, 6)	15057,4	0	1,5738	-	100	OM
(150, 6)	33736,7	0	2,479	-	100	OM
(200, 6)	60037,2	0	7,8266	-	100	OM
(250, 6)	84294,5	0	18,3285	-	100	OM
(300, 6)	134690	0	25,9	-	100	OM
(350, 6)	183481,5	0	40,0093	-	100	OM

Table 3.1: Experiment results for (MF) and (MF_w).

Impact of Valid Inequalities

It is interesting to observe the impact of the addition of valid inequalities presented on Section 3.3.3 on the computational effort. For this particular purpose, we performed a set of computational tests and we compared the (MF) formulation with and without adding valid inequalities. The results are shown in Table (3.1). These results show that the (MF) formulation is more efficient in terms of execution time for large size instances. Moreover, the valid inequalities that we have added to this formulation have proven its effectiveness. In fact, when we compare the formulations (MF) and (MF_w) we note that the (MF_w) formulation presents in many cases an out-of-memory problem when trying to solve it on CPLEX and no optimal solution has been provided. On the other hand, (MF) is able to provide for all instances the optimal solution in a short period of time which, in general, does not exceed one minute.

Then, we have fixed the number of DCs $|D| = 6$ and we have plotted the execution time and the values of the objective function for an increasing number of VMs. The results are depicted in Figures (3.3a) and (3.3b). We remark that the amount of traffic within the backbone network increases when we add more VMs. This is due to the communication traffic between different VMs. In fact, we have considered that all VMs are communicating with each other in order to increase the problem difficulty and test the stability of our model.

Despite the effectiveness of the proposed valid inequalities, the problem remains complex and hard to solve for a number of VMs $|V| \geqslant 400$. Thus, in the next section, we present another efficient formulation to solve the offline VM placement in geo-distributed DCs.

3.4 Second Proposal

In this section, we present our second attempt to solve the static VM placement problem in geo-distributed DCs. Due to the large number of VMs in real cloud environment, the formulations presented in the previous sections cannot be used efficiently. Using extensive experiments, we show the effectiveness of this formulation. This work has been presented in [65]. First, we present the classical formulation of the hub location problem proposed [102] and we adapt it to fit our problem. Then, in order to reduce the execution time of the linear program, we have used variable aggregation technique. Finally, we present experiments results showing the effectiveness of our proposal.

3.4.1 The Classical Formulation

The problem is considered as a complete graph $G = (N, E)$, where N is the set of the nodes constituted by VMs and DCs and E is the set of the edges. By considering a complete graph structure, we aim to estimate the amount of traffic exchanged between each pair of DCs so that we can dimension physical capacity links of the backbone network. We consider that VMs are connected to DCs by virtual links. We are given a traffic matrix that indicates the amount of communication traffic between each pair of VMs. We assume that each VM can be assigned into two possible DCs in order for example to satisfy geographical proximity considerations. But only one DC is effectively assigned to each VM.

We adapt a well-established classical formulation presented in [102] for the single allocation capacitated hub location problem. Although the two problems are quite different, they have many similarities that permit the adaptation of the formulation of [102] to fit our problem. This formulation had been largely used in the literature as being the most efficient formulation for the problem of *Hub Location* [103].

In this formulation, we consider the following decision variables, similar to those of [102].

- z_i^h, takes 1 if the VM $i \in V$ is placed in the DC $h \in D$, 0 otherwise.
- f_{kh}^i, designates the amount of traffic generated by the VM $i \in V$ and circulating between DCs $h \in D$ and $k \in D$.

We denote by O_i, the total flow emanating from a VM i. We have:

$$O_i = \sum_{j \in V} d_{ij} \quad \forall i \in V \tag{3.24}$$

The linear model denoted by (CF) is described as follows:

$$\min \sum_{k \in D} \sum_{\substack{h \in D \\ h \neq k}} \sum_{i \in V} f_{kh}^i \tag{3.25}$$

Subject to:

$$z_i^h.O_i - \sum_{j \in V} d_{ij}.z_j^h = \sum_{k \in D} f_{hk}^i - \sum_{k \in D} f_{kh}^i \quad \forall i \in V, \qquad \forall h \in D \tag{3.26}$$

$$z_i^h \leqslant a_i^h \qquad \forall i \in V, \forall h \in D \tag{3.27}$$

$$\sum_{h \in D} z_i^h = 1 \qquad \forall i \in V \tag{3.28}$$

$$\sum_{i \in V} u_{ir}.z_i^h \leqslant cap_r^h \qquad \forall r \in R, \forall h \in D \tag{3.29}$$

$$z_i^h \in \{0,1\} \qquad \forall i \in V, \forall h \in D$$

$$f_{hk}^i \geqslant 0 \qquad \forall i \in V, \forall h,k \in D$$

The objective function (3.25) aims to minimize the amount of traffic generated by communicating VMs on the backbone network. The constraint (3.26) ensures the flow conservation. As for the constraint (3.27), it is a location constraint that indicates that the placement of different VMs must be restricted to a particular number of DCs that satisfy location constraint. This constraint aims to maintain service performance and to reduce time delay by placing high-communicating VMs in proximity of end-users. The matrix denoted by a_i^k is an input of the problem. It can be produced by measuring performance between each pair of nodes of the graph G. The constraint (3.28) ensures that every VM is running on only one DC. The final constraint (3.29) is a capacity constraint. It ensures that the amount of hardware resources consumed by different VMs placed in a given DC does not exceed the hardware capacities of this DC.

The aim of this formulation is to solve optimally the problem of placing communicating VMs with correlated traffic in geographically distributed DCs for large-scale Cloud system. Unfortunately, this formulation is time and resource consuming. Thus, in order ro reduce the execution time of the linear programs, we have applied variable aggregation techniques as shown in the next section.

3.4.2 Variable Aggregation

We note that the formulation presented above, can be reformulated to another equivalent formulation that turns out to be more efficient as it reduces the number of the variables.

In this formulation, we consider new decision variables:

- The first, designates the amount of traffic originated from a VM $i \in V$ and destined to a DC $h \in D$.

$$v_{ih} = \sum_{k \in D} f_{kh}^i \quad \forall i \in V, \forall h \in D \tag{3.30}$$

- The second decision variable designates the amount of traffic generated by VMs and sent to the backbone network.

$$\varphi_{ih} = \sum_{k \in D} f_{hk}^i \quad \forall i \in V, \forall h \in D \tag{3.31}$$

Since we aim to minimize the amount of traffic circulating on the backbone network and we do not consider any associated cost, there is no need to consider the first decision variable in the objective function and in the flow conservation constraint. Thus, this reformulation reduces considerably the number of variables which becomes $|V|.|D|$ instead of $|V|.|D|^2$. A comparative experiment study will be provided in next section.

Suppose that we have $|V| = 1000$ and $|D| = 10$, then the number of variables of the classical formulation (CF) is equal to 10^5. However, when we adopt variable aggregation approach, the number of variables is reduced to 10^4. Hence, the new equivalent formulation denoted by (AG) is presented as follows:

$$\min \sum_{i \in V} \sum_{h \in D} \varphi_{ih} \tag{3.32}$$

Subject to:

$$O_i.z_i^h - \sum_{j \in V} d_{ij}.z_j^h \leqslant \varphi_{ih} \qquad \forall i \in V, \forall h \in D \tag{3.33}$$

$$z_i^h \leqslant a_i^h \qquad \forall i \in V, \forall h \in D \tag{3.34}$$

$$\sum_{h \in D} z_i^h = 1 \quad \forall i \in V \tag{3.35}$$

$$\sum_{i \in V} u_{ir}.z_i^h \leqslant cap_r^h \qquad \forall r \in R, \forall h \in D \tag{3.36}$$

$$z_i^h \in \{0, 1\} \qquad \forall i \in V, \forall h \in D$$

$$\varphi_{ih} \geqslant 0 \qquad \forall i \in V, \forall h \in D$$

The new objective function (3.32) aims to minimize the amount of traffic sent to the backbone network. By minimizing the traffic sent to the backbone network, we implicitly reduce the traffic between the DCs that are connected through this network. In the flow conservation constraint (3.33), we replace the decision variables used in the first model by the new decision variables defined in (3.31) and (3.30). The constraints (3.34), (3.35) and (3.36) are exactly the same constraints of the first model (CF).

Theorem 1. The classical formulation (CF) and the aggregated formulation (AG) are equivalent.

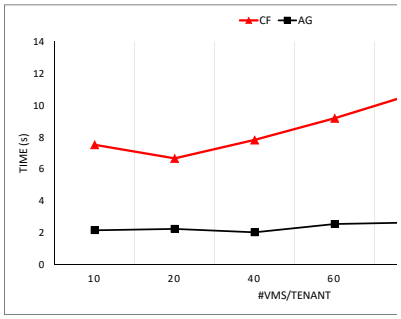
Proof. To prove the equivalence of the two formulations, we will show how to obtain a CF solution from an AG solution and vice versa.

It is worth noting that the value of the objective function does not change. So if we can obtain a CF (AG) solution from an AG (CF) solution, then the solution that minimizes CF (AG) will minimize also a corresponding solution in AG and the optimal solution for the problem can be obtained equivalently by means of CF or AG.

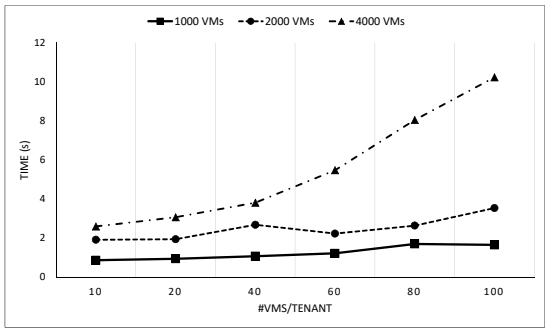
Let us consider the AG formulation. The variable aggregation that has been exploited to transform CF to AG formulation are the following:

$$v_{ih} = \sum_{k \in D} f_{kh}^i \qquad \forall i \in V, \forall h \in D \tag{3.37}$$

$$\varphi_{ih} = \sum_{k \in D} f_{hk}^i \qquad \forall \in V, \forall h \in D \tag{3.38}$$



(a) Time versus number of VMs p
CF and *AG*.



(b) Variation of the execution time of *AG* with respect to the number of VMs per tenant.

Figure 3.4: Experiment results performed on *CF* and *AG*.

The flow conservation constraint in CF can be written as follows:

$$x_i^h.O_i - \sum_{j \in V} d_{ij}.x_i^h = \varphi_{ih} - v_{ih} \qquad \forall i \in V, \forall h \in D \tag{3.39}$$

From CF to AG: Let us consider a generic admissible solution for CF $\langle \overline{f_i^{kh}}, \overline{x_i^h} \rangle$. To obtain the AG formulation, just sum the flows originated from i (source node) and destined to the backbone network. Since we aim to minimize the amount of traffic circulating on the backbone network and we do not consider any associated cost, there is no need to consider the first decision variable in the objective function and in the flow conservation constraint.

Thus, the flow conservation constraint can be written as follows:

$$x_i^h.O_i - \sum_{j \in V} d_{ij}.x_i^h \leqslant \varphi_{ih} \qquad \forall i \in V, \forall h \in D \tag{3.40}$$

From AG to CF: Let us consider a generic admissible solution for AG $\langle \overline{\varphi_{ih}}, \overline{x_i^h} \rangle$. To obtain the flow conservation equality, we add an intermediary decision variable as follows:

$$v_{ih} = \sum_{k \in D} f_{kh}^i \qquad \forall i \in V, \forall h \in D \tag{3.41}$$

Thus, we ensure the conservation of the flows destined to the backbone and the flows received via the backbone. Hence, the flow conservation constraint follows immediately. □

In the next section, we present experiments results showing the effectiveness of the proposed optimization model.

3.4.3 Experiment Results

In this section, we present the experiment results conducted on the optimization models presented in Section 3.4.

We have used three instance types (Small, Medium and Large) which are provided by Amazon Elastic Computing Cloud (EC2). Without loss of generality, we

	(AG)			(CF)		
	S	T	G	S	T	G
$(2000, 10)$	31713	2,176	0	31713	7,538	0
$(2000, 20)$	65287	2,262	0	65287	6,684	0
$(2000, 40)$	132221	2,047	0	132221	7,842	0
$(2000, 60)$	197789	2,568	0	197789	9,202	0
$(2000, 80)$	255693	2,663	0	255693	10,748	0
$(2000, 100)$	314895	2,683	0	314895	12,229	0

Table 3.2: Equivalence between (CF) and (AG).

assume that all DCs have the same hardware capacities. We consider that the servers are housed in racks. Every server has 8 cores and 16 GB of RAM. We consider that each rack hosts 30 server and each DC has an average of 500 racks. For each experiment, we randomly generate 10 groups of tenant requests. All the experiment results are averaged.

Let us denote by:

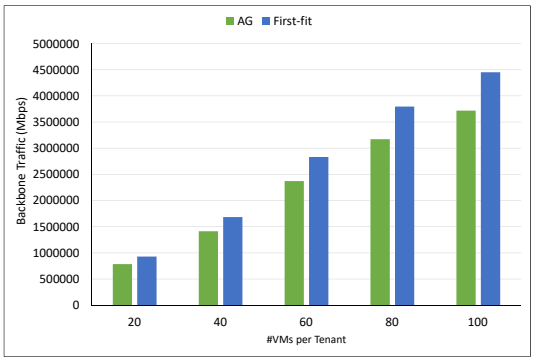
- S, is the value of the optimal solution provided by CPLEX and expressed in (Mbps).
- T, is the convergence time, expressed in seconds.
- G, is the gap between S and the lower bound provided by CPLEX and expressed in %. Note that $G = 0$ indicates that the optimal solution is reached.

The Table (3.2) demonstrates the equivalence between the two formulations (CF) and (AG). Both formulations provide exactly the same values of the objective function.

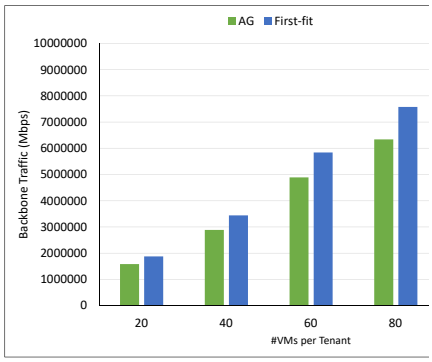
In order to show the effectiveness of the strengthening technique that we have used, we have compared the performance of the two formulations (CF) and (AG) using the same data instances. We fixed $|V| = 1000$ and we plotted the execution time for an increasing number of tenant requests. The results are depicted in Figure (3.4a). The results show the effectiveness of the variables aggregation approach. Moreover, the values of the execution time of (AG) are more stable than those of (CF) as the number of VMs per tenant increases.

To verify the scaling properties of the final linear program (AG), we fixed the number of DCs ($|D| = 6$), and plotted the execution time for an increasing number of VMs ($|V| = 1000$ to 4000) while varying the number of VMs per tenant. Figure (3.4b) shows the obtained results. We note that the execution time of the optimization model increases with the number of inter-communicating VMs. However, it remains reasonable even for 4000 VMs. Indeed, it does not exceed a dozen of seconds.

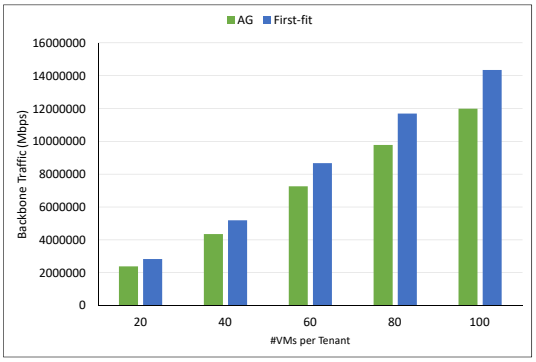
In order to show the quality of the solution provided by AG in terms of minimizing the traffic volume, we have compared it with the well-known placement algorithm *First-Fit*. We have chosen to compare our proposal with the *First-Fit* algorithm as the use of the latter is very common in cloud systems [104], [105], [106]. We have fixed the number of DCs to six and we have varied the number of VMs per tenant from 20 to 100. We have plotted the values of the objective function of (AG), which



(a) Backbone traffic versus the number of VMs/tenant, $|V| = 1000$.



(b) Backbone traffic versus the n VMs/tenant, $|V| = 2000$.



(c) Backbone traffic versus the number of VMs/tenant, $|V| = 3000$.

Figure 3.5: Backbone traffic for $|V| = 1000$, 2000 and 3000 VMs.

describes the amount of traffic in the backbone network in ($Mbps$). Then, we have compared the values provided by the solver and the values provided by *first-fit*. We have conducted this test for a range of VMs varying from 1000 to 3000. The results are depicted in figure (3.5). Compared to the first-fit algorithm, it can be seen that our model (AG) reduces the backbone traffic by 19% for symmetric traffic matrices.

In the next chapters, we will use the formulation denoted by (AG) to solve the *initial* VM placement problem denoted by IVMP.

3.5 Conclusion

In this chapter, we have focused on the offline problem of VM placement in a geographically distributed DCs. We presented different ILP formulations to solve this problem. Moreover, we used several strengthening techniques in order to enhance the execution time of the linear programs. However, the proposed models are static (i.e offline). They do not consider a reconfiguration of the cloud system and they are often used to solve the *initial* VM placement problem. Hence, in the next chapter, we study the online (i.e. dynamic) version of the problem which considers live VM migration.

Chapter 4

Online VM Placement and Migration Optimization in Geo-Distributed DCs

Contents

4.1 Introduction

In this chapter, we formally define the online VM placement problem in geo-distributed DCs. Due to the complexity of the problem, we divide it into two sub-problems. First, we remind the formulation of the *Initial* (i.e. offline) VM placement problem that we have presented in the previous chapter. Second, we formally define the dynamic (i.e. online) VM placement problem in geo-distributed DCs. Finally, we detail and comment the experiment and simulation results conducted on the proposed optimization models. This work has been presented in [107] and [108].

4.2 Problem Description

We model the physical infrastructure by a complete graph $G(V \cup D, E)$, where V denotes the set of VMs and D the set of DCs. The set of edges E represents the set of the links of the backbone network.

Due to the complexity of the problem, we divide it into two sub-problems:

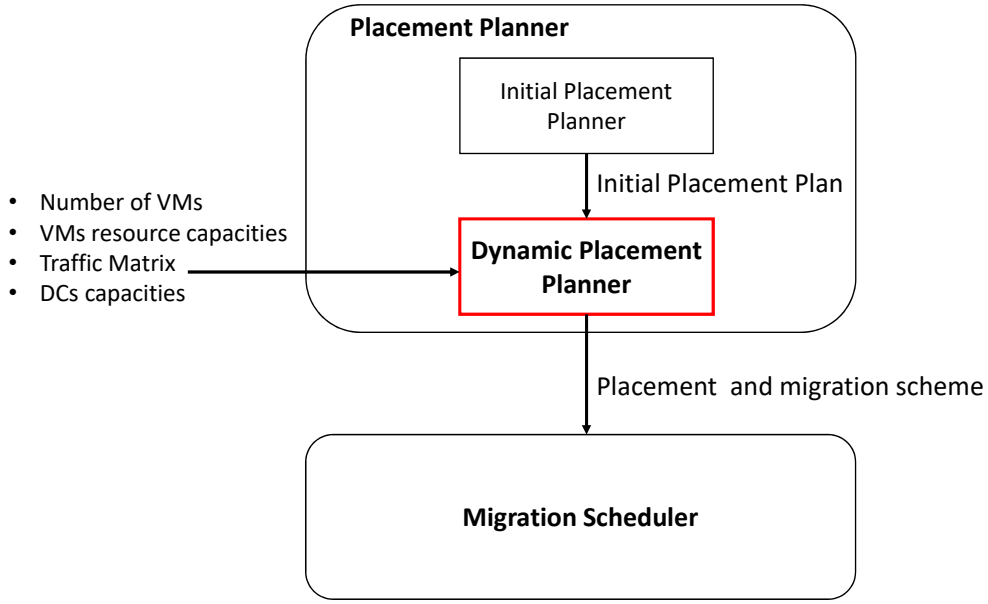


Figure 4.1: Dynamic Placement Planner Overview.

