A Resource Management Framework for IaaS in Cloud Computing Environment

By
Khaled Metwally

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In Partial Fulfillment of the Requirements
for a Doctor of Philosophy degree in
Computer and Electrical Engineering

School of Electrical Engineering and Computer Science (EECS)
Faculty of Engineering
University of Ottawa

Abstract

Cloud computing Infrastructure-as-a-Service (IaaS) has gained momentum in the cloud computing research field due to its ability to provide efficient infrastructures. Cloud Service Providers (CSPs) are striving to offer Quality of Service (QoS)-guaranteed IaaS services while also improving their resource utilization and maximizing profit. In addition, CSPs are challenged by the need to manipulate diverse and heterogeneous resources, realizing multiple objectives for both customers and CSPs, and handling scalability issues. These challenges are the motivations behind this work which aims at developing a multi-layered framework for constructing and managing efficient IaaS. The fundamental layer in this framework, the Virtual Infrastructure (VI) composition layer, is dedicated to composing and delivering VIs as an IaaS service. This framework relies on a preparatory step that is defined when all the available resources in the managed space are collected in a large repository, the Virtual Resource Pool (VRP). The VRP creation process unifies the representation of all the diverse and heterogeneous resources available.

Subsequently, the proposed framework performs various resource allocation approaches as working solutions through the VI composition layer. These approaches adopt efficient techniques and methodologies in performing their operations. The working solutions are initiated by designing a composition approach that relies on an ontology-based model representation. The composition approach exploits semantic similarity, closeness centrality, and random walk techniques for efficient resource allocation. As a result, it provides an efficient solution in a reasonable computational time with no guarantee for the optimality of the obtained solutions. To achieve an optimal solution, the composition approach uses a mathematical modeling formulation. In this solution, the concepts of the composition approach have been integrated into a multi-objective Mixed Integer Linear Programming (MILP) model that has been solved optimally. Despite the optimality of the resulting solution, the MILP-based model restricts IaaS resource allocation to a computational running-time challenge, and the issue of limited-size datacenters. To circumvent these issues, a cost-efficient model is proposed. The new model introduces a Column Generation (CG) formulation for the IaaS resource allocation problem in large datacenters acquainted with QoS requirements. Furthermore, this formulation is realistic, adopts large-scale optimization tools that are adequate for large datacenters, and ensures optimal solutions in a reasonable time.

However, growing costs in large datacenters in accordance with the growth of recent large-scale application demands, makes large datacenters economically inefficient. Thus, we advocate a distributed framework for IaaS provisioning that guarantees affordable, scalable, and QoS-assured infrastructure for hosting large-scale applications in geo-distributed datacenters. The framework incorporates two decentralized resource allocation approaches, hierarchical and distributed, that use efficient economic models. These approaches are quite promising solutions for the scalability and computational complexity issues of existing centralized approaches. Finally, the cost-efficient model has been extended to fit the distributed infrastructure by considering additional constraints that impact CSP revenue. Simulation results showcase the effectiveness of the presented work along with the potential benefits of the proposed solutions in terms of satisfying the customers' requirements, while achieving a better resource utilization and CSP payoffs.

Acknowledgements

My journey to completing this dissertation came with many obstacles and challenges. However, there were a great number of people who supported my efforts to help make this thesis possible. I am truly blessed to have had the opportunity to learn from, work with, and be mentored by those who made my graduate experience one that I will always cherish.

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Ahmed Karmouch for his advice, guidance, comments, and the continuous support of my Ph.D study. I have learned a lot from him during this journey and his guidance will be always appreciated throughout my career. Also, I am sincerely grateful to Dr. Abdallah Jarray, for his generous support and constant encouragement throughout my Ph.D. study. He taught me how to identify and attack problems. His broad knowledge and passion for research set up an excellent example for me and will no doubt have a great impact on my future career.

My sincere thanks also goes to my fellow labmates, the members of IMAGINE lab Heli Amarasinghe, Javier Salazar, and Wijaya Ekanayake for providing a cheerful and comfortable working environment. My deepest gratitude to my parents, who have given me so much love and support over the years. I probably owe them much more than I realize at this time. Their continuous support and encouragement provided me with the determination to persevere with my studies. They have always been a source of inspiration throughout my study. My thanks also extend to my wife, *Samah*, for her constant encouragement, patience, support, and love.