- *The Initial Placement Problem.* It consists of finding the optimal initial (or first) placement scheme for intercommunicating VMs such that the location and capacity constraints are all satisfied while minimizing the amount of traffic circulating on the backbone network. It is an offline model that provides static placement plans.
- *The Dynamic VM Placement Problem.* It invokes the VM migration as well as the interaction between the initial (or the previous) placement scheme and the migration decisions.

We define the *Dynamic VM Placement Problem* (DVMP) by considering two states of the system:

- *Initial state:* Refers to the initial (or the previous) placement scheme of the VMs in the cloud system.
- *New state:* Refers to the new configuration of the cloud system, taking into account the migration of the already existing VMs and the placement decisions.

In the next section, we present the optimization models used to solve the *DVMP* in geo-distributed DCs.

4.3 Problem Formulation

In this section, we formally define the *DVMP* within a distributed Cloud infrastructure as a MILP programs.

4.3.1 Initial VM Placement Problem

The following formulation has been presented in Chapter 3, Section 3.4.2. We refer to this formulation by the *Initial VM Placement Problem* (IVMP). For the sake of

simplicity, we consider the same decision variables indexed by (0) to indicate the initial (previous) state of the system:

- We designate by $\varphi_i^{(0)h}$ the amount of traffic originated from the VM $i \in V$ and sent from the DC $h \in D$.
- We define the decision variable $x_i^{(0)h}$ as:

$$x_i^{(0)h} = \begin{cases} 1 & \text{If the VM } i \text{ is placed in the DC } h \\ 0 & \text{Otherwise} \end{cases}$$

We denote by O_i the total traffic emanating from a VM i. We have:

$$O_i = \sum_{j \in V} d_{ij} \quad \forall i \in V \tag{4.1}$$

Our objective is to minimize the amount of traffic generated by the communication between different VMs in order to prevent possible link congestion. Hence, the objective function (4.2) can be defined as follows:

$$\min \sum_{i \in V} \sum_{h \in D} \varphi_i^{(0)h} \tag{4.2}$$

Subject to the following constraints:

$$\varphi_i^{(0)h} \geqslant O_i . x_i^{(0)h} - \sum_{j \in V} x_j^{(0)h} . d_{ij} \qquad \forall i \in V, \forall h \in D \tag{4.3}$$

$$\sum_{h \in D} x_i^{(0)h} = 1 \qquad \forall i \in V \tag{4.4}$$

$$\sum_{i \in V} u_{ir} . x_i^{(0)h} \leqslant cap_r^h \qquad \forall r \in R, \forall h \in D \tag{4.5}$$

$$x_i^{(0)h} \leqslant a_i^{(0)h} \qquad \forall i \in V, \forall h \in D \tag{4.6}$$

$$\varphi_i^{(0)h} \geqslant 0 \qquad \forall i \in V, \forall h \in D$$

$$x_i^{(0)h} \in \{0, 1\} \qquad \forall i \in V, \forall h \in D$$

The first constraint (4.3) ensures the flow conservation. As for the constraint (4.4), it ensures that every VM is running on only one DC. The constraint (4.5) represents the capacity constraint on the DCs. It ensures that the amount of resources consumed by different VMs placed in a given DC does not exceed the resource capacity of this DC. The final constraint (4.6) is a location constraint that restricts the placement of VMs to a particular number of DCs that satisfy a location constraint. This constraint aims to maintain service performance and to reduce time delay by placing VMs with high communication volumes in proximity of end-users. This MILP formulation provides the initial (i.e. first) placement scheme of VMs. We consider that the cloud system is in the initial sate.

In the next section, we present the dynamic version of the VM placement problem in a distributed cloud infrastructure.

4.3.2 Dynamic VM Placement Problem

Because of the fluctuating demand, it is important to adjust the placement of different VMs dynamically and in an online manner. In this work, we consider live migration of VMs over WAN that connects different DCs placed in different regions. In fact, VM live migration is a technology that offers the ability to migrate VM through WAN from one DC to another [109]. VM live migration brings multiple benefits. It provides higher performance and improves the QoS [110].

However, in WAN VM live migration, a significant amount of traffic is generated during the migration process. Literally, for a WAN live migration, it is crucial to transfer the VMs images as well as its local persistent state and its on-going network connections especially for distributed and intensive Input/Output applications. Nevertheless, with the lack of high end-to-end network bandwidth over WANs and the potential transfer of large amounts of data, the downtime and migration time are expected to be high [45]. Hence, it is important to consider the minimization of the backbone traffic in the migration decisions.

In a dynamic cloud environment, migration can be performed within different scenarios. In this work, we consider two main scenarios of VM migration across distributed cloud infrastructure:

1. The arrival or/and departing of VMs.

2. The change of the traffic matrix.

Let us consider V_o as the set of VMs that have been already placed in the system, V_n as the set of new arriving VMs and V_d the set of departing VMs. We denote by $V = (V_o \cup V_n) \setminus V_d$ the set of VMs in the system at a certain time t. This formulation takes as an input the solution of the initial or previous placement problem modeled by $x_i^{(0)h}$ which refers to the previous location of the VM $i \in V_o$ in the system.

In fact, $x_i^{(0)h}$ is provided for the first time by solving the optimization model *IVMP* presented in the previous chapter. At the beginning of each time slot, the *DVMP* program is executed and the values of $x_i^{(0)h}$ are updated with the current placement scheme before the reconfiguration of the system.

We consider the following decision variables indexed by (1) to identify the new state of the system.

- We designate by $\varphi_i^{(1)h}$ the amount of traffic originated from the VM $i \in V$ and sent from the DC $h \in D$.

- We define the binary decision variable $x_i^{(1)h}$ as follows:

$$x_i^{(1)h} = \begin{cases} 1 & \text{If the VM } i \text{ is placed in the DC } h \\ 0 & \text{Otherwise} \end{cases}$$

- To model the migration decision, we introduce the binary decision variable z_i as follows:

$$z_i = \begin{cases} 1 & \text{If the VM } i \in V_o \text{ and } i \text{ is migrated} \\ 0 & \text{Otherwise} \end{cases}$$

The objective of this formulation is to minimize the traffic on the backbone network (i.e. Inter-DCs traffic). In fact, we consider the sum of the traffic generated by the communication between VMs and the traffic generated during the migration process. The migration decision concerns only the VMs ($i \in V_o$) that are already placed in the cloud system. Let us consider M_i the amount of traffic generated by the migration of the VM $i \in V_o$. As we consider WAN migration, we define the migration traffic as a function of the memory size of the VM and its local disk size.

The objective function (4.7) minimizes the amount of traffic circulating on the backbone network, which consists in the communication traffic and the traffic generated during the migration process.

$$\min \sum_{i \in V} \sum_{h \in D} M_i.z_i + \varphi_i^{(1)h} \tag{4.7}$$

Subject to the following constraints:

$$z_i \geqslant |x_i^{(1)h} - x_i^{(0)h}| \qquad \forall i \in V, \forall h \in D \tag{4.8}$$

$$\varphi_i^{(1)h} \geqslant O_i.x_i^{(1)h} - \sum_{j \in V} x_j^{(1)h}.d_{ij} \qquad \forall i \in V, \forall h \in D \tag{4.9}$$

$$\sum_{h \in D} x_i^{(1)h} = 1 \qquad \forall i \in V \tag{4.10}$$

$$\sum_{i \in V} u_{ir}.x_i^{(1)h} \leqslant cap_r^h \qquad \forall r \in R, \forall h \in D \tag{4.11}$$

$$x_i^{(1)h} \leqslant a_i^{(1)h} \qquad \forall i \in V, \forall h \in D \tag{4.12}$$

$$\varphi_i^{(1)h} \geqslant 0 \qquad \forall i \in V$$

$$x_i^{(1)h} \in \{0, 1\} \qquad \forall i \in V, \forall h \in D$$

$$z_i \in \{0, 1\} \qquad \forall i \in V_o$$

If a VM is migrated, its old location is obviously different from its new one. This fact is modeled by the set of constraints (4.8). As for the constraint (4.9), it is a flow conservation constraint. The constraint (4.10) ensures that a VM is running on only one DC. The capacity constraint on the DCs is ensured by the constraint (4.11). Finally, constraint (4.12) refers to the location constraint. This model provides the optimal migration and placement scheme for different VMs that minimizes the backbone traffic constituted by both communication and migration traffic. However, it does not provide the migration scheduling plan of different VMs to be migrated. Nevertheless, as we will show in the next chapter, the migration sequence of inter-communicating VMs has an influence on the overall amount of network traffic and may lead to link congestion and performance degradation of the whole system.

In the next section, we present a numerical example showing the behavior of both the *Initial Placement Planner* and the *Dynamic Placement Planner*.

4.3.3 A Numerical Example

In this section, we consider an example illustrating that the models of the prior subsections work well with migration at minimizing backbone traffic. Let us consider

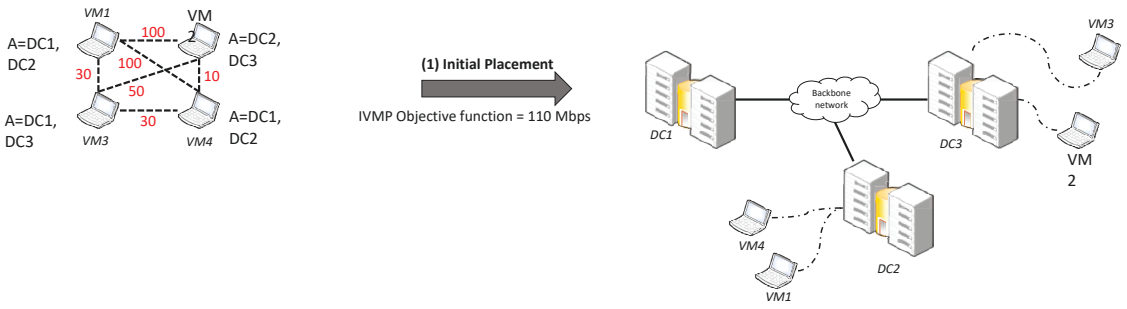


Figure 4.2: Initial Placement scenario (IVMP).

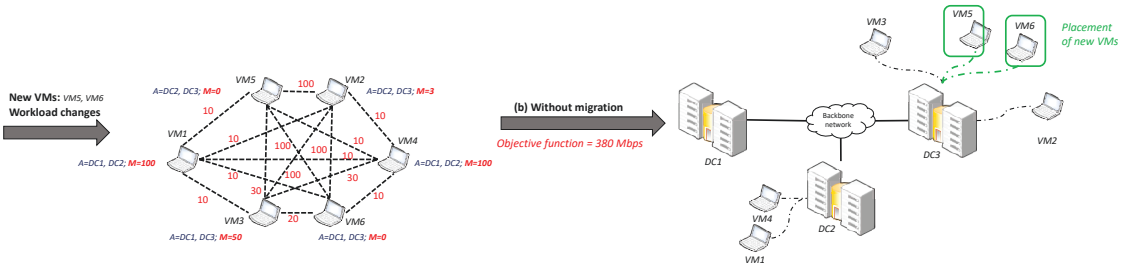
the example presented in Figure (4.2). We are given an application composed of 4 VMs, namely, *VM1, VM2, VM3* and, *VM4* exchanging data between them. For simplicity purposes, we assume that the traffic matrices are symmetric. Tables (4.1a) and (4.1b) present the amounts of network traffic exchanged between each pair of VMs.

First, we need to place this application in the cloud system, for the first time, while simply ensuring the location constraint (represented by the vector A in the figure). This constraint restricts the placement of the VMs in a certain set of DCs. In addition, the placement scheme must ensure minimum inter-DCs traffic. With the above data, the *IVMP* model presented in Section 4.3.1, has been solved and has provided an initial optimal placement scheme. We note that the objective function of the *IVMP* model, which represents the overall traffic exchanged between DCs, is equal to 110 Mbps.

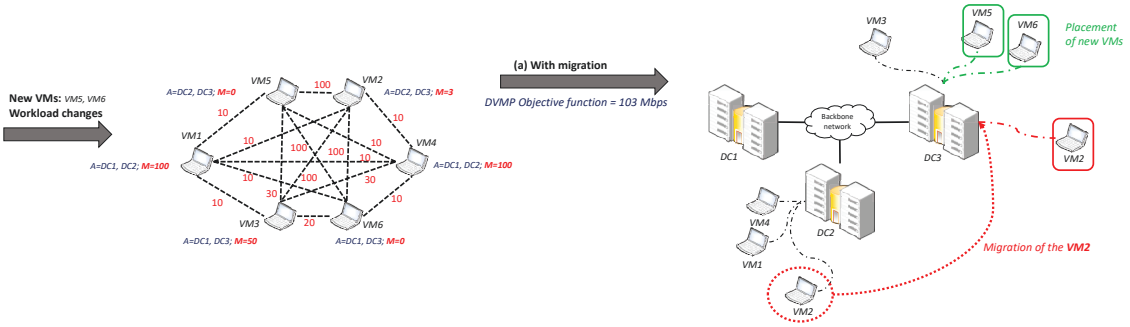
A scenario we consider consists in the arrival of two VMs *VM5* and *VM6*, that need to be placed. The traffic matrix between all VMs has also been changed. Let us consider a first situation where no VM migration is performed as illustrated by Figure (4.3a). In this case, we need to simply place the new VMs and ignore the change in the traffic matrix. We obtain thus, an overall backbone traffic that equals to 380 Mbps.

Let us now solve the *DVMP* presented in Section 4.3.2. This model minimizes both communication and migration traffic. The *DVMP* decides that *VM2* must be migrated from *DC1* to the *DC2* as presented in Figure (4.3b). The cost of migrating the different VMs is illustrated by the letter M in the figure. We note that the objective function of the *DVMP* represents the sum of the inter-DCs communication traffic and the traffic generated during the migration process. After performing the migration, the new objective function is equal to 103 Mbps. In contrast to the example of Figure (4.3a), the migration of *VM2* has minimized the backbone traffic and has provided a better solution for the problem.

In the next section, we present the different experiments that we have conducted on the proposed MILP formulations in order to evaluate the performance and the effectiveness of our solutions.



(a) Placement of new VMs without migration.



(b) Execution of DVMP (with migration).

Figure 4.3: Example of VM placement with and without migration.

	VM1	VM2	VM3	VM4
VM1	0	100	30	100
VM2	100	0	50	10
VM3	30	50	0	30
VM4	100	10	30	0

(a) Initial Traffic Matrix.

	VM1	VM2	VM3	VM4	VM5	VM6
VM1	0	10	10	30	10	10
VM2	10	0	100	10	100	100
VM3	10	100	0	30	30	20
VM4	30	10	30	0	10	10
VM5	10	100	30	10	0	100
VM6	10	100	20	10	100	0

(b) New Traffic Matrix.

Table 4.1: Traffic Matrix values.

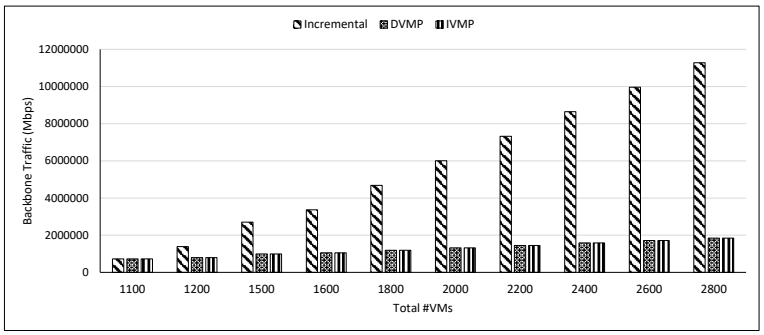
4.4 Performance Evaluation

In order to evaluate the performance of our modules presented in the previous sections, we have first evaluated the effectiveness of the proposed exact methods in terms of execution time and quality of the provided solutions using the commercial solver CPLEX [92]. Then, to validate our approach under realistic conditions, we have used the simulation toolkit CloudSim [93].

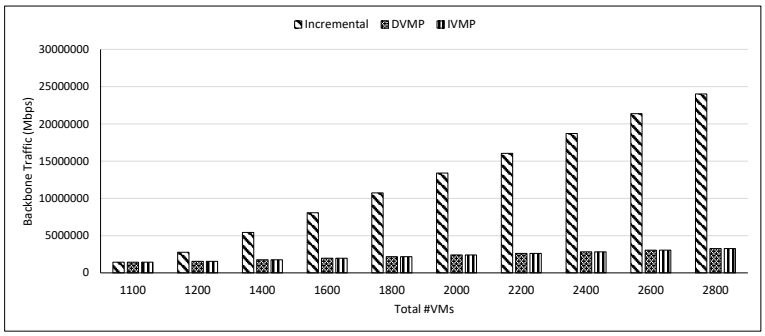
In the following, we present the configurations of the conducted experiments, the performance metrics that we have evaluated as well as the obtained results.

4.4.1 Experiments using CPLEX

The different experiments were carried out on a machine that has an Intel Xeon 3; 3 GHz CPU and 8GB of RAM. We have used the commercial solver CPLEX 12.5 to solve and evaluate the different MILP formulations.



(a) Backbone traffic versus total # VMs (20 VMs/Tenant).



(b) Backbone traffic versus total # VMs (40 VMs/Tenant).

Figure 4.4: Comparing the incremental, IVMP and DVMP for static traffic matrix.

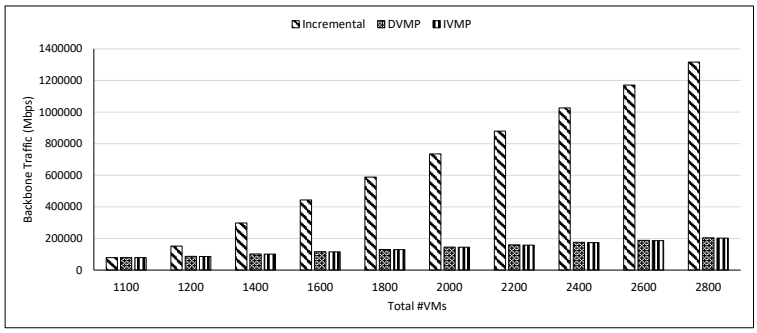
Data Centers. Without loss of generality, we assume that all DCs have the same hardware capacities. We consider that the servers are housed in racks. Every server has 8 cores and 16 GB of RAM. We consider that each rack hosts 30 servers and each DC has an average of 500 racks. In all experiments, we have fixed the number of DCs to six.

VM Request. Each tenant may send a VM request. We consider that the VM requests are independent. Each request represents a set of VMs that may exchange data with each other. We assume that VMs have an instance type (Small, Medium and Large). The different values of hardware metrics are provided by Amazon Elastic Computing Cloud (EC2) [29]. Without loss of generality, we assume that each VM can be placed on two possible DCs, known a priori, but eventually, each VM is assigned to only one DC.

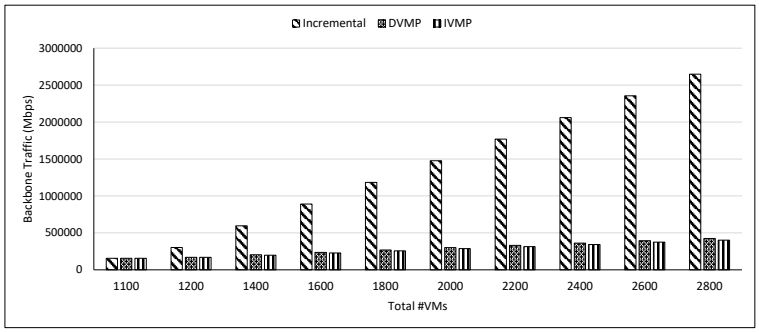
Traffic Matrix. In order to study the performance of our models, we consider symmetric traffic matrices where its values vary between 1 and 100 Mbps. The traffic matrix represents communication traffic or bandwidth requirement between each pair of VMs. We have generated the traffic matrix randomly.

Performance of the Placement Planner Module

In this section, we present the experiment results conducted on the proposed formulations. We study different performance evaluation metrics.



(a) Backbone traffic versus total # VMs (20 VMs/Tenant).



(b) Backbone traffic versus total # VMs (40 VMs/Tenant).

Figure 4.5: Comparing the incremental, IVMP and DVMP models for a dynamic traffic matrix.

The Initial VM Placement Planner. The experiment results conducted on the *IVMP* are presented in Chapter 3, Section 3.4.3. Please refer to the previous chapter for more details.

The Dynamic Placement Planner. In this section, we present different experiments conducted on the *DVMP* formulation presented in Section 4.3.2.

First, in order to test the efficiency of our solution, we have compared it with other placement algorithms. First, we have implemented an *incremental* placement algorithm. The main idea of this algorithm is to fix the placement scheme of the VMs that are already placed in the system and try to place new arriving VMs according to the residual capacities of DCs. In this case, VM migration is not considered and the incremental algorithm takes placement decisions for the new arriving VMs only.

Then, we have executed the *IVMP* at the beginning of each time slot. In contrast to the *DVMP* model, the *IVMP* model considers that all future demands are known a priori and no migration cost is considered. It provides at the beginning of each time slot new placement scheme of both the existing VMs and the new ones. This scheme is considered as the *ideal* placement scheme since it does not consider migration costs.

We have considered the case of a static traffic matrix, where the values of the exchanged data between each pair of VMs do not vary over time. In this set of tests, the number of initially placed VMs is fixed to 1000 VMs. The results of the conducted tests are depicted in Figure (4.4).

We note that the incremental solution diverges and the gap between the *DVMP*

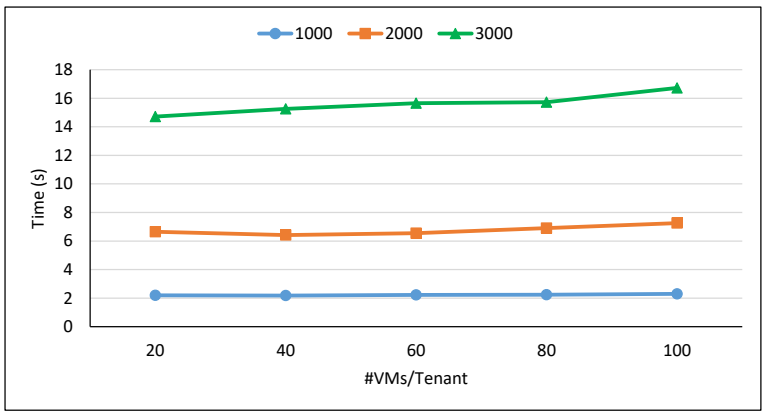


Figure 4.6: Convergence time of the DVMP versus the number of VMs/Tenant.

algorithm and the *incremental* algorithm reaches 80%. On the other hand, we observe that there is no gap between *IVMP* and *DVMP*. This result can be explained by the fact that for a static traffic matrix, there is no change in the communication pattern, hence, *DVMP* places the new arriving VMs without making any migration since the residual capacities of the DCs are able to satisfy all new VMs requests.

In a dynamic environment, the inter-VM bandwidth requirement (i.e inter-VMs communication) may change over time. Hence, it may be profitable to adjust the VMs placement scheme according to the fluctuation of the traffic matrix. In this set of tests, the number of initially placed VMs is fixed to 1000 VMs. The results of the conducted tests are shown in Figure (4.5).

We can make the same observations regarding the gap between the *incremental* algorithm and *DVMP*. The gap between the two solutions reaches almost 85%. Furthermore, when the traffic pattern changes, we note that *DVMP* performs some migrations. In contrast, the gap between the *IVMP* and the *DVMP* remains very small. It can be explained by the fact that the number of migrations remains very small compared to the total number of VMs in the system. Thus, the difference between the two solutions is very small.

In order to evaluate the performance of the *DVMP*, we have varied the number of VMs per tenant (from 20 to 100 VMs per tenant). We have plotted the execution time of the *DVMP*. The results are shown in Figure (4.6). We note that the graph is almost constant even when we increase the number of VMs per tenant for a total number of VMs varying from 1000 to 3000. We can conclude that the number of VMs per tenant has no impact on the execution time of *DVMP*.

Despite the benefits of VM migration, extensive migrations may impact the whole system performance. Hence, it is crucial to keep the number of migrations as small as possible. To study this metric, we have compared the number of migrations considered by both *DVMP* and *IVMP*. In *DVMP*, we consider that each VM has a migration cost as for the *IVMP* algorithm, which will be executed iteratively at the beginning of each time slot, no migration cost is considered. The results are depicted in Figure (4.7).

We note that for *IVMP*, the number of migrations is very high and varies a lot over time. As for *DVMP*, the number of migrations is very small compared to the total number of VMs in the system and it is more stable over time. Regardless of this huge difference, the values of the objective solution for both algorithms are

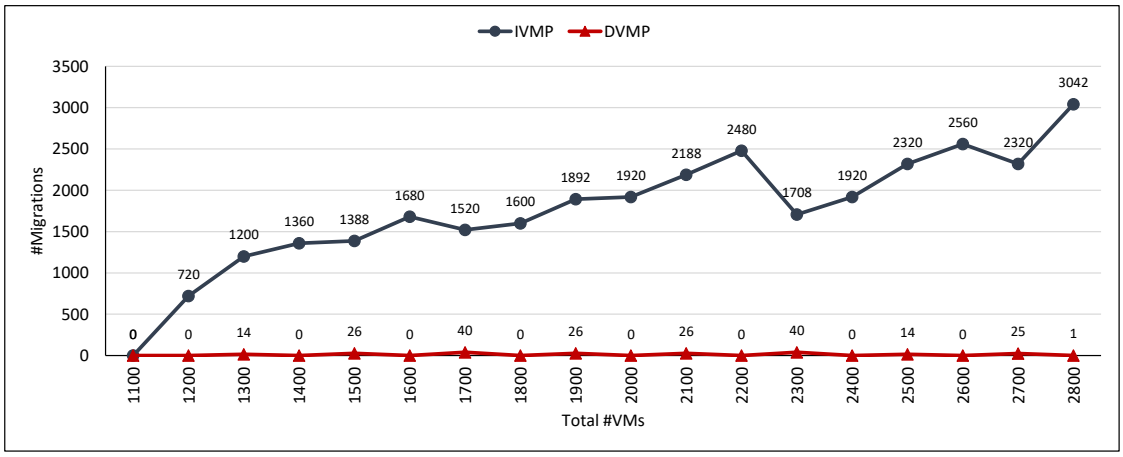


Figure 4.7: Variation of the # migrations for both IVMP and DVMP.

very close. We can conclude that *DVMP* gives almost the same solution as *IVMP*, which can be considered as the "*ideal*" solution, but in contrast to *IVMP*; it is able to maintain the system performance and QoS by minimizing the number of migrations.

In the next section, we present the simulation results conducted on the proposed optimization models.

4.4.2 Simulations using CloudSim

Evaluating the performance and the efficiency of placement policies in real cloud environment under critical conditions and for different applications and service models is challenging. In fact, the use of real cloud environment has several limitations such as the system size and configuration which make the reproduction of some results a very difficult task. Moreover, with a real cloud system, the evaluation of some critical scenarios and failures is not supported. In addition, the access to real cloud environment is costly [111].

To cope with these limitations, there are some simulation tools that offer the possibility of evaluating different placement, allocation and scheduling policies under different conditions. These tools, provide a controllable environment, free of cost, that mimic the behavior of a real cloud environment. In such an environment, users may test their models within critical situations in order to study the overheads of their models and to cope with possible performance degradation before deploying in real world [111].

In [112], the authors gave an analysis of the existent simulation tools in Cloud Computing. Indeed, simulation techniques are used in several science research fields. It is based on Information Technology, Principal of Similarity and Modeling Theory. It uses simulation models of real or conceptual systems for dynamic experimentation [113].

In this work, we have used the Cloud simulator CloudSim [111]. CloudSim is an open source simulator, which enables seamless modeling, simulation, and experimentation of cloud computing and application services. In addition, CloudSim has been widely used in the literature to perform simulations in cloud systems [114, 72, 115].

We have implemented an extension of the CloudSim simulator that enables the

Type	Characteristics
HP ProLiant ML110 G4	1 x [Xeon 3040 1860 MHz, 2 cores], 4GB
HP ProLiant ML110 G5	1 x [Xeon 3075 2660 MHz, 2 cores], 4GB

Table 4.2: Host characteristics.

Instance	MIPS	PES	RAM
High-CPU Medium Instance	2500	1	870
Extra Large Instance	2000	1	1740
Small Instance	1000	1	1740
Micro Instance	500	1	613

Table 4.3: VM instance types.

simulation of the communication between VMs placed in distant DCs. In addition, the extension that we have implemented allows the migration of VM from a DC to distant one. This work has been presented in [108].

We have varied the number of DCs $D = 4, 6, 8, 10$. The capacity of a DC is directly related to its number of physical hosts. The DCs are considered heterogeneous. We have used two types of hosts. The host characteristics are presented in the table (5.3). The VM instance types used in the simulation are presented in table (4.3). We assume that each VM can be placed in two possible DCs, known a priory, but eventually, each VM is assigned to only one DC. The number of applications as well as the exchanged traffic are generated randomly.

To mimic realistic VM arriving process, we have generated the VMs arrival request according to a *Poisson* distribution and we have varied the mean from 20 VMs per hour to 60 VMs per hour.

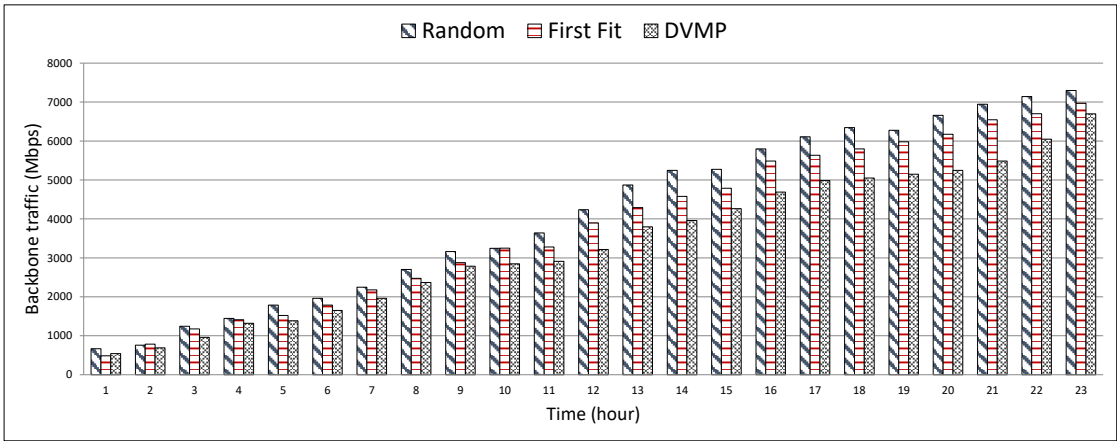


Figure 4.8: Backbone traffic (Mean VM arrival = 20 VMs per hour).

First, we have compared some baseline placement algorithms (random and first-fit) with our proposed one. We have run the simulation during 24 hours and have plotted the amount of traffic exchanged between the DCs at the beginning of each hour. The number of inter-communicating VMs is generated randomly according to

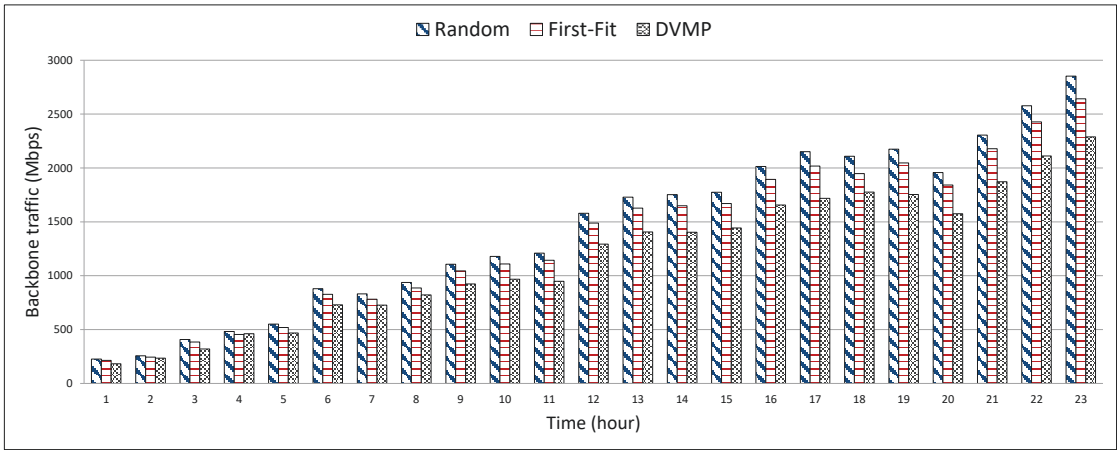


Figure 4.9: Backbone traffic (Mean VM arrival = 60 VMs per hour).

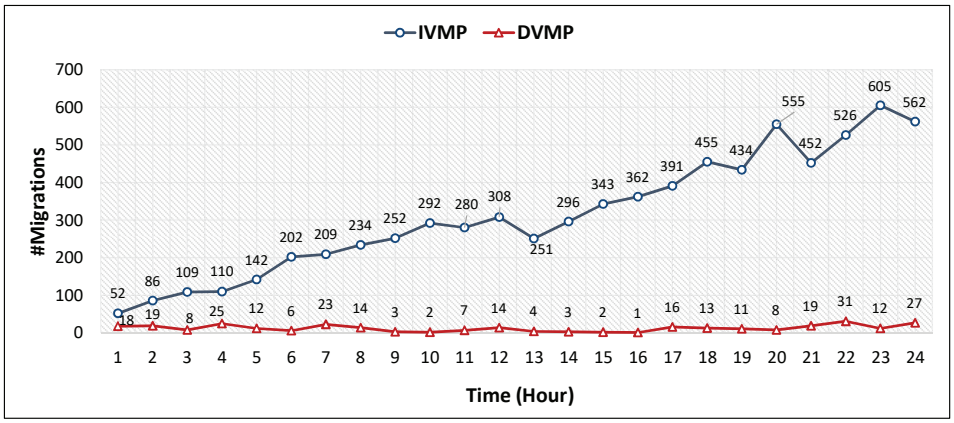


Figure 4.10: Number of migrations during 24 hours.

the *Poisson* distribution. The results are depicted in Figures (4.8) and (4.9).

Our placement policy gives more efficient placement plan that reduces the amount of the inter-DCs traffic compared to both *random* and *first-fit* algorithms.

Second, we have executed both *IVMP* and *DVMP* algorithms for 24 hours and have plotted the number of performed migrations. The results are depicted in Figure (4.10). As shown in the previous section, the number of migrations performed by the *IVMP* algorithm is huge compared to the number of migrations performed by the *DVMP* algorithm. We can conclude that the consideration of the migration cost has an important impact on the placement and migration decisions. In addition, excessive migrations may lead to a huge performance degradation in the cloud system [116] [117]. Hence, the *DVMP* algorithm provides placement and migration plans that aim to maintain the system stability and thus the system performance over time.

In the Figure (4.11), we have plotted the traffic exchanged between the different DCs and we have varied the number of DCs. The results show that the total inter-DCs traffic decreases when we add more DCs to the system.

In the next set of test, we have studied the impact of the number of DCs on the

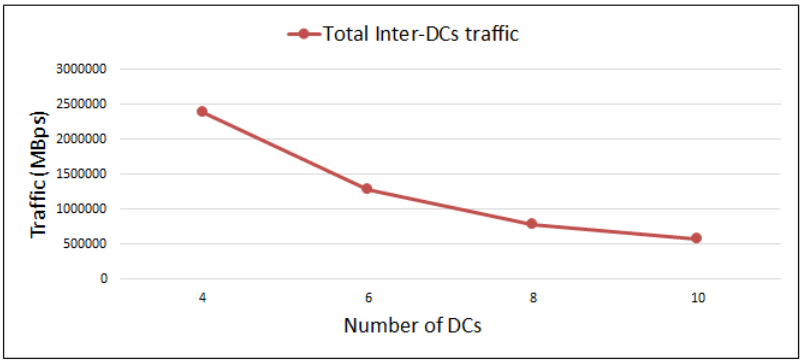


Figure 4.11: Total Inter-DCs traffic Vs number of DCs.

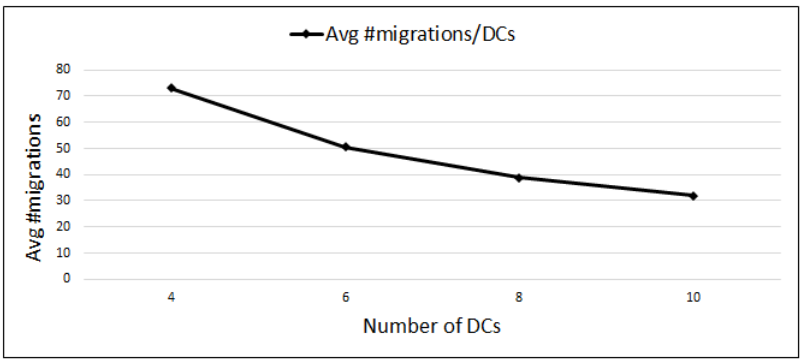


Figure 4.12: Average number of migrations per DCs Vs number of DCs.

number of performed migrations. Thus, we have varied the number of DCs from 4 to 10 and we have plotted the average number of migration per DCs. The results are depicted in figure 4.12. The graph shows that the average number of migration decreases when we add more DCs to the system.

4.5 Conclusion

In this chapter, we have proposed online MILP formulations that aim to solve the problem of dynamic VM placement across geographically distributed DCs. Our aim was to find an optimal placement and migration scheme for the different inter-communicating VMs. Through the conducted experiments, we have shown that the *DVMP* model is more efficient than the incremental model by almost 80%. In addition, the solution values of the *DVMP* model are very close to the solution values of the *IVMP* problem in terms of the amount of the backbone traffic. However, the variation of the number of migrations in the *DVMP* model remains very small compared to the huge number of migrations proposed by the *IVMP* model. Thus, the *DVMP* model ensures the stability of the system by minimizing the number of migrations while reducing the inter-DC traffic. Live Migration of inter-communicating VMs will produce additional traffic, as the VMs will continue to communicate with each other. Hence, the migration ordering has a huge impact on the overall network traffic during the migration process. In the next chapter, we will study this problem by proposing both exact and heuristic solution to solve it.

Chapter 5

Traffic-aware VM Migration Scheduling Problem

Contents

5.1 Introduction

In this chapter, we focus on the VM migration scheduling problem within a geo-distributed DCs. We propose both exact and heuristic methods to solve it. In fact, the migration of inter-communicating VMs over the backbone network can lead to the increase of the traffic on the network links. Hence, it is important to find the best migration scheduling of VMs that minimizes the communication traffic. An effective migration scheduling of VMs may prevent from network link congestion and maintain the performance of both VMs in source and destination as well as the migrating VM. The work presented in this chapter has been published in [107] and [118].

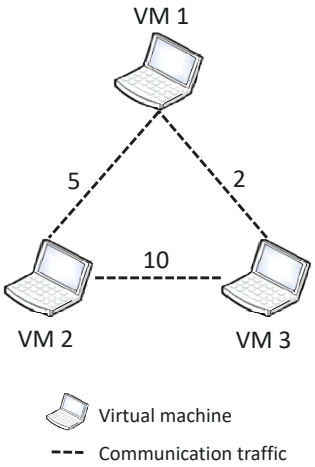


Figure 5.1: Example of three intercommunicating VMs.