To the spirit of my father and my lovely family

Table of Contents

List of Tables

List of Figures

Acronyms

ACO Ant Colony Optimization.

BFD Best Fit Decreasing.

BRW Biased Random Walk.

CCUC Cloud Computing Use Case.

CDN Content Delivery Network.

CG Column Generation.

CSP Cloud Service Provider.

DC datacenter.

EA Evolutionary Algorithm.

FFD First Fit Decreasing.

GA Genetic Algorithm.

IaaS Infrastructure-as-a-Service.

IIAC Independent Infrastructure Allocating Configuration.

ILP Integer Linear Programming.

INDL Infrastructure and Network Description Language.

InP Infrastructure Provider.

IoC Internet of Content.

IoS Internet of Services.

IoT Internet of Things.

ISP Internet Service Provider.

IT Information Technology.

KVM Kernel-based Virtual Machine.

LP Linear Programming.

MILP Mixed Integer Linear Programming.

NDL Network Description Language.

NFS Network File System.

NIST National Institute of Standards and Technology.

NML Network Mark-up Language.

NV Network Virtualization.

NVE Network Virtualization Environment.

OGF Open Grid Forum.

OPEX Operational Cost.

OWL Ontology Web Language.

PaaS Platform as a Service.

PM Physical Machine.

QoE Quality of Experience.

QoS Quality of Service.

RMP Restricted Master Problem.

RW Random Walk.

SaaS Software as a Service.

SAJACC Standards Acceleration to Jumpstart Adoption of Cloud Computing.

SDN Software-Defined-Network.

SLA Service Level Agreement.

SLO Service Level Objective.

SN Substrate Network.

SOA Service Oriented Architecture.

SON Semantic Overlay Network.

SP Service Provider.

VC Virtual Cluster.

VDC Virtual Datacenter.

VI Virtual Infrastructure.

VM Virtual Machine.

VMM Virtual Machine Manager.

VN Virtual Network.

VNE Virtual Network Embedding.

VNO Virtual Network Operator.

VNP Virtual Network Provider.

VoD Video-on-Demand.

VR Virtual Resource.

VRP Virtual Resource Pool.

Chapter 1

Introduction

1.1 Overview

Over the past two decades, the Internet has steadily evolved to fit a new *Future Internet* vision. The *Future Internet* can be defined as "the union and cooperation of the *Internet of Content (IoC)*, *Internet of Services (IoS)*, and *Internet of Things (IoT)*, supported by an expanding network infrastructure foundation" [13], where content, services, and things have become the main orientation of this new vision. The Internet has become the most significant networking infrastructure that realizes this incorporation through enabling the creation, contribution, sharing, and integration of information, knowledge, and objects. The Internet thus far has become the world's largest service infrastructure. Furthermore, *Future Internet's* new vision challenges existing Internet infrastructure and service provisioning paradigms, as well. Various computing paradigms have emerged to contribute to satisfying this new vision, e.g., clustering and grid computing, but many challenges have occurred. In particular, network manipulations on these provisioning paradigms are still at the primitive stage and the current IP-based Internet protocol along with the enormous amount of investment in the infrastructure of the Internet make any disruptive innovation in Internet architecture very difficult. A revolution in networking in these paradigms is required to cope with the rapid advances in recent applications. As a result, the Internet is evolving at a fast pace and is called to emerge as the *Future Internet*. Substantial challenges have arisen throughout this evolution of the Internet [14].

The emergence of the *Future Internet's* new vision depends on a set of properties and requirements to be achieved by any service provider in order to join this new vision

or implement it. In addition, fundamental changes in network architecture and service delivery models are also required by the *Future Internet*. Cloud computing incorporates a set of technologies, e.g., Network Virtualization (NV) and Service Oriented Architecture (SOA), along with other earlier paradigms. These technologies allow cloud computing to become an important trend in the future development of Information Technology (IT) and to strive for IoS realization. Rapid changes that have occurred in recent years have impacted end-user prospects, such that they are no longer satisfied with a traditional interconnected-host view of the Internet. Instead, they are more concerned with an increase in provisioned services, and the emergence of new applications that stimulate and attract users to activities such as on-line gaming and video streaming applications.