5.2 Best Migration Sequence

In this section, we present both exact and heuristic solutions to solve the VM migration scheduling problem in geo-distributed DCs. Our aim is to minimize the traffic volume within the backbone network.

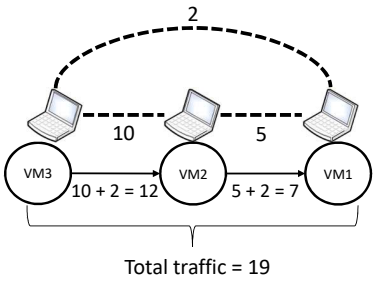
Furthermore, this paper investigates the problem of VM migration scheduling which aims to find the best migration sequence for inter-communicating VMs while ensuring the minimum backbone traffic. In fact, few proposals have dealt with the migration scheduling problem across WAN. However, their main concern was to study of the migration techniques [119], to reduce migration time [120], or to minimize the energy consumption during the migration [121].

In the following, we present an example showing the importance of VM scheduling problem.

5.2.1 Migration Scheduling Example

Let us consider the solutions provided by the *DVMP* and the *IVMP* optimization models presented in the previous chapters. Assume that some VMs, corresponding to the nonzero values of the variables $z_i = 1$, that must be moved from the DC h ($x_i^{(0)h} = 1$) to the DC k ($x_i^{(1)k} = 1$). These VMs are exchanging data flows given by the elements d_{ij} of the traffic matrix. At each step of the migration process, there is a certain amount of data traffic exchanged between the two DCs on which the migration is performed. The volume of this traffic depends on the migration sequence of the VMs. For instance, we consider the three VMs network depicted in Figure (5.1). A first order of migration could be $VM1 - VM2 - VM3$ as illustrated in Figure (5.2a). By moving the VM #1 the traffic is equal to 7, and then it raises to 15 after migrating the VM #3 and the total traffic becomes 15+7=22. Nevertheless, when considering the migration order of $VM3 - VM2 - VM1$ as depicted in figure (5.2b), the total flow is equal to 19 leading to a better solution.

In the next section, we formally define the VM migration scheduling problem as an optimization model.



(b) Second Scheduling scenario.

Figure 5.2: Migration Scheduling Example.

5.2.2 Exact Solution

In this section, we present the mathematical formulation proposed to solve the VM migration scheduling problem in geo-distributed DCs. As shown in the example of Section 5.2.1, we model the problem as a graph, where VMs (to be migrated) are considered as nodes and the traffic exchanged between each pair of VMs as the flow that will be sent.

In such a linear network, there is one single path for the flow between each pair of VMs. The total traffic engaged in the network is the sum of the products of the amounts of flow between each pair of VMs by the lengths of each path expressed in number of hops. Consequently, obtaining a minimum value of the total traffic consists in finding a Hamiltonian cycle of minimum value spanning all the VMs to be migrated.

It is important to underline that the migration scheduling problem can be decomposed by pairs of DCs, and thus, reducing its complexity. The order of DC pairs has no influence on the value of the total flow generated during to the migration process and injected into the backbone network.

Proposition 5. The total traffic caused by the migration scheduling is independent of the data centers pairs ordering.

Proof. At any step of the migration process, we denote by $x_i^h = 1$ if the VM remains connected to the DC h (i.e. placed in h). It takes the value 0 otherwise. The data traffic injected to the backbone network and caused by the migration process can be written as follows.

$$\min \sum_{i \in V} \sum_{j \in V} \sum_{h \in D} \sum_{k \in D} d_{ij} x_i^h x_j^k \tag{5.1}$$

Subject to:

$$\sum_{h \in D} x_i^h = 1, \ \forall i \in V. \tag{5.2}$$

But, from (5.2), we obtain:

$$x_i^k + \sum_{h \neq k} x_i^h = 1, \ \forall i \in V. \tag{5.3}$$

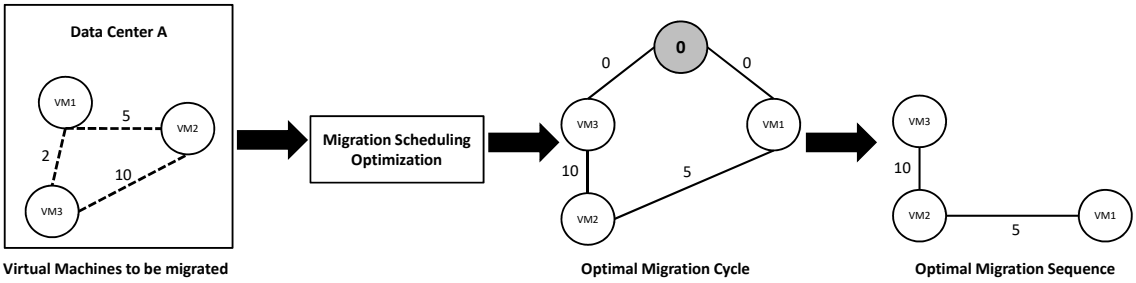


Figure 5.3: Running Example.

After replacing (5.3) in (5.1), we have:

$$z = \sum_{i \in V} \sum_{j \in V} \sum_{h \in D} d_{ij} x_i^h (1 - x_j^h) \tag{5.4}$$

From the new form of the expression (5.1) presented in equation (5.4), we deduce that the total traffic due to the migration process does not depend on the DC pairs, thus, independent from DC pairs ordering. □

Consequently, the migration scheduling problem is also independent from the DC pair ordering and can be considered by DC pairs separately. This problem can be seen as a variant of the well-known *Traveling Salesman Problem* (TSP) [122] where the aim is to find the tour of minimum total cost. In our case, the objective is to minimize the communication traffic during the migration process.

The formulation proposed in [123] is the most efficient for the TSP problem as it provides a polynomial number of constraints and have $O(n^2)$ variables.

In this formulation, we are given an undirected complete graph $G = (M, E)$, where $M = N \cup \{0\}$. N denotes the set of VMs that will be migrated and E the set of edges. Since there is no specified starting VM to migrate, we add a dummy node 0, which refers to a starting point of the migration sequence. This particular node does not exchange flow with other nodes. The optimal sequence is obtained by omitting this extra node. We denote by c_{ij} the amount of traffic exchanged between each pair of VMs that will be migrated. Figure (5.3) presents a numerical example of the exact optimization model.

In this formulation, we are given the following decision variables:

- u_i, designates the position of the VM $i \in M$ in the migration sequence.
- y_{ij}, is a binary decision defined as follows.

$$y_{ij} = \begin{cases} 1 & \text{If the link (i,j) belongs to the tour} \\ 0 & \text{Otherwise} \end{cases}$$

- We define the decision variable $w_{ij} \geqslant 0$ as the distance between VMs i and j.

The objective function can be denoted as follows:

$$\min \sum_{i \in N} \sum_{j \in N} w_{ij}.c_{ij} \tag{5.5}$$

Subject to:

$$w_{ij} \geqslant u_i - u_j \qquad \forall i, j \in M \qquad (5.6)$$

$$w_{ij} \geqslant u_j - u_i \qquad \forall i, j \in M \qquad (5.7)$$

$$\sum_{\substack{j \in M \\ i \neq j}} y_{ij} = 1 \qquad \forall i \in M \qquad (5.8)$$

$$\sum_{\substack{j \in M \\ i \neq j}} y_{ji} = 1 \qquad \forall i \in M \qquad (5.9)$$

$$u_i - u_j + n.y_{ij} + (n-2).y_{ji} \leqslant n-1 \qquad \forall i \in N, \forall j \neq i \in N \qquad (5.10)$$

$$1 + (n-2).y_{i0} + \sum_{j \in N} y_{ji} \leqslant u_i \qquad \forall i \in N \qquad (5.11)$$

$$n - (n-2).y_{0i} - \sum_{j \in N} y_{ij} \geqslant u_i \qquad \forall i \in N \qquad (5.12)$$

$$y_{ij} \in \{0, 1\} \qquad \forall i \in M, \forall j \in M$$

$$w_{ij} \geqslant 0 \qquad \forall i \in M, \forall j \in M$$

The aim of the objective function (5.5) is to minimize the overall network traffic generated during the migration of inter-communicating VMs. The constraints (5.6) and (5.7) ensure that the distance between the VMs i and j corresponds to the difference of their respective positions. The set of constraints (5.8) and (5.9) ensure that each node (VM) is migrated exactly once. As for the constraint (5.10), it ensures the elimination of the subtours and guarantees a linear arrangement representing the migration sequence. Finally, the constraints (5.11) and (5.12) eliminate tours that serve more or less than exactly n VM nodes. The order of migration is directly obtained from the solution to the problem above by eliminating the *dummy* node 0 and its arcs.

Proposition 6. The VM migration scheduling (VMMS) problem is *NP-Hard*.

Proof. The proof is based upon reduction of the VMMS problem to the well-known Traveling Salesman problem (TSP). The TSP is well known as being *NP-complete* [124]. Hence, the VM migration scheduling problem is *NP-Hard*. □

In the next section, we propose a heuristic solution to solve the VMMS problem efficiently.

5.2.3 Heuristic Solution

As proved in the previous section, the VMMS problem is *NP-Hard*. Hence, we propose the heuristic described in Algorithm (1) which depicts the transformation of the problem to the TSP. The underlying idea of the heuristic is to consider the migration process as a network flow problem where the objective function minimizes the total flow on the links. This network has a linear topology, yet to determine, that represents the scheduling of the migration process of the VMs. The load on each link is made up of the transiting traffic through it and the entering/exiting traffic from/to its end nodes. Therefore, arranging the VMs in such a manner that the heaviest flows are routed on the direct links would yield an effective solution.

Algorithm 1 VM Migration Scheduling heuristic.

Input: Number of VMs to migrate N, traffic matrix
Output: The best migration sequence
1: Initialize $u_1 = 1$
2: Solve the *TSPMax* problem
3: Get the values of the variables u_i and y_{ij} from the solution of the *TSPMax*
4: **repeat**
5: Eliminate an arc from the solution cycle. Let (i_0, j_0) be that arc
6: Compute the total flow
7: $f_{i_0 j_0} = \sum_{i \in N} \sum_{j \in N} |u_j - u_i| \times c_{ij}$
8: Reinstate the arc (i_0, j_0) to the solution
9: **until** All arcs have been removed exactly once
10: Find $f_{i_b j_b} = \inf_{(i,j)}(f_{ij})$
11: The best migration sequence is the sequence obtained by eliminating the arc (i_b, j_b).

Hence, we simply transform the TSP to a maximization problem that we refer to as *TSPMax*. This transformation has as objective to route the heaviest traffic on the direct links. Then, the heuristic consists in solving a TSP with a maximization objective function, rather than a minimization one, where the links' weights are represented by the amount of flow exchanged between their end nodes. With such an objective function, TSP is rapidly solved. Let f_{ij} be the total flow circulating between the node i and j ($i \in N$, $j \in N$).

The objective function of the *TSPMax* is denoted as follows:

$$\max \sum_{i \in N} \sum_{\substack{j \in N \\ i \neq j}} y_{ij}.c_{ij} \tag{5.13}$$

Subject to:

$$\sum_{\substack{j \in N \\ i \neq j}} y_{ij} = 1 \qquad \forall i \in N \tag{5.14}$$

$$\sum_{\substack{j \in N \\ i \neq j}} y_{ji} = 1 \qquad \forall i \in N \tag{5.15}$$

$$u_i - u_j + (n-1).y_{ij} + (n-3).y_{ji} \leqslant n-2 \qquad \forall i \neq 1 \in N, \forall j \neq 1 \in N \tag{5.16}$$

$$1 + (n-3).y_{i1} + \sum_{\substack{j \in N \\ j \neq 1}} y_{ji} \leqslant u_i \qquad \forall i \neq 1 \in N \tag{5.17}$$

$$n - 1 - (n-3).y_{1i} - \sum_{\substack{j \in N \\ j \neq 1}} y_{ij} \geqslant u_i \qquad \forall i \neq 1 \in N \tag{5.18}$$

$$y_{ij} \in \{0, 1\} \qquad \forall i \in N, \forall j \in N$$

The aim of the objective function (5.13) is to prioritize the migration of VMs that are highly correlated and are exchanging an important amount of traffic. The set of constraints (5.14) and (5.15) ensures that each node (VM) is migrated exactly once.

	Heuristic		Solver	
#VMs	Obj. fct (Mbps)	Time (s)	Obj. fct (Mbps)	Time (s)
10	12758	0,3	11218	249
20	121724	0,48	-	-
30	412040	1.8	-	-

Table 5.1: Comparison between the VMMS heuristic and the exact method.

As for the constraint (5.16), it ensures the elimination of the subtours. Finally, the constraints (5.18) and (5.17) eliminate tours that serve more or less than exactly n VM nodes.

In the next section, we present the experiment results conducted on the proposed VMMS heuristic.

5.2.4 Performance Evaluation

In order to evaluate the performance of the proposed solution, we have conducted experiments on the heuristic proposed in Section 5.2.3. In this set of tests, we have considered symmetric traffic matrices. First, we have compared the performance of the heuristic and the exact model presented in Section 5.2.2.

Table (5.1) presents the comparison results. We note that the gap between the objective function of the heuristic and the exact model is very small. In addition, the results show that the migration scheduling heuristic takes a very short time to provide the best migration sequence (less than 2 seconds). However, the exact method, solved by the ILP solver, takes more than 4 minutes for 10 VMs. Note that, the solver was not able to find the optimal solution for large size problems (20 and 30 VMs) due to a problem of memory. In fact, the exact method is very time and resource consuming due to the Branch and Bound algorithm that performs poorly because of the bad quality of the lower bound and the symmetry of the formulation.

Figure (5.4) shows the variation of the communication traffic generated during the migration sequence of different numbers of inter-communicating VMs for both symmetric and asymmetric traffic matrices. We note that the communication traffic is more important if the migrated VMs have a symmetric traffic matrix.

As for the execution time of the heuristic, the results presented in Figure (5.5) show that the execution time graphs are almost linear. However, it remains very small and does not exceed a dozen seconds.

In order to show the interest in using the VMMS heuristic, we have compared the amount of backbone traffic generated during the migration while using random migration sequence and the solution obtained by the proposed VMMS. The results are depicted in Figure (5.6). The figure shows a huge difference between the two methods. It is clear that VMMS reduces the amount of traffic generated during the migration.

In this section, we have proposed exact and heuristic solution to find the best migration sequence of inter-communicating VMs within a geo-distributed cloud infrastructure. However, we assumed that the VMs will remain in the system for the whole time. In a realistic cloud environment, some VMs have a fixed lifetime and will leave the cloud system. Thus, it is important to take this parameter into account when making the migration decisions in order to prevent from excessive or

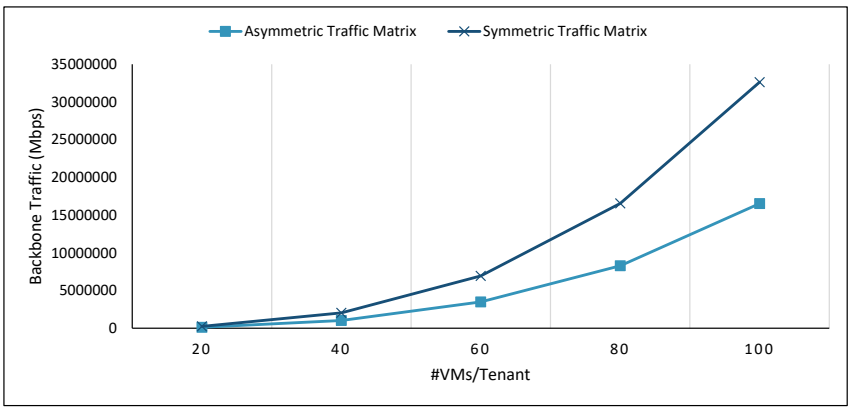
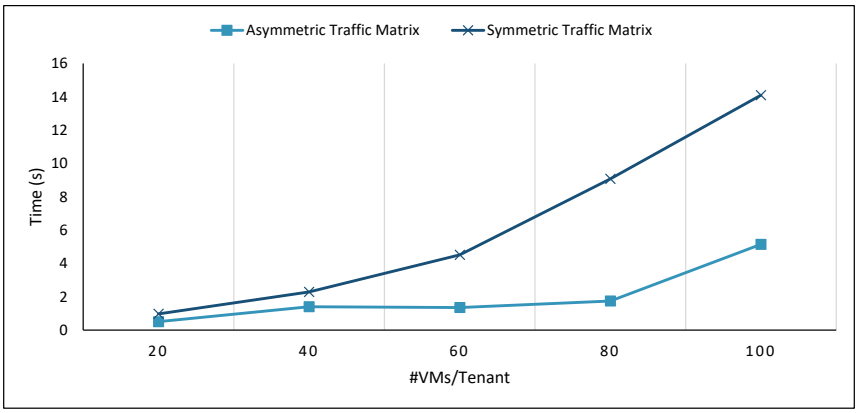


Figure 5.4: Backbone traffic versus. number of VMs per tenant.



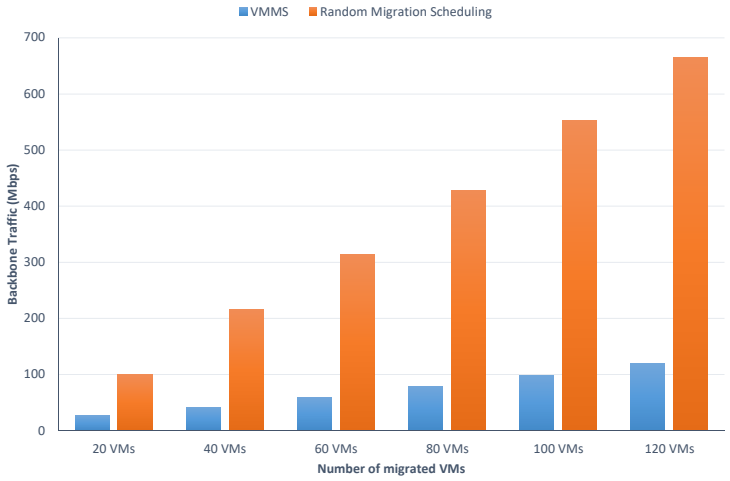


Figure 5.6: Backbone traffic vs. number of migrated VMs.

unnecessary migrations and at the same time maintain the system stability.

In the next section, we propose both exact and heuristic solutions to solve the VM scheduling problem with time-window constraints.

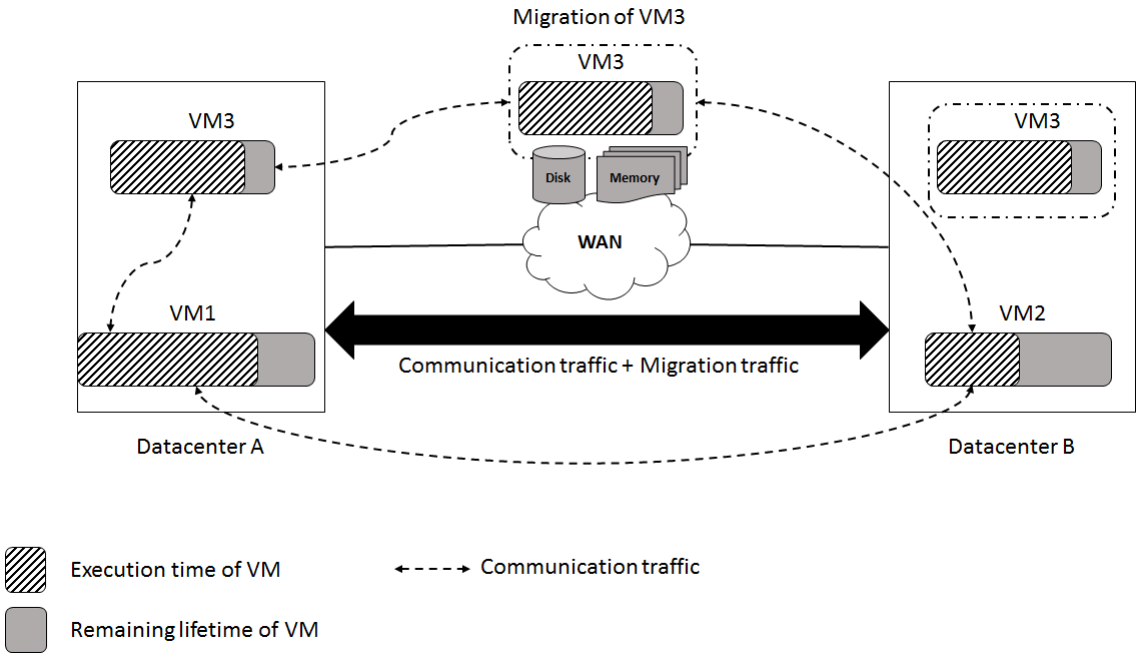


Figure 5.7: Example of VMs migration with finite lifetime.

5.3 VM Scheduling with Time-Window Constraints

In this section, we focus on the VM scheduling problem with time-window constraints.

In a dynamic cloud environment, there are new VMs arrivals and departures. The traffic pattern of different VMs is dynamic and may change over time. We assume that each VM has a fixed execution time that is known a priori. Our aim is to optimize the placement and migration decisions by reducing the number of migrations and while minimizing the inter-DCs traffic. Thus, we prevent from possible link congestion problems and maintain the system performance and stability. In fact, the system stability is proportional to the number of performed migrations [125]. A minimum number of migrations will improve the system performance [126]. During the live WAN migration of VMs, there are two types of traffic: (i) migration traffic which includes memory and disk states, (ii) as well as communication traffic produced by inter-VMs communication during the migration process as shown in Figure (5.7). Therefore, migration decisions need to take into account the VM's lifetime period in order to prevent from performing useless and costly migrations.

Few recent works have considered the fact that VMs have a finite execution period while making placement and migration decisions. In [127], the aim was to minimize the overall energy consumption of the DCs. In [128], the authors have proposed a deactivation-aware placement algorithm and a periodic defragmentation algorithm that aim to minimize the total DC cost. In [129], two VM scheduling algorithms were proposed to optimize the virtual-to-physical machine mapping. The objective was to minimize the cumulative up time in order to save energy. In [130], a formal definition of VM re-scheduling is given. However, none of aforementioned works have studied the traffic-aware VM scheduling problem within a geo-distributed DCs and none of them have considered inter-VM communication traffic.

Symbol	Description
s_i	The start time of the VM $i \in V$
e_i	The termination time of the VM $i \in V$
ε_i	The VM's life time.
τ_i	The duration of the migration of the VM i ($i \in V$)

Table 5.2: Notations.

We make the following assumptions:

- The time is divided into time slots of equal length $t \in [0..T]$.
- Each VM i has a start time s_i and a termination time e_i. The VM lifetime ε_i is fixed and considered as an input. Note that the VM's lifetime is defined as follows: $\varepsilon_i = e_i - s_i$.
- Each request may include many inter-communicating VMs. However, VMs requests are independent.
- The VM's resource requirements are assumed to be static (i.e. it does not change over time).
- We consider a sequence of migrations.
- A reserved bandwidth is allocated for the migration of different VMs.
- Migration may take several time slots.

5.3.1 Exact Solution

In this section, we formally define the traffic-aware VM scheduling problem. Table (5.2) describes the notations used in the proposed formulation.

In this formulation, we consider the following decision variables:

- φ_{ih}^t, denotes the amount of traffic originated from the VM $i \in V$ and sent from the DC $h \in D$ during the time slot $t \in T$.
- x_{ih}^t, is equal to 1 if the VM $i \in V$ is placed in the DC $h \in D$ during the time $t \in T$, 0 otherwise.
- z_{it}^{hk}, is equal to 1 if the VM $i \in V$ is migrated from the DC $h \in D$ to the DC $k \in D$ during the time $t \in T$, 0 otherwise.
- y_i^t, is equal to 1 if the VM $i \in V$ is being migrated during $t \in T$.

We denote by O_i the total traffic emanating from a VM $i \in V$.

$$O_i = \sum_{j \in V} d_{ij} \quad \forall i \in V \tag{5.19}$$

The objective of this formulation is to minimize the backbone network traffic (i.e. inter-DCs traffic). This traffic includes communication traffic and the traffic generated by the VM migration.

Let us consider $T_s = \{\forall t \in T : t \leqslant e_i \text{ and } t \geqslant s_i\}$.

$$\min \sum_{t \in T} \sum_{i \in V} \sum_{h \in D} \varphi_{ih}^t + y_i^t . M_i \tag{5.20}$$

Subject to the following constraints:

$$\varphi_{ih}^t \geqslant O_i . x_{ih}^t - \sum_{j \in V} . d_{ij} . x_{jh}^t \qquad \forall i \in V, \forall h \in D, \forall t \in T_s \tag{5.21}$$

$$\sum_{i \in V} u_{ir} . x_{ih}^t \leqslant cap_r^h \qquad \forall r \in R, \forall h \in D, \forall t \in T \tag{5.22}$$

$$\sum_{h \in D} x_{ih}^t = 1 \qquad \forall t \in T_s, \forall i \in V \tag{5.23}$$

$$x_{ih}^t \leqslant a_{ih} \qquad \forall i \in V, \forall h \in D, \forall t \in T_s \tag{5.24}$$

$$t . x_{ih}^t \leqslant e_i \qquad \forall t \in T_s, \forall i \in V, \forall h \in D \tag{5.25}$$

$$t . x_{ih}^t \leqslant x_{ih}^t . e_i \qquad \forall t \in T, \forall i \in V, \forall h \in D \tag{5.26}$$

$$x_{ih}^t . t \geqslant x_{ih}^t . s_i \qquad \forall t \in T, \forall i \in V, \forall h \in D \tag{5.27}$$

$$z_{it}^{hk} \geqslant x_{ih}^{t-1} + x_{ik}^t - 1 \qquad \forall t \in T, \forall i \in V, \forall k, h \in D, k \neq h \tag{5.28}$$

$$y_i^t \geqslant z_{it}^{hk} \qquad \forall i \in V, \forall t \in T, \forall k, h \in D \tag{5.29}$$

$$O_i . \sum_{h \in D} x_{ih}^t . (1 - \sum_{\substack{j \in V \\ i \neq j}} x_{jh}^t) . (e_i - t) \geqslant (y_i^t . M_i + O_i . \sum_{h \in D} x_{ih}^t . (1 - \sum_{j \in V i \neq j} x_{jh}^t) . \tau_i \forall i \in V, \forall t \in T_s \tag{5.30}$$

$$z_{it} \in \{0, 1\} \qquad \forall i \in V, \forall t \in T$$

$$x_{ih}^t \in \{0, 1\} \qquad \forall i \in V, \forall h \in D, \forall t \in T$$

$$y_i^t \in \{0, 1\} \qquad \forall i \in V, \forall t \in T$$

$$\varphi_{ih}^t \geqslant 0, \qquad \forall i \in V, \forall h \in D, \forall t \in T$$

The constraint (5.21) is a flow conservation constraint. The constraint (5.22) ensures that the set of VMs placed in a DC at each time slot t does not exceed the capacity of this DC. The constraint (5.23) ensures that each VM is running on only one DC at each time slot t. The constraint (5.24) restricts the placement of VMs in a particular number of DCs that satisfy a location constraint. The constraint (5.25) ensures that each VM completes its segment continually. The set of constraints (5.26) and (5.27) ensure that the VMs do not violate the deadline constraints. The set of constraints (5.28) and (5.29) permit to know if a VM is actually being migrated and determine the source and the destination of the migration. Finally, if the communication traffic generated by a VM before finishing its execution is less than the migration traffic produced during the migration of the same VM, it is obvious that no migration is needed in this case. This fact is expressed by the set of constraints (5.30).

However, the constraint (5.30) is not linear. In order to linearize it, we introduce a new decision variable w_{ij}^{ht} defined as follows:

$$w_{ij}^{ht} = x_{ih}^t . x_{jh}^t \qquad \forall i \in V, \forall j \in V, \forall h \in D, \forall t \in T_s \tag{5.31}$$

We replace the equation (5.31) in the constraint (5.30):

$$O_i.\sum_{\substack{j\in V\\ i\neq j}}\sum_{h\in D}(x_{ih}^t - w_{ij}^{ht}).(e_i - t) \geqslant y_i^t.M_i.\tau_i + O_i.\tau_i.\sum_{\substack{j\in V\\ i\neq j}}\sum_{h\in D} x_{ih}^t \ \forall i \in V, \forall t \in T_s \quad (5.32)$$

Then, we must add the following logical constraint:

$$w_{ij}^{ht} \geqslant x_{ih}^t + x_{jh}^t - 1 \qquad \forall t \in T_s, \forall h \in D, \forall i, j \in V \quad (5.33)$$

This formulation has turned out to be time and resource consuming as it presents a huge number of variables ($O|N|^4$) where N refers to the problem size. In the next section, we propose the heuristic solution.

5.3.2 Heuristic Solution

The main idea of the heuristic is to reduce the number of migrations by considering any placement algorithm which will be executed it at the beginning of each time slot in order to find the VMs that are considered candidates for the migration. It compares then the current placement scheme with the new one provided by the placement algorithm. In fact, placement algorithms produce a huge number of migrations. Excessive migrations may lead to a huge performance degradation of the cloud system [116][117].

For each time slot, the heuristic selects the arriving VMs and removes the departing ones. Then it executes the placement algorithm. We have decided to use our placement algorithm proposed in 3 as we have shown its efficiency to solve VM placement problem while minimizing the backbone traffic. However, this algorithm does not take into consideration neither the migration costs, nor the remaining lifetime of the VMs. It provides simply new VMs placement plan.

The heuristic calculates for each VM candidate, the migration traffic and the communication traffic to ensure that the selected VM is worth being migrated. It checks also if the migration of a VM does not violate the DC capacity constraints. The migration sequence has also an important impact on the amount of traffic circulating on the backbone network. Thus, the proposed heuristic sorts the list of VMs that will be migrated by decreasing size and performs the migration. The heuristic is presented in Algorithm 2.

Let us denote by:

- *ListVMs*, the list of all VM requests,
- S, the list of selected VM,
- L, the list of VMs to be migrated,
- *CandidateMig*, the list of VMs that are considered as candidate for the migration,
- C_i^t, the communication traffic originated from the VM i, $(i \in V)$,
- MC_i^t, the communication traffic originated from VM i during migration,
- M_i^t, the traffic generated by the migration of the VM i over the WAN.

Algorithm 2 VM migration scheduling algorithm with time window constraints.

Input: List of VMs *ListVMs*, traffic matrix
Output: List of VMs to Migrate L

$S \leftarrow \emptyset$
for each t in T **do**
  $CandidateMig \leftarrow \emptyset$
  $L \leftarrow \emptyset$
  for each v in *ListVMs* **do**
    if $s_i = t$ **then**
      $S \leftarrow S \cup \{i\}$
    else
      if $s_i \geq t$ and $e_i \leq t$ **then**
        $S \leftarrow S \cup \{i\}$
      else
        Remove the VM i from S
      end if
    end if
  end for
  Solve the placement problem for the VMs in S
  Record the new placement decisions in *CandidateMig*
  for each i in *CandidateMig* **do**
    if VM i is candidate for migration **then**
      Calculate the generated traffic for the VM i
      if $C_i^t \geqslant M_i^t + MC_i^t$ **then**
        if the capacity constraint on the DC source is not violated **then**
          $L \leftarrow L \cup \{i\}$
        end if
      end if
      Sort L by size descending and migrate the VMs
    end if
  end for
end for

$$C_i^t = O_i . \sum_{h \in D} x_{ih}^t . (1 - \sum_{\substack{j \in V \\ i \neq j}} x_{jh}^t) . (e_i - t) \quad \forall i \in V, \forall t \in T_s \tag{5.34}$$

$$MC_i^t = O_i . \sum_{h \in D} x_{ih}^t . (1 - \sum_{\substack{j \in V \\ i \neq j}} x_{jh}^t) . \tau_i \quad \forall i \in V, \forall t \in T_s \tag{5.35}$$

$$M_i^t = y_i^t . M_i . \tau_i \quad \forall i \in V, \forall t \in T_s \tag{5.36}$$

In the next sections, we provide the obtained experiment results.

5.3.3 Quality of the heuristic solution

We have fixed the number of DCs to six. The VM's start-time and end-time were generated randomly. We have executed the algorithms during one day. The results

# VMs	Gap (G %)	
	20 VMs/Tenant	**80 VMs/Tenants**
1000	12, 32	12
2000	13	12, 45
3000	13, 57	12, 22
4000	12, 8	12, 5

Table 5.3: Average optimality gap.

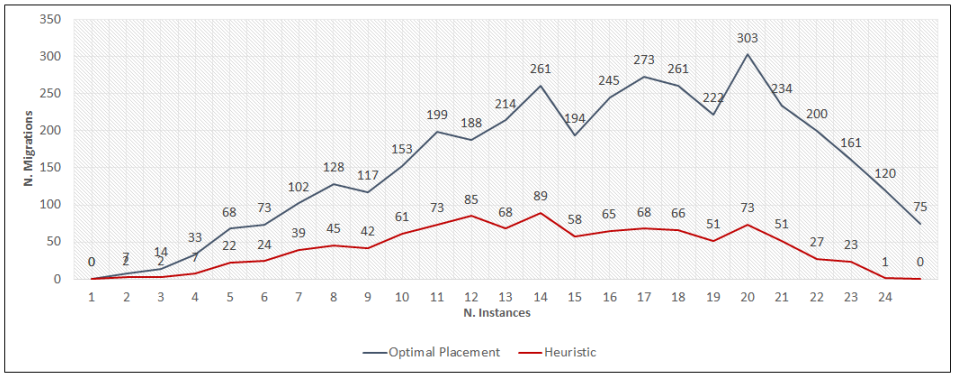


Figure 5.8: Number of migrations per instance for 2000 VMs/20 VMs per tenant.

were taken at the beginning of each hour. The values of the traffic matrices were generated randomly and ranged between 0 to 100 Mbps.

To evaluate the quality of the heuristic, we have compared the solution provided by the heuristic with the solution provided by the placement algorithm presented in [65]. Let us denote by:

- S_h, the heuristic solution,
- S^*, the optimal solution of [65],
- G, the optimality gap (%) defined as follows:

$$G = \frac{S_h - S^*}{S_h} \times 100 \tag{5.37}$$

Table 5.3, presents the average optimality gap G between the two approaches. We note that the gap does not exceed $13,57\%$. It means that the heuristic placement solution is very close to the solution provided by the placement algorithm. However, in contrast to the placement algorithm, the heuristic considers the VM's lifetime and the migration cost while making placement decisions.

5.3.4 System Stability

In order to show the impact of the number of migrations on the system stability, we have compared the number of migrations of the heuristic with those obtained by the optimal placement algorithm presented in [65]. The Figures (5.8), (5.9), (5.10) and (5.11), show the variation of the number of migrations performed by both the

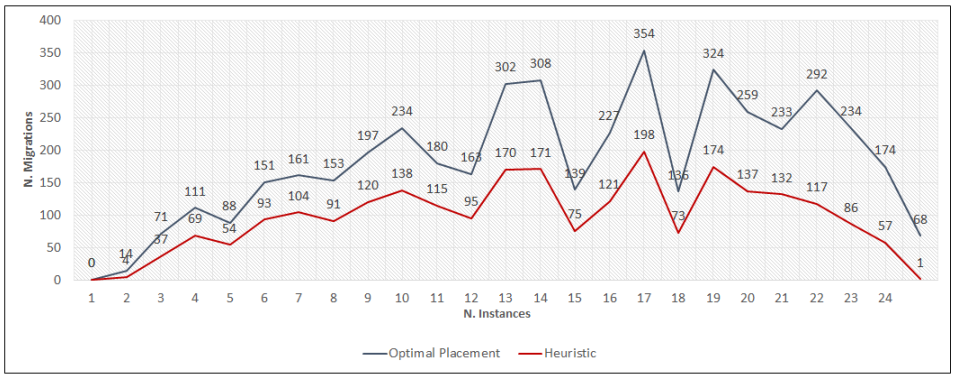


Figure 5.9: Number of migrations per instance for 2000 VMs/80 VMs per tenant.

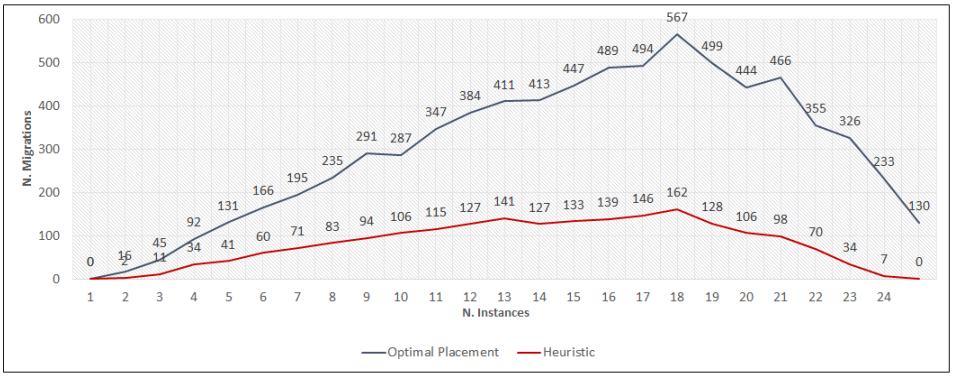


Figure 5.10: Number of migrations per instance for 4000 VMs/20 VMs per tenant.

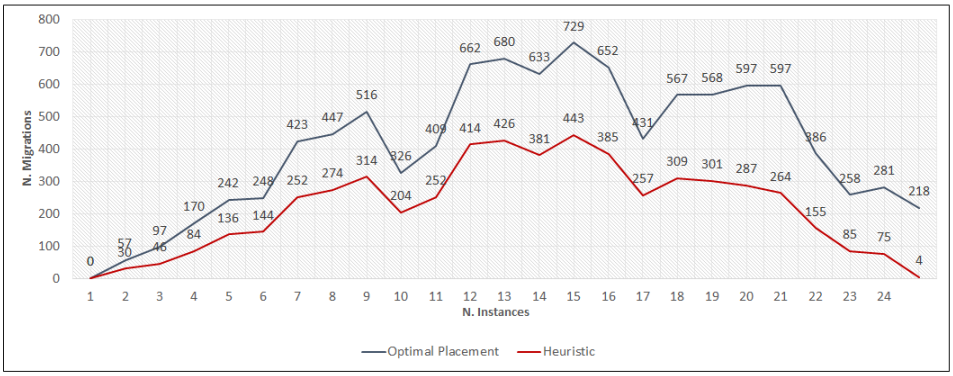


Figure 5.11: Number of migrations per instance for 4000 VMs/80 VMs per tenant.

VM placement model of [65] and the VM scheduling heuristic for 24 hours and for a total number of VMs varying from 2000 to 4000 VMs.

We note that the number of migrations produced by the placement algorithm is huge compared to the number of migrations obtained by the heuristic. We note also that the number of performed migrations increases with the number of VMs. We can conclude that the consideration of the VM's lifetime in the migration decision process helps to improve the system stability and prevent from performing excessive migrations.

5.4 Conclusion

In this chapter, we have focused on the problem of VM migration scheduling in geo-distributed DCs. We have proposed first, exact and heuristic solutions to find the best migration sequence of inter-communication VMs. Then, we have studied the impact of time constraints in the migration decisions. We have considered that VMs have a fixed execution time and we proposed exact as well as heuristic solution to solve the problem. Experiment results show the effectiveness of our approach. In the next chapter, we will focus on the problem of stochastic VM placement within a geo-distributed cloud infrastructure.

Chapter 6

Proactive VM Placement Problem for Risk Management

Contents

6.1 Introduction

In this chapter, we tackle the network-aware stochastic version of the VM placement problem in geo-distributed DCs. Due to the existence of highly non-uniform inter-VMs communication traffic, it is impossible to have an accurate estimation of the expected traffic volume within the backbone network. Hence, in this chapter, we propose a proactive stochastic optimization model which ensures the minimization of the overloading risk of the DC edge routers.

6.2 Problem Description

In a geo-distributed cloud infrastructure, network congestion is a crucial issue. The increasing amounts of traffic generated by the traffic-intensive applications hosted in the VMs may cause bottlenecks in the network resulting in performance degradation of the whole system. Hence, VM placement plan must be optimized in order to prevent from possible SLAs violations in the future. In particular, it is important to minimize the expected traffic circulating within the backbone network (i.e. Inter-DCs traffic) which is the aim of this work.

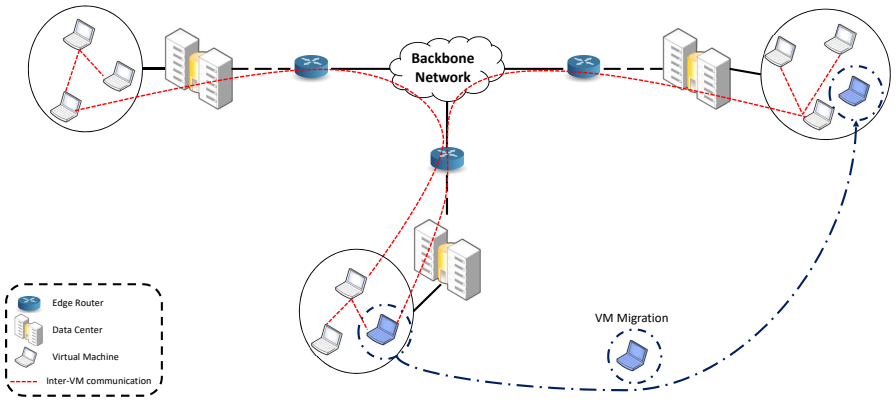


Figure 6.1: System Model.

.

To tackle this problem, VM migration techniques are commonly used in order to optimize the configuration of the cloud system. In fact, VM migration is used as a tool to cope with the demand fluctuations and the dynamic aspects of traffic patterns. As a matter of fact, VM migration brings with it many benefits; (1) it provides flexibility in the management of a DC, and (2) it enables moving VMs across DCs in order to adjust and optimize the cloud infrastructure [7]. However, the reconfiguration of the cloud system using VM migration rises many challenges including:

- The DC in which VMs will be migrated to, must have enough resource capacity in order to host these VMs.
- The overhead of VM migration, which consists in the amount of data transferred during the migration process, must be minimized.
- Due to the dynamic changes in the application's workload, it is not efficient to make migration decisions based only on the current state of the system. Accurate traffic prediction is necessary, however, it is a very difficult task [15].
- Excessive migrations may lead to a huge performance degradation of the cloud system [116, 117].

Recent studies [15, 16, 17] have shown that the workload of VMs is highly dynamic and bursty which may cause the existent placement and migration schemes to be inefficient. These applications are characterized by highly non-uniform traffic pattern. In this work, we consider that the workload consists in inter-VMs communication traffic. In addition, most of the existent works [18, 19, 20] make migration decisions based on deterministic demand estimation and workload characterization without considering stochastic properties.

As shown in Figure (6.1), we consider a geo-distributed cloud environment where DCs are connected within a backbone network.

According to [25], DC edge routers are responsible for connecting the DCs to WAN. The DC edge router has uplinks for the data transfer up to the WAN and downlinks for receiving data from WAN. These links have fixed bandwidth capacities. In our work, we focus on minimizing the traffic sent to the backbone network.

Hence, we consider only the bandwidth capacity of the uplinks. In fact, by minimizing the traffic sent from one DCs to other DCs over the network, we implicitly minimize the traffic received by other DCs.

In the rest, we refer to the bandwidth capacity of uplinks by the bandwidth capacity of the edge router.

In this work, VM migration is used as a tool to cope with the fluctuation of inter-VMs bandwidth requirements as well as the variance of this demand in the future.

In such an environment, the cloud provider has no knowledge about the inter-VMs' bandwidth demand. This parameter is considered as uncertain. The traffic matrix represents communication traffic between each pair of VMs.

In this work, we make the following assumptions:

- The entire infrastructure is owned and managed by the same IaaS provider.
- Each VM is characterized by its hardware configuration in terms of CPU, RAM and Storage.
- Each DC is characterized by its capacity in terms of hardware resources CPU, RAM, and Storage.
- Each VM may have a location constraint. Thus, it can be only placed in a defined set of DCs.
- There are multiple independent clients submitting requests to provision VMs that may be heterogeneous.

6.3 Problem Formulation

In this section, we first present the problem as a Stochastic Integer Program. Then, we present, an equivalent optimization formulation to solve it as an ILP.

Let us denote by E_h the bandwidth capacity of the edge router of the DC $h \in D$. In the following, we use the decision variables defined bellow.

- φ_i^h, defines the amount of traffic originated from the VM $i \in V$ and sent from the DC $h \in D$ (i.e. the traffic sent to the backbone network).
- x_i^h, is equal to 1 if the VM $i \in V$ is placed in the DC $h \in D$, 0 otherwise.
- z_i^h, is equal to 1 if the VM i is migrated from the DC $h \in D$.
- f_{hk}^i, which denotes the amount of traffic originated from the VM $i \in V$ and circulating between the DCs $h \in D$ and $k \in D$.

6.3.1 Stochastic Optimization Model

In this section, we formally define the problem as a SIP. The purpose of stochastic programming is to find an optimal solution with giving uncertainty in some parameters. In this paper, we allow uncertainty of the inter-VMs communication traffic. Previous studies [15, 16, 17], have shown that the VM workload is bursty.

Network-aware stochastic VM placement problem rises new challenging problems including:

- How to estimate stochastic inter-VM bandwidth resource demand?
- How to detect bottlenecks in the DC edge router?
- How to make VMs migration decisions while ensuring network and DC capacity as well as proximity location constraints?

In this formulation, we consider a random variable $\widetilde{d_{ij}}$ that describes the amount of traffic exchanged between each pair of VMs (i.e. inter-VMs bandwidth demand). The variable follows a probability distribution that can be estimated from runtime measurement. We assume that the distribution can be obtained using statistical process to analyze historical data. Many studies [85, 131, 15] have shown that the resource demand of VMs follows a *Normal* distribution $\mathcal{N}(\mu, \sigma^2)$. Thus, we consider that the inter-VMs bandwidth demand follows also the *Normal distribution*.

Our aim is to minimize the amount of traffic circulating between the different DCs. This traffic includes by the communication traffic (i.e. inter-VMs communication) and the migration traffic (i.e. the amount of data transferred during the migration process).

The objective function is defined as follows:

$$\min \sum_{i \in V} \sum_{h \in D} M_i.z_i^h + \varphi_i^h \tag{6.1}$$

Subject to the following constraints:

$$\varphi_i^h \geqslant \sum_{j \in V} \widetilde{d_{ij}}.x_i^h - \sum_{j \in V} x_j^h.\widetilde{d_{ij}} \qquad \forall i \in V, \forall h \in D \tag{6.2}$$

$$\sum_{k \in D} f_{hk}^i - \sum_{k \in D} f_{kh}^i = \sum_{j \in V} \widetilde{d_{ij}} x_i^h - \sum_{j \in V} \widetilde{d_{ij}}.x_j^h \qquad \forall i \in V, \forall h \in D \tag{6.3}$$

$$Pr(\sum_{i \in V} M_i.z_i^h + \sum_{i \in V} \varphi_i^h \leqslant E_h) \geqslant 1 - \epsilon \qquad \forall h \in D \tag{6.4}$$

$$\sum_{h \in D} x_i^h = 1 \qquad \forall i \in V \tag{6.5}$$

$$x_i^h \leqslant a_i^h \qquad \forall i \in V, \forall h \in D \tag{6.6}$$

$$\sum_{i \in V} u_{ir}.x_i^h \leqslant cap_r^h \qquad \forall r \in R, \forall h \in D \tag{6.7}$$

$$z_i^h \geqslant x_{0i}^h - x_i^h \qquad \forall i \in V, h \in D \tag{6.8}$$

$$z_i^h \in \{0.1\} \qquad \forall i \in V, h, k \in D$$

$$x_i^k \in \{0.1\} \qquad \forall i \in V, k \in D$$

$$\varphi_i^h \geqslant 0 \qquad \forall i \in V, h, k \in D$$

$$f_{hk}^i \geqslant 0 \qquad \forall i \in V, h, k \in D$$

The constraint (6.2) and (6.3) are both flow conservation constraints. They are both stochastic due to the random variable $\widetilde{d_{ij}}$ which refers to the inter-VMs communication traffic.

The DC edge router, which ensures the connection between the DC and the backbone network, has a fixed bandwidth capacity.

The constraint (6.4) ensures that for each DC edge router, the total traffic, which includes the inter-VMs communication traffic and the migration traffic, does not exceed the bandwidth capacity of the edge router with a high probability $(1-\epsilon)$. A a matter of fact, ε is a QoS metric called overloading probability. The constraint (6.4) ensures the service quality guarantee with a threshold ε and it minimizes the risk of overloading the router in the future.

The constraint (6.5) is a demand satisfaction constraint. It ensures that every VM is running on only one DC. As for the constraint (6.6) it stipulates that VM $i \in V$ cannot be assigned to all DCs. It restricts the placement of VMs in a particular number of DCs that satisfy for example proximity to end-users, technology constraint, etc. The constraint (6.7) represents the capacity constraint on the DCs. It ensures that the amount of resources consumed by different VMs placed in a given DC does not exceed the resource capacities of the DC. The constraint (6.8), ensures that only already existing VMs can be considered as candidates for the migration.

The above presented model is a stochastic optimization program. Suppose that it has finite support, hence, we can enumerate the set of all different uncertainty scenarios. We can then formulate an equivalent deterministic optimization problem that can be solved as a Mixed Integer Linear Program (MILP). However, the size of the problem space can grow very large as the number of scenarios increases. Therefore, in the next section, we propose an alternative solution to formulate the problem as a MILP by applying sampling-based methods.

6.3.2 Equivalent Optimization Formulation

In order to solve the stochastic problem, we propose an equivalent formulation using sampling methods.

Let us consider the function $g(.)$ defined as follows.

$$g(x) = \sum_{j \in V} \widetilde{d_{ij}}.x_i^h - \sum_{j \in V} x_j^h.\widetilde{d_{ij}} \qquad \forall i \in V, \forall h \in D \tag{6.9}$$

In such situations, it is clearly impossible to enumerate all the possible outcomes. Hence, sampling techniques are a commonly used tool. In order to discretize the stochastic function $g(.)$, we apply *Sample Average Approximation* (SAA) method [132].

In fact, sampling-based methods have been successfully used in many different fields of stochastic optimization, such as, applications of vehicle routing, engineering design, supply chain network design, machine learning etc [55]. The appeal of sampling-based methods results from the fact that they often approximate well, with a small number of samples, problems that have a very large number of scenarios [133].

In this work, we use Monte Carlo methods [134] to generate samples of $N = \{1, .., n\}$ replications of the random variable $\widetilde{d_{ij}}$ using the *Normal distribution* $\mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$, where μ_{ij} is the mean and σ_{ij}^2 is the variance.

Let us consider the function $g_n(.)$, presented in (6.10), as the discretization of the stochastic function $g(.)$ by applying SAA methods.

$$g_n(x) = \frac{1}{n} \sum_{i=1}^{n} g(x, \xi_i) \tag{6.10}$$

Where ξ_i is a random element such that: $\widetilde{d_{ij}} = \frac{1}{n}\sum_{k=1}^{n}\xi_{ij}^{k}$ and n is the number of iterations.

Hence, the equivalent deterministic constraint of (6.2) is obtained by replacing $g(x)$ by $g_n(x)$.

$$g_n(x) = \sum_{j\in V}(\frac{1}{n}\sum_{k=1}^{n}\xi_{ij}^{k}).x_i^h - \sum_{j\in V}(\frac{1}{n}\sum_{k=1}^{n}\xi_{ij}^{k}).x_j^h \qquad \forall i \in V, \forall h \in D \tag{6.11}$$

Thus, the constraint (6.2) becomes:

$$\varphi_i^h \geqslant g_n(x) \qquad \forall i \in V, \forall h \in D \tag{6.12}$$

As mentioned above, we consider that the inter-VMs bandwidth demand $\widetilde{d_{ij}}$ follows the *Normal distribution* $\mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$. Hence, we can estimate $\widetilde{d_{ij}}$ by its mean μ_{ij} ($\widetilde{d_{ij}} \simeq \mu_{ij}$).

Let us consider the following equation:

$$\varphi_i^h = \sum_{k\in D} f_{hk}^i \qquad \forall h \in D, \forall i \in V \tag{6.13}$$

The value of f_{hk}^i can be obtained from the folw conservation constraint (6.3) as follows:

$$\sum_{k\in D} f_{hk}^i = \sum_{k\in D} f_{kh}^i + \sum_{j\in V}\widetilde{d_{ij}}x_i^h - \sum_{j\in V}\widetilde{d_{ij}}.x_j^h \qquad \forall i \in V, \forall h \in D \tag{6.14}$$

If we replace (6.13) in (6.4), we obtain:

$$Pr(\sum_{i\in V} M_i.z_i^h + \sum_{i\in V}(\sum_{k\in D} f_{kh}^i + \sum_{j\in V}\widetilde{d_{ij}}x_i^h - \sum_{j\in V}\widetilde{d_{ij}}.x_j^h) \leqslant E_h) \geqslant 1-\epsilon \quad \forall h \in D \tag{6.15}$$

Since, $\widetilde{d_{ij}}$ follows the Normal distribution $\mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$, then, because we assume that the traffic of each pair of VM $(i,j) \in V$, is independent if $i \neq j$, the aggregate traffic demand $\sum_{i\in V}\sum_{j\in V}\widetilde{d_{ij}}$ follows the Normal distribution $\mathcal{N}(\sum_{i\in V}\sum_{j\in V}\mu_{ij}, \sum_{i\in V}\sum_{j\in V}\sigma_{ij}^2)$ according to the property of normal distribution and Central Limit Theorem (CLT) [135].

Note that, the term $\sum_{i\in V} M_i.z_i^h$ is deterministic, thus, it does not follow a probability distribution. Let us denote by

$$\alpha^h = \sum_{i\in V}\sum_{j\in V}\widetilde{d_{ij}}x_i^h - \sum_{j\in V}\widetilde{d_{ij}}.x_j^h \tag{6.16}$$

We need to estimate the *Normal* distribution parameters μ_{α^h} and $\sigma_{\alpha^h}^2$.

Since, $x_i^h \in \{0,1\}, \forall i \in V, h \in D$, and because we assume that the traffic of each pair of VM $(i,j) \in V$, is independent if $i \neq j$, then, by applying SAA methods, we can estimate μ_{α^h} and $\sigma_{\alpha^h}^2$ as follows:

$$\mu_{\alpha^h} = \sum_{i\in V}\sum_{j\in V}\mu_{ij}x_i^h - \sum_{i\in V}\sum_{j\in V}\mu_{ij}.x_j^h \qquad \forall h \in D \tag{6.17}$$

$$\sigma_{\alpha^h}^2 = \sum_{i\in V}\sum_{j\in V}\sigma_{ij}^2 x_i^h + \sum_{i\in V}\sum_{j\in V}\sigma_{ij}^2.x_j^h \qquad \forall h \in D \tag{6.18}$$

Hence, it easy to show that the constraint (6.15) is equal to the overloading probability constraint presented in (6.19), where $\phi^{-1}(.)$ is the inverse of the cumulative distribution function of the *Standard Normal* distribution. In this work, we consider that $\epsilon \leqslant 0.5$ and $\phi^{-1}(1-\epsilon) \geqslant 0$.

$$\frac{E_h - \sum_{i\in V}\sum_{k\in D} f_{kh}^i + \sum_{i\in V} M_i.z_i^h + \sum_{i\in V}\sum_{j\in V}\mu_{ij}x_i^h - \sum_{i\in V}\sum_{j\in V}\mu_{ij}.x_j^h}{\sqrt{\sum_{i\in V}\sum_{j\in V}\sigma_{ij}^2 x_i^h + \sum_{i\in V}\sum_{j\in V}\sigma_{ij}^2.x_j^h}} \geqslant \phi^{-1}(1-\epsilon) \tag{6.19}$$

The deterministic equivalent formulation is presented as follows. The constraints (6.21) and (6.22), are flow conservation constraints. As for the constraint (6.23), it ensures that the amount of resource consumed by all the VMs placed in a DC, must not exceed the capacity of the DC in term of resources $r \in R$. The constraint (6.24) denotes the network overloading probability constraint for each DC edge router. The constraint (6.25) ensures that each VM is running on only one DC. The constraint (6.26), stipulates that VM cannot be assigned to all DCs. The constraint (6.27) is a migration constraints.

$$\min \sum_{i\in V}\sum_{h\in D} M_i.z_i^h + \varphi_i^h \tag{6.20}$$

Subject to the following constraints:

$$\varphi_i^h \geqslant \sum_{j\in V}\mu_{ij}.x_i^h - \sum_{j\in V} x_j^h.\mu_{ij} \qquad \forall i \in V, \forall h \in D \quad (6.21)$$

$$\sum_{k\in D} f_{hk}^i - \sum_{k\in D} f_{kh}^i = \sum_{j\in V}\mu_{ij}.x_i^h - \sum_{j\in V}\mu_{ij}.x_j^h \qquad \forall i \in V, \forall h \in D \quad (6.22)$$

$$\sum_{i\in V} u_{ir}.x_i^h \leqslant cap_r^h \qquad \forall r \in R, \forall h \in D \quad (6.23)$$

$$E_h \geqslant \sum_{i\in V} M_i.z_i^h + \sum_{i\in V}\sum_{k\in D} f_{kh}^i + \mu_{\alpha^h} + \phi^{-1}(1-\epsilon).\sqrt{\sigma_{\alpha^h}^2} \qquad \forall h \in D \quad (6.24)$$

$$\sum_{h\in D} x_i^h = 1 \qquad \forall i \in V \quad (6.25)$$

$$x_i^h \leqslant a_i^h \qquad \forall i \in V, \forall h \in D \quad (6.26)$$

$$z_i^h \geqslant x_{0i}^h - x_i^h \qquad \forall i \in V, h \in D \quad (6.27)$$

$$z_i^h \in \{0.1\} \qquad \forall i \in V, h \in D$$

$$x_i^k \in \{0.1\} \qquad \forall i \in V, k \in D$$

$$\varphi_i^h \geqslant 0 \qquad \forall i \in V, h, k \in D$$

$$f_{hk}^i \geqslant 0 \qquad \forall i \in V, h, k \in D$$

SIP aims at taking into consideration the probabilistic information in the mathematical programs. One of the well-known approaches is *Chance-constrained programming* where the aim is to find the best feasible solution for a given probability tolerance which we refer to in this formulation by ε.

If we consider a finite number of scenarios, a chance-constrained program can be equivalently written as an integer linear program as proposed in the formulation above. However, one of the risen challenges is that the obtained equivalent model

is non-linear due to the constraint (6.24). In fact, when dealing with combinatorial problems, we are lead to very hard integer non-linear programs [136].

The linearization of the constraint (6.24) leads to a very large number of variables which will impact the efficiency of the formulation and will enlarge the research space. Unfortunately, the application of this method seems restricted to small-size problems [136].

To cope with this problem, we propose, in the next section, to adopt an iterative two-step approach to solve the SIP model.

6.3.3 Network-aware Stochastic VM Placement Algorithm

In this section, we propose a heuristic to solve the network-aware stochastic VM placement in geo-distributed DCs. It is presented in Algorithm 3.

Let us denote by:

$$\beta^h = \phi^{-1}(1-\epsilon).\sqrt{\sigma^2_{\alpha^h}} \tag{6.28}$$

$$\gamma^h = \sum_{i \in V} M_i.z_i^h + \sum_{i \in V} \sum_{k \in D} f_{kh}^i + \mu_{\alpha^h} \tag{6.29}$$

We denote by $SIP_{\gamma+\beta}$,the optimization program presented in Section (6.3.2). We define SIP_γ, the optimization program where the constraint (6.24) is replaced by the following constraint.

$$E_h \geqslant \gamma^h \qquad \forall h \in D \tag{6.30}$$

Let us define the list $PrefList$ which contains the different values of ε. $PrefList$ is sorted by increasing value of ε. Smaller ε means that the risk of overloading must be very low.

The algorithm tries to solve the stochastic optimization problem within two iterations. In the first iteration, it solves the SIP_γ model without considering the non linear term β^h. In fact, the optimization program SIP_γ provides a deterministic VM placement scheme as it does not consider the variance of inter-VMs communication traffic in the future. In addition, the term γ ensures that the overall traffic sent via the DC edge router does not exceed its capacity.

Afterword, the algorithm evaluates the term β^h by considering the solution provided by SIP_γ. In the second iteration, it tries to solve the $SIP_{\gamma+\beta}$ model by adding the term β^h to the overloading probability constraint. If the $SIP_{\gamma+\beta}$ model is feasible, then the new placement scheme is the solution provided by $SIP_{\gamma+\beta}$, otherwise, we relax the value of ε and try to solve it again.

The new placement scheme is either the solution provided by $SIP_{\gamma+\beta}$, if it exists, or the solution provided by SIP_γ with an SLA violation due to the high risk of network overload.

Stochastic optimization is a simulation of different scenarios that can happen in a realistic environment. The proposed algorithm helps Cloud managers to know in advance, when SLA violations can take place and what is the threshold of service guarantee. This can be helpful for dimensioning edge router bandwidth capacity in order to handle dynamic bandwidth provisioning and prevent from possible network overloading problems in the future.

The approach used in the proposed algorithm can be considered as iterative. The quality of the solution can be improved by re-injecting the solution provided

Algorithm 3 Network-aware Stochastic VM Placement Algorithm.

Input: Initial Placement scheme, stochastic traffic matrix, $PrefList$
Output: New VMs Placement Plan

1: $\varepsilon \leftarrow PrefList[0]$
2: Solve SIP_{γ}
3: Record solutions
4: Calculate β^h with the solutions provided by SIP_{γ}
5: Solve $SIP_{\gamma+\beta}$
6: **if** $SIP_{\gamma+\beta}$ is feasible **then**
7: New Placement Plan $\leftarrow$ Solutions of $SIP_{\gamma+\beta}$
8: **else**
9: $i \leftarrow i+1$
10: **end if**
11: **while** $SIP_{\gamma+\beta}$ is not feasible **and** $i \leqslant PrefList.size()$ **do**
12: $\varepsilon \leftarrow PrefList[i]$
13: Check feasibility of $SIP_{\gamma+\beta}$
14: $i \leftarrow i+1$
15: **end while**
16: **if** $SIP_{\gamma+\beta}$ is feasible **then**
17: New Placement Plan $\leftarrow$ Solutions of $SIP_{\gamma+\beta}$
18: **else**
19: New Placement Plan $\leftarrow$ Solutions of SIP_{γ}
20: Add SLA violation
21: **end if**

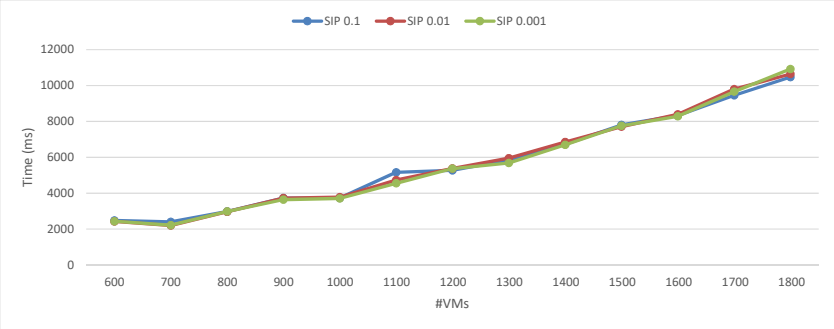


Figure 6.2: Execution Time versus the total number of VMs.

by $SIP_{\gamma+\beta}$ at line 7 and re-solving the program. However, we have noticed in the experiments that there is no significant difference between the provided solutions. Hence, for the sake of presentation and simplicity, we adopt a two-step approach as described in Algorithm (3).

In the next section, we present the experiment results showing the effectiveness of the proposed approach.

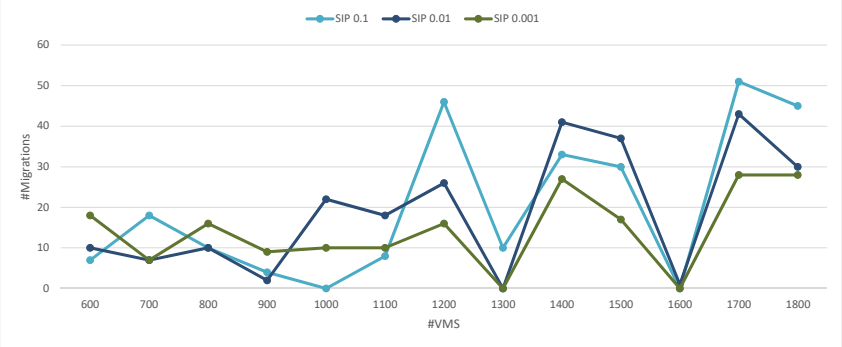


Figure 6.3: Variation of the number of migrations for $\varepsilon \in \{0.1, 0.01, 0.001\}$ with loose bandwidth capacity.

6.4 Performance Evaluation

In this section, we present the parameter settings as well as the numerical results of the conducted experiments.

The different experiments were carried out on a machine that has an Intel Xeon 3; 3 GHz CPU and 8GB of RAM. We have used the commercial solver CPLEX 12.5 [92] to solve and evaluate the different MILP formulations. In all tests, we have considered a complete graph representing the network topology. Without loss of generality, we assume that all DCs have the same hardware capacities. We consider that the servers are housed in racks. Every server has 8 cores and 16 GB of RAM. We consider that each rack hosts 30 servers and each DC has an average of 500 racks. In all experiments, we have fixed the number of DCs to six.

We generated for each pair of VMs traffic a sample of 10000 replications according to Normal distribution. Then, we applied SAA to approximate the values of the traffic matrix. We randomly generated groups of (mean, variance range) for the inter-VM communication traffic and set each pair of VMs traffic to a value generated by a randomly chosen group. At the beginning, the VMs are allocated randomly to the different DCs while insuring only DC capacity and proximity location constraints.

We assume that client's demands are independent. For simplicity of illustration, we assume that the number of required VMs for each client is the same in a given realization. In addition, we assume that the VMs belonging to the same client are exchanging data and have inter-communication traffic.

We have implemented the deterministic equivalent model in Java with the above listed parameters and have solved it using CPLEX [92]. At the beginning, the VMs are allocated randomly to the different DCs according to their location constraints. Then, the optimization model is executed at the beginning of each time slot.

We studied first the performance of the equivalent deterministic optimization model in terms of running time. We have varied the value of ϵ and we plotted the execution time of the model. The results are depicted in Figure (6.2). We note that the value of ε has no considerable impact on the execution time of the model. We can also note that the execution time does not exceed 10 sec for a total number of VM $|V| = 1800$. We can conclude that the proposed algorithm is efficient in terms of execution time.

Figure 6.4: Variation of the number of migrations for tight bandwidth capacity.

The value of the parameter ε controls the overloading probability constraint presented in the inequality (6.24). In addition, it also affects the bandwidth utilization. Smaller ε requires the system to reserve more bandwidth in order to accommodate the possible variance of Inter-VM communication demands. To ensure the non violation of the overloading probability with smaller ε, some VMs may have to be migrated to another DC. We have varied the value of ϵ and we plotted the number of migrations performed for each value. In order to study the impact of the different values of the level of service quality ε, we have considered that the bandwidth capacity of the DC edge router are large enough to satisfy the bandwidth demand for all values of ε. The results are depicted in Figure (6.3).

We note that smaller ε produces less VM migration. This can be explained by the fact that smaller ε means that the risk of network overloading is very small. Since, migration produces additional traffic, $SIP_{\gamma+\beta}$ tries to minimize the number of migration in order to prevent from extensive migrations and reduce the probability of network overload in the future. We can also say that the total number of migrations produced by $SIP_{0.001}$ is less than the number of migration of $SIP_{0.01}$ and $SIP_{0.1}$ respectively (i.e. the number of migrations $SIP_{0.001} \leqslant SIP_{0.01} \leqslant SIP_{0.1}$).

However, when the DC edge router bandwidth capacity is very tight, we note that the number of migration for the $SIP_{\gamma+\beta}$ model increases comparing to SIP_{γ} when the total number of inter-communicating VMs increases. In fact, to ensure the overloading probabilistic service guarantee with smaller ε and tight bandwidth capacity, some VMs need to be migrated. As a matter of fact, VM migration tries to place high-communicating VMs within the same DC in order to reserve more bandwidth to accommodate future demand. Figure (6.4) illustrates the experiment results performed on $SIP_{\gamma+\beta}$, where $\varepsilon \in \{0.1, 0.01, 0.001\}$, and SIP_{γ} respectively.

The SIP_{γ} model provides deterministic schemes and do not consider the variance of the inter-VM communication traffic. In contrast, the $SIP_{\gamma+\beta}$ model is able to keep a long-term state while triggering smaller number of migrations. Moreover, we can say that the $SIP_{\gamma+\beta}$ model proactively avoids the overload of DC edge router in the future. As a conclusion, even in the worst case scenario, the algorithm provides a feasible solution without exceeding the bandwidth capacity of the router. In the best case scenario, the algorithm provides the best solution with the highest level of service quality.

In order to show the quality of the solution provided by our approach, we have

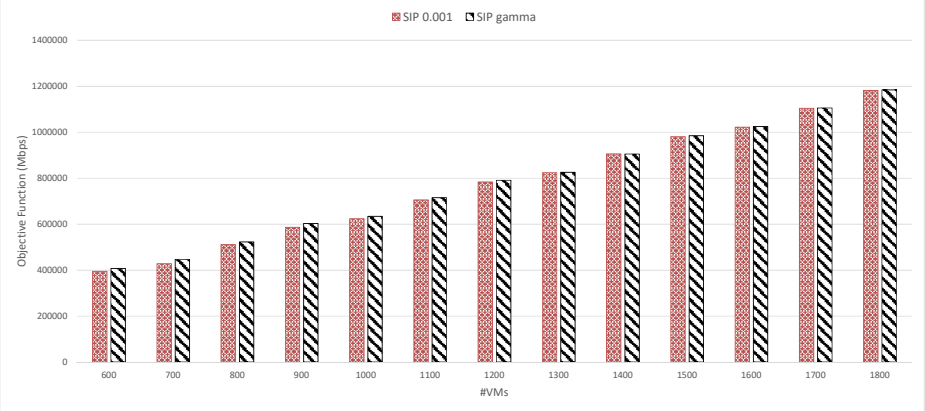


Figure 6.5: Inter-DCs traffic Vs the number of VMs.

compared the SIP_{γ}, which provides a deterministic placement scheme for the different inter-communicating VMs, and $SIP_{\gamma+\beta}$ in terms of minimizing the inter-DCs traffic. Thus, we plotted the objective function provided by both models and we have varied the number of VMs. The results are depicted in Figure (6.5). In fact, the objective function presents the overall traffic circulating between each pair of DCs (i.e. within the backbone network). We note that $SIP_{\gamma+\beta}$ provides better solution than SIP_{γ} as it minimizes the inter-DCs traffic.

6.5 Conclusion

In this chapter, we proposed a Stochastic Integer Programming formulation that aim to solve the VM placement problem in geo-distributed DCs while minimizing the risk of network overload. We considered the uncertainty of inter-VMs communication traffic. Our objective was to minimize the expected overall traffic circulating in the backbone network in order to prevent from congestion problems and maintain the QoS in the future. In order to solve the problem, we proposed an equivalent optimization model based on well-known sampling methods as well as an efficient algorithm and we used the commercial solver CPLEX to solve the proposed optimization models. The results of the conducted experiments show the effectiveness of our proposed approach.

Chapter 7

Conclusion

Contents

7.1 Contributions

Managing a geo-distributed cloud infrastructure is a very complex task. One of the key challenges faced by cloud providers, is to find optimal placement and migration scheme for the different VMs in the system. In this thesis, we addressed several complex problems related to the placement of VMs in geo-distributed cloud environment. To solve these problems, we proposed an autonomic DC management tool based on optimization approaches.

In particular, we proposed network-aware placement and migration strategies that have the objective of minimizing the traffic volume among the backbone network (i.e. inter-DCs traffic). In addition, we proposed inter-DCs migration scheduling policies for inter-communicating VMs aiming at reducing data transfer during the migration process.

To deal with the above mentioned problems, we considered two types of inter-VM communication traffic patterns. The first one tends to be stable and could be estimated accurately. For this reason, we propose a deterministic optimization models to find the optimal placement and migration scheme for the different VMs in the cloud system. On the other hand, with non-uniform traffic pattern, stochastic optimization models are proposed to solve the problem.

In order to validate our placement and scheduling approaches, we provided experimental tests as well as simulation results that have shown the effectiveness of our optimization models in terms of execution time and reducing inter-DCs traffic volume.

7.2 Perspectives

In this thesis, we studied several complex problems related to VM placement within a geo-distributed cloud infrastructure. However, there are some extensions to this work that we aim to address in our future work. In this section, we divide the perspectives into short and long-term perspectives.

7.2.1 Short-term Perspectives

In this work, we studied the VM placement problem mainly form a point of view of the cloud provider, having the objective to minimize data transfer costs over the backbone network. However, we implicitly added constraints that aim at maintaining the QoS and SLAs of the hosted applications. Thus, we also tackled the problem from the consumer perspectives. In particular, we considered location constraints that restrict the placement of VMs in a defined set of DCs. This information is generally provided by consumers. Furthermore, we considered service level constraints as well as scheduling strategies aiming at preventing from performance degradation issues during the migration process.

As an extension of this work, we aim at considering pricing models while making migration and placement decisions. In fact, SLAs contracts have several levels of QoS each one having a distinct price. Therefore, it is interesting to consider this parameter while making placement and migration decisions. Pricing models can vary form cloud provider to another. Even for the same provider, there are several pricing policies according to the VM's configuration or the QoS of the hosted application. These parameters will increase the complexity of the placement problem. Hence, from both consumer and provider perspectives, it is important to find a tradeoff between QoS and costs/prices. We will try to study this particular problem within our future work.

7.2.2 Long-term Perspectives

Internet of Things (IoT) is a new paradigm where many surrounding objects are interconnected in a dynamic network infrastructure and exchanging data between each other in order to offer a given service. These objects can be heterogeneous including personal devices, sensors, cameras, etc [137]. IoT objects are characterized by their limited computing and storage resources which rise many issues regarding the availability, performance and security [138].

To cope with these limitations, cloud computing brings its unlimited resource capabilities and its well-established technologies as a solution. As a matter of fact, cloud and IoT are two complementary paradigms that can be merged together in order to offer better quality and delivery of services [138]. This merged paradigm is called *CloudIoT*. However, although cloud computing can improve for example, IoT communication, there are some limitations that can be arisen when trying to transfer a huge amount of data from the edge of the Internet onto cloud.

CloudIoT networks are considered as the ideal platform for implementing IoT services for a wide variety of smart environment, such as smart grids, smart cities and buildings. In this context, many problems need to be addressed. One of the key issues, is to optimize the placement of edge nodes and finding the best routing

of flows circulating within the network in order to meet QoS requirements and save the overall costs [139].

ClouIoT services involve several heterogeneous network technologies, where many applications require continuous data transmission which will increase significantly the consumption of bandwidth resources. Hence, optimizing bandwidth utilization needs additional effort.

As a long-term perspectives, we aim to study and solve the above mentioned problems with a highly distributed CloudIoT infrastructure where some of the open issues are urgent especially with respect to network communication.

Bibliography

[1] P. M. Mell and T. Grance, "Sp 800-145. the nist definition of cloud computing," Gaithersburg, MD, United States, Tech. Rep., 2011.

[2] R. Boutaba, Q. Zhang, and M. F. Zhani, "Virtual machine migration in cloud computing environments: Benefits, challenges, and approaches," *Communication Infrastructures for Cloud Computing*, pp. 383–408, 2013.

[3] F. Xu, F. Liu, H. Jin, and A. V. Vasilakos, "Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions," *Proceedings of the IEEE*, vol. 102, no. 1, pp. 11–31, 2014.

[4] L. Gu, D. Zeng, S. Guo, and B. Ye, "Joint optimization of vm placement and request distribution for electricity cost cut in geo-distributed data centers," in *Computing, Networking and Communications (ICNC), 2015 International Conference on.* IEEE, 2015, pp. 717–721.

[5] K. Church, A. G. Greenberg, and J. R. Hamilton, "On delivering embarrassingly distributed cloud services," in *7th ACM Workshop on Hot Topics in Networks - HotNets-VII, Calgary, Alberta, Canada, October 6-7, 2008.* ACM SIGCOMM, 2008, pp. 55–60.

[6] M. H. Ferdaus, M. Murshed, R. N. Calheiros, and R. Buyya, "Network-aware virtual machine placement and migration in cloud data centers," *Emerging research in cloud distributed computing systems*, vol. 42, 2015.

[7] T. Wood, K. K. Ramakrishnan, P. J. Shenoy, J. E. van der Merwe, J. Hwang, G. Liu, and L. Chaufournier, "Cloudnet: Dynamic pooling of cloud resources by live WAN migration of virtual machines," *IEEE/ACM Trans. Netw.*, vol. 23, no. 5, pp. 1568–1583, 2015.

[8] Z. Zhang, M. Zhang, A. G. Greenberg, Y. C. Hu, R. Mahajan, and B. Christian, "Optimizing cost and performance in online service provider networks." in *NSDI*, 2010, pp. 33–48.

[9] K.-y. Chen, Y. Xu, K. Xi, and H. J. Chao, "Intelligent virtual machine placement for cost efficiency in geo-distributed cloud systems," in *Communications (ICC), 2013 IEEE International Conference on.* IEEE, 2013, pp. 3498–3503.

[10] L. Gu, D. Zeng, S. Guo, Y. Xiang, and J. Hu, "A general communication cost optimization framework for big data stream processing in geo-distributed data centers," *IEEE Transactions on Computers*, vol. 65, no. 1, pp. 19–29, 2016.

[11] A. G. Greenberg, J. R. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2009.

[12] Y. Chen, S. Jain, V. K. Adhikari, Z.-L. Zhang, and K. Xu, "A first look at inter-data center traffic characteristics via yahoo! datasets," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 1620–1628.

[13] J. Wang, "Survey of state-of-the-art in inter-vm communication mechanisms," *Research Proficiency Report*, 2009.

[14] B. Solomon, D. Ionescu, M. Litoiu, and G. Iszlai, "Designing autonomic management systems for cloud computing," in *ICCC-CONTI, 2010*, 2010.

[15] L. Yu, L. Chen, Z. Cai, H. Shen, Y. Liang, and Y. Pan, "Stochastic load balancing for virtual resource management in datacenters," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[16] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 267–280.

[17] S. Kandula, S. Sengupta, A. G. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements & analysis," in *Proceedings of the 9th ACM SIGCOMM Internet Measurement Conference, IMC 2009, Chicago, Illinois, USA, November 4-6*. ACM, 2009, pp. 202–208.

[18] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1366–1379, 2013.

[19] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE transactions on parallel and distributed systems*, vol. 24, no. 6, pp. 1107–1117, 2013.

[20] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in *Network and Service Management (CNSM), 2010 International Conference on*. Ieee, 2010, pp. 9–16.

[21] N. Serrano, G. Gallardo, and J. Hernantes, "Infrastructure as a service and cloud technologies," *IEEE Software*, vol. 32, no. 2, pp. 30–36, 2015.

[22] N. Manohar, *A Survey of Virtualization Techniques in Cloud Computing*. India: Springer India, 2013, pp. 461–470.

[23] Y. Jadeja and K. Modi, "Cloud computing-concepts, architecture and challenges," in *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*. IEEE, 2012, pp. 877–880.

[24] A. Headquarters, "Cisco data center infrastructure 2.5 design guide," in *Cisco Validated Design I.* Cisco Systems, Inc, 2007.

[25] C. Headquarters, "Data center networking: Enterprise distributed data centers solutions reference nework design," in *Solutions Reference Network Design.* Cisco Systems, Inc, 2003.

[26] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," in *ACM SIGCOMM computer communication review*, vol. 39, no. 4. ACM, 2009, pp. 51–62.

[27] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63–74, 2009.

[28] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," in *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4. ACM, 2009, pp. 39–50.

[29] Amazon inc., Amazon Elastic Compute Cloud Amazon EC2. http://aws.amazon.com/ec2/. 2014.

[30] Azure services platform. http://www.microsoft.com/azure/default.mspx.

[31] J. M. Pujol, V. Erramilli, G. Siganos, X. Yang, N. Laoutaris, P. Chhabra, and P. Rodriguez, "The little engine (s) that could: Scaling online social networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 4, pp. 1162–1175, 2012.

[32] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, and P. Rodriguez, "Greening the internet with nano data centers," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies.* ACM, 2009, pp. 37–48.

[33] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Plachouras, and L. Telloli, "On the feasibility of multi-site web search engines," in *Proceedings of the 18th ACM conference on Information and knowledge management.* ACM, 2009, pp. 425–434.

[34] J. M. Pujol, V. Erramilli, and P. Rodriguez, "Divide and conquer: Partitioning online social networks," *arXiv preprint arXiv:0905.4918*, 2009.

[35] X. Dong, T. El-Gorashi, and J. M. Elmirghani, "Green ip over wdm networks with data centers," *Journal of Lightwave Technology*, vol. 29, no. 12, pp. 1861–1880, 2011.

[36] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman *et al.*, "Reservoir-when one cloud is not enough," *Computer*, vol. 44, no. 3, pp. 44–51, 2011.

[37] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A survey," *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, 2013.

[38] AMAZON. Amazon cloudwatch user guide. [Online]. Available: http://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/acw-ug.pdf

[39] LogicMonitor. Netapp. [Online]. Available: https://www.logicmonitor.com/monitoring/netapp/

[40] NIMSOFT. Getting started guide. [Online]. Available: https://support.nimsoft.com/downloads/server60/NMS_6.00/en_US/NimsoftMonitorGettingStarted20Guide.pdf

[41] S. S. Manvi and G. K. Shyam, "Resource management for infrastructure as a service (iaas) in cloud computing: A survey," *Journal of Network and Computer Applications*, vol. 41, pp. 424–440, 2014.

[42] D. Kliazovich, P. Bouvry, and S. U. Khan, "Dens: Data center energy-efficient network-aware scheduling," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, Dec 2010, pp. 69–75.

[43] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *INFOCOM*. IEEE, 2010, pp. 1154–1162.

[44] D. Ersoz, M. S. Yousif, and C. R. Das, "Characterizing network traffic in a cluster-based, multi-tier data center," in *27th International Conference on Distributed Computing Systems (ICDCS '07)*, June 2007, pp. 59–59.

[45] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo, "Dynamic resource management using virtual machine migrations," *IEEE Communications Magazine*, vol. 50, no. 9, pp. 34–40, 2012.

[46] P. Horn, "Autonomic Computing: IBM's Perspective on the State of Information Technology," Tech. Rep., 2001.

[47] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[48] IBM, "An architectural blueprint for autonomic computing," Tech. Rep., 2005.

[49] S. Bradley, A. Hax, and T. Magnanti, *Applied Mathematical Programming*. Addison-Wesley Publishing Company, 1977. [Online]. Available: https://books.google.tn/books?id=MSWdWv3Gn5cC

[50] S. Rao, *Engineering Optimization: Theory and Practice: Fourth Edition*. John Wiley and Sons, 6 2009.

[51] K. L. Hoffman and T. K. Ralphs, "Integer and combinatorial optimization," in *Encyclopedia of Operations Research and Management Science*. Springer, 2013, pp. 771–783.

[52] M. W. Krentel, "The complexity of optimization problems," in *Proceedings of the eighteenth annual ACM symposium on Theory of computing*. ACM, 1986, pp. 69–76.

[53] G. Cornuéjols, "Valid inequalities for mixed integer linear programs," *Mathematical Programming*, vol. 112, no. 1, pp. 3–44, 2008.

[54] D. F. Rogers, R. D. Plante, R. T. Wong, and J. R. Evans, "Aggregation and disaggregation techniques and methodology in optimization," *Operations Research*, vol. 39, no. 4, pp. 553–582, 1991.

[55] A. I. Ettien, N. Ben Hadj-Alouane, and A. B. Hadj-Alouane, "A scenario approach for a capacity planning problem with stochastic demands," *International Journal of Logistics Systems and Management*, vol. 3, no. 2, pp. 158–173, 2006.

[56] R. Fletcher, *Practical Methods of Optimization*. Wiley, 2013.

[57] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1984.

[58] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, Sep. 2003.

[59] D. P. Bovet and P. Crescenzi, *Introduction to the theory of complexity*, ser. Prentice Hall international series in computer science. Prentice Hall, 1994.

[60] I. Correia, S. Nickel, and F. Saldanha-da Gama, "The capacitated single-allocation hub location problem revisited: A note on a classical formulation," *European Journal of Operational Research*, vol. 207, no. 1, pp. 92–96, 2010.

[61] J. F. Campbell, "Hub location and the p-hub median problem," *Oper. Res.*, vol. 44, no. 6, pp. 923–935, Dec. 1996.

[62] T. Lee, K. Lee, and S. Park, "Optimal routing and wavelength assignment in wdm ring networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 2146–2154, Oct 2000.

[63] M. Tornatore, G. Maier, and A. Pattavina, "Wdm network design by ilp models based on flow aggregation," *IEEE/ACM Transactions on Networking (TON)*, vol. 15, no. 3, pp. 709–720, 2007.

[64] A. Shankar and U. Bellur, "Virtual machine placement in computing clouds," Indian Institute of Technology Bombay, Technical Report, 2010.

[65] H. Teyeb, A. Balma, N. B. Hadj-Alouane, and S. Tata, "Optimal virtual machine placement in a multi-tenant cloud," in *Service-Oriented Computing - ICSOC 2014 Workshops - FOR-MOVES, Paris, France, November 3-6, 2014*, ser. Lecture Notes in Computer Science, vol. 8954. Springer, 2014, pp. 308–319.

[66] A. Amokrane, M. F. Zhani, R. Langar, R. Boutaba, and G. Pujolle, "Greenhead: Virtual data center embedding across distributed infrastructures," *IEEE Trans. Cloud Computing*, vol. 1, no. 1, pp. 36–49, 2013.

[67] B. Kantarci, L. Foschini, A. Corradi, and H. T. Mouftah, "Inter-and-intra data center vm-placement for energy-efficient large-scale cloud systems," in *Workshops Proceedings of the Global Communications Conference, GLOBECOM 2012, 3-7 December 2012, Anaheim, California, USA*. IEEE, 2012, pp. 708–713.

[68] M. R. Korupolu, A. Singh, and B. Bamba, "Coupled placement in modern data centers," in *IPDPS*. IEEE, 2009, pp. 1–12.

[69] R. Cohen, L. Lewin-Eytan, J. Naor, and D. Raz, "Almost optimal virtual machine placement for traffic intense data centers," in *INFOCOM, Turin, Italy*. IEEE, 2013, pp. 355–359.

[70] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, and E. Silvera, "A stable network-aware VM placement for cloud systems," in *CCGRID*. IEEE Computer Society, 2012, pp. 498–506.

[71] T. Yapicioglu and S. Oktug, "A traffic-aware virtual machine placement method for cloud data centers," in *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, ser. UCC '13. Washington, DC, USA: IEEE Computer Society, 2013.

[72] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[73] B. Zhang, Z. Qian, W. Huang, X. Li, and S. Lu, "Minimizing communication traffic in data centers with power-aware VM placement," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, 2012.

[74] D. S. Dias and L. H. M. K. Costa, "Online traffic-aware virtual machine placement in data center networks," in *Global Information Infrastructure and Networking Symposium, GIIS 2012, Choroni, Venezuela, December 17-19, 2012*. IEEE, 2012, pp. 1–8.

[75] J. T. Piao and J. Yan, "A network-aware virtual machine placement and migration approach in cloud computing," in *2010 Ninth International Conference on Grid and Cloud Computing*. IEEE, 2010, pp. 87–92.

[76] V. Mann, A. Gupta, P. Dutta, A. Vishnoi, P. Bhattacharya, R. Poddar, and A. Iyer, "Remedy: Network-aware steady state VM management for data centers," in *Networking (1)*, ser. Lecture Notes in Computer Science, vol. 7289. Springer, 2012.

[77] T. Duong-Ba, T. P. Nguyen, B. Bose, and T. Tran, "Joint virtual machine placement and migration scheme for datacenters," in *GLOBECOM, Austin, TX, USA*. IEEE, 2014.

[78] I. Goiri, K. Le, J. Guitart, J. Torres, and R. Bianchini, "Intelligent placement of datacenters for internet services," in *2011 International Conference on Distributed Computing Systems, ICDCS 2011, Minneapolis, Minnesota, USA, June 20-24, 2011*. IEEE Computer Society, 2011, pp. 131–142.

[79] F. Larumbe and B. Sansò, "Optimal location of data centers and software components in cloud computing network design," in *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012, Ottawa, Canada, May 13-16, 2012*. IEEE Computer Society, 2012, pp. 841–844.

[80] H. Goudarzi and M. Pedram, "Geographical load balancing for online service applications in distributed datacenters," in *IEEE CLOUD, Santa Clara, CA, USA*. IEEE, 2013.

[81] Q. Zhang, Q. Zhu, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "Dynamic service placement in geographically distributed clouds," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12-Supplement, pp. 762–772, 2013.

[82] Z. Usmani and S. Singh, "A survey of virtual machine placement techniques in a cloud data center," *Procedia Computer Science*, vol. 78, pp. 491–498, 2016.

[83] A. Shapiro, D. Dentcheva *et al.*, *Lectures on stochastic programming: modeling and theory*. SIAM, 2014, vol. 16.

[84] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*. IEEE, 2009, pp. 103–110.

[85] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 71–75.

[86] S. T. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 702–710.

[87] H. Jin, D. Pan, J. Xu, and N. Pissinou, "Efficient vm placement with multiple deterministic and stochastic resources in data centers," in *Global Communications Conference (GLOBECOM), 2012 IEEE*, Dec 2012, pp. 2505–2510.

[88] R. Ghosh, F. Longo, R. Xia, V. K. Naik, and K. S. Trivedi, "Stochastic model driven capacity planning for an infrastructure-as-a-service cloud," *IEEE Transactions on Services Computing*, vol. 7, no. 4, pp. 667–680, 2014.

[89] J. Chase and D. Niyato, "Joint optimization of resource provisioning in cloud computing," *IEEE Transactions on Services Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[90] D. Xu, X. Liu, and Z. Niu, "Joint resource provisioning for internet datacenters with diverse and dynamic traffic," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[91] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *Computer Communication Review*, vol. 40, no. 1, pp. 92–99, 2010.

[92] IBM Corporation ILOG CPLEX. http://www.ilog.com/products/cplex/. Visited: 04-02-2013.

[93] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, 2011.

[94] H. Teyeb, A. Balma, N. B. Hadj-Alouane, and S. Tata, "Optimal virtual machine placement in large-scale cloud systems," in *IEEE CLOUD, Anchorage, AK, USA*. IEEE, 2014, pp. 424–431.

[95] I. R. Martín and J. J. S. González, "Solving a capacitated hub location problem," *European Journal of Operational Research*, vol. 184, no. 2, pp. 468–479, 2008.

[96] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[97] V. Mak, "Iterative variable aggregation and disaggregation in ip: An application," *Operations research letters*, vol. 35, no. 1, pp. 36–44, 2007.

[98] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[99] D. Bienstock and O. Günlük, "Computational experience with a difficult mixedinteger multicommodity flow problem," *Mathematical Programming*, vol. 68, no. 1-3, pp. 213–237, 1995.

[100] A. Balma, N. B. Hadj-Alouane, and A. B. Hadj-Alouane, "A near-optimal solution approach for the multi-hop traffic grooming problem," *Journal of Optical Communications and Networking*, vol. 3, no. 11, pp. 891–901, 2011.

[101] L. A. Wolsey, "Strong formulations for mixed integer programs: valid inequalities and extended formulations," *Mathematical programming*, vol. 97, no. 1-2, pp. 423–447, 2003.

[102] A. T. Ernst and M. Krishnamoorthy, "Solution algorithms for the capacitated single allocation hub location problem," *Annals of Operations Research*, vol. 86, pp. 141–159, 1999.

[103] S. Alumur and B. Y. Kara, "Network hub location problems: The state of the art," *European Journal of Operational Research*, vol. 190, no. 1, pp. 1–21, 2008.

[104] E. Feller, "Autonomic and energy-efficient management of large-scale virtualized data centers. (gestion autonome et économique en énergie des grands centres de données virtualisés)," Ph.D. dissertation, University of Rennes 1, France, 2012. [Online]. Available: https://tel.archives-ouvertes.fr/tel-00785090

[105] B. Xia and Z. Tan, "Tighter bounds of the first fit algorithm for the bin-packing problem," *Discrete Applied Mathematics*, vol. 158, no. 15, pp. 1668–1675, 2010.

[106] Y. Li, X. Tang, and W. Cai, "On dynamic bin packing for resource allocation in the cloud," in *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*. ACM, 2014, pp. 2–11.

[107] H. Teyeb, N. B. Hadj-Alouane, S. Tata, and A. Balma, "Optimal dynamic placement of virtual machines in geographically distributed cloud data centers," *International Journal of Cooperative Information Systems*, p. 1750001, 2017.

[108] R. Benali, H. Teyeb, A. Balma, S. Tata, and N. B. Hadj-Alouane, "Evaluation of traffic-aware VM placement policies in distributed cloud using cloudsim," in *25th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2016, Paris, France, June 13-15, 2016*. IEEE Computer Society, 2016, pp. 95–100.

[109] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg, "Live wide-area migration of virtual machines including local persistent state," in *VEE, San Diego, California, USA*. ACM, 2007.

[110] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: A performance evaluation," *CoRR*, vol. abs/1109.4974, 2011.

[111] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," in *HPCS*. IEEE, 2009, pp. 1–11.

[112] R. Ashalatha, J. Agarkhed, and S. Patil, "Analysis of simulation tools in cloud computing," in *Wireless Communications, Signal Processing and Networking (WiSPNET), International Conference on*. IEEE, 2016, pp. 748–751.

[113] Q. Pan, J. Pan, and C. Wang, "Simulation in cloud computing envrionment," in *2013 International Conference on Service Sciences (ICSS)*, April 2013, pp. 107–112.

[114] P. Humane and J. N. Varshapriya, "Simulation of cloud infrastructure using cloudsim simulator: A practical approach for researchers," in *ICSTM'15, Chennai, India*, 2015.

[115] J. T. Piao and J. Yan, "A network-aware virtual machine placement and migration approach in cloud computing," in *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, 2010.

[116] A. Koto, H. Yamada, K. Ohmura, and K. Kono, "Towards unobtrusive VM live migration for cloud computing platforms," in *Asia-Pacific Workshop on Systems, APSys '12, Seoul, Republic of Korea*. ACM, 2012.

[117] S. Lim, J. Huh, Y. Kim, and C. R. Das, "Migration, assignment, and scheduling of jobs in virtualized environment," in *3rd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud'11, Portland, OR, USA*, I. Stoica and J. Wilkes, Eds., 2011.

[118] H. Teyeb, A. Balma, S. Tata, and N. B. Hadj-Alouane, "Traffic-aware virtual machine migration scheduling problem in geographically distributed data centers," in *IEEE CLOUD, San Francisco, USA, 2016*, 2016, accepted.

[119] H. Liu and B. He, "Vmbuddies: Coordinating live migration of multi-tier applications in cloud environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 1192–1205, 2015.

[120] S. Akiyama, T. Hirofuchi, R. Takano, and S. Honiden, "Fast wide area live migration with a low overhead through page cache teleportation," in *CCGRID, Delft, Netherlands*. IEEE Computer Society, 2013.

[121] X. Guan, B. Choi, and S. Song, "Topology and migration-aware energy efficient virtual network embedding for green data centers," in *ICCCN, Shanghai, China*. IEEE, 2014.

[122] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *J. ACM*, vol. 7, no. 4, pp. 326–329, 1960.

[123] M. Desrochers and G. Laporte, "Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints," *Operations Research Letters*, 1991.

[124] C. Rego, D. Gamboa, F. Glover, and C. Osterman, "Traveling salesman problem heuristics: Leading methods, implementations and latest advances," *European Journal of Operational Research*, vol. 211, no. 3, pp. 427–441, 2011.

[125] X. Chen, S. Chen, F. Tseng, L. Chou, and H. Chao, "Minimizing virtual machine migration probability for cloud environments," in *HPCC/EUC*. IEEE, 2013.

[126] A. Strunk, "Costs of virtual machine live migration: A survey," in *SERVICES*. IEEE Computer Society, 2012.

[127] M. F. H. Bhuiyan and C. Wang, "Capability-aware energy-efficient virtual machine scheduling in heterogeneous datacenters," in *IEEE SMC, San Diego, CA, USA*, 2014.

[128] S. Dutta and A. Verma, "Service deactivation aware placement and defragmentation in enterprise clouds," in *CNSM 2011, Paris, France*. IEEE, 2011.

[129] T. Knauth and C. Fetzer, "Spot-on for timed instances: Striking a balance between spot and the instances," in *CGC 2012, Xiangtan, Hunan, China*. IEEE, 2012.

[130] G. Luo, Z. Qian, M. Dong, K. Ota, and S. Lu, "Network-aware re-scheduling: Towards improving network performance of virtual machines in a data center," in *ICA3PP 2014, Dalian, China*. Springer, 2014.

[131] H. Jin, D. Pan, J. Xu, and N. Pissinou, "Efficient vm placement with multiple deterministic and stochastic resources in data centers," in *Global Communications Conference (GLOBECOM), 2012 IEEE*. IEEE, 2012, pp. 2505–2510.

[132] S. Kim, R. Pasupathy, and S. G. Henderson, "A guide to sample average approximation," in *Handbook of simulation optimization*. Springer, 2015, pp. 207–243.

[133] T. Homem-de Mello and G. Bayraksan, "Monte carlo sampling-based methods for stochastic optimization," *Surveys in Operations Research and Management Science*, vol. 19, no. 1, pp. 56–85, 2014.

[134] K. J. Preacher and J. P. Selig, "Advantages of monte carlo confidence intervals for indirect effects," *Communication Methods and Measures*, vol. 6, no. 2, pp. 77–98, 2012.

[135] A. DasGupta, *Normal Approximations and the Central Limit Theorem*. New York, NY: Springer New York, 2010, pp. 213–242.

[136] O. Klopfenstein, "Solving chance-constrained combinatorial problems to optimality," *Comp. Opt. and Appl.*, vol. 45, no. 3, pp. 607–638, 2010.

[137] M. I. Hussain, "Internet of things: challenges and research opportunities," *CSI Transactions on ICT*, vol. 5, no. 1, pp. 87–95, Mar 2017.

[138] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: A survey," *Future Generation Computer Systems*, vol. 56, pp. 684 – 700, 2016.

[139] M. Barcelo, A. Correa, J. Llorca, A. M. Tulino, J. L. Vicario, and A. Morell, "Iot-cloud service optimization in next generation smart environments," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 4077–4090, Dec 2016.