Customer Quality of Service (QoS) expectations for provisioned services are increasing and there is a rising expectation of the potential capability of cloud systems as an IT infrastructure to help create new value. Cloud computing is evolving as a significant computing paradigm for sharing resources, including infrastructures, software, applications, and business processes [15]. Such offerings are referred to as Software as a Service (SaaS), Platform as a Service (PaaS), and IaaS [16]. Among them, IaaS is the most basic and supportive service. Infrastructure clouds are the foundation for the future IoS. As such, advances in IaaS cloud computing should address the challenges inherent in meeting the *Future Internet's* requirements by providing new tools and capabilities that let users deploy and manage their applications and development platforms efficiently [17].

In the process, cloud computing has shown the economic benefits of scaling and the efficient use of a generic infrastructure to support a variety of services while realizing the *Future Internet* vision. Many experts believe that cloud computing will change the technological foundation of the Internet, and even reshape the pattern of the entire industry. The potential characteristics of the cloud paradigm enable on-demand provisioning of applications, platforms, and infrastructures that support *Future Internet* requirements. Clients are attracted to the cloud paradigm by its inherent characteristics, the ability

to dynamically scale the available resources and the flexibility of payment options (cost-efficient-economy of scale). However, due to encapsulating many technologies, the cloud community addresses several technology challenges while realizing the *Future Internet* vision. Specific issues that are related to these challenges include: (1) deploying future IaaS clouds, (2) efficiently managing such clouds to deliver scalable and elastic on-demand services, (3) developing cloud aggregation architectures that let cloud providers collaborate and interoperate, and (4) improving the cloud infrastructure's security, reliability, and energy efficiency [17]. Furthermore, competition among different CSPs arises to satisfy their customers' QoS requirements while aiming for high profits.

The widespread development of cloud computing technology has lead to large gatherings of infrastructure resources in clusters (such as servers, network equipment, and storage) named datacenters (DCs). These DCs host diverse and heterogeneous interconnected resources that are organized into racks representing a tree-based model [4]. Service providers rely on these DCs as the main source of resources and services. Frequently, DCs are exposed to fluctuating demand patterns with stringent QoS requirements. In addition, the scale of these DCs is expanding. Indeed, management of a large number of clusters has become very impractical. Efficient virtualization technologies constitute the core of DC implementation and help in efficient resource management. Efficient resource management of DCs maintains the CSPs' goals of achieving customers' QoS expectations while maximizing their revenue. Research in the current literature reveals that the cloud community must tackle challenges to unleash the full potential of IaaS clouds, realize the *Future Internet* vision, and enable future IoS, IoC, and IoT deployment.

In this dissertation, these research and technology challenges are addressed and several barriers to adoption are overcome. This dissertation thus focuses on the system architecture of the cloud IaaS resource management framework, in a bid to improve DC resource utilization and the quality of provisioned services. The desired management framework aims to improve the customers' satisfaction level as well as maximizing CSPs revenue.

Hereafter, the challenges with regards to DC resource management and resource allocation issues will be explored. This chapter briefly discusses the different aspects of DC resource management and resource allocation in the IaaS cloud service model. The motivation for and the objectives of the presented research work are also described. Furthermore, an outline is given of this dissertation's contributions to the current research. Finally, the organization of the remainder of the thesis is presented.

1.2 Resource Management Challenges

Resource management in the cloud computing context can be described as the process of allocating computing, storage, networking and energy resources to a set of applications, in a manner that seeks to satisfy the performance objectives of the applications, the infrastructure providers, and the cloud customers [18]. Resource management is one of the principal problems in providing high-performance service for cloud computing. Recently, CSPs have faced many issues and challenges in managing their DCs efficiently. These challenges are due to the growing scale of modern DCs, the heterogeneity/diversity of available resources and their interdependencies, the variability and unpredictability of the customer's workload, as well as the range of objectives for the different stakeholders in a cloud ecosystem. Consequently, both academia and industry have begun notable research efforts in this area to deal with problems such as how to optimize the allocation and utilization of resources at the datacenter level. Also, attention is directed towards guaranteeing the customers an acceptable level of QoS and methods of cutting the costs of datacenter management [18].

The primary task of a CSP is to satisfy customers' demands for infrastructure resources by achieving customers' Service Level Agreements (SLAs). In addition, CSPs also may pursue multiple objectives at different times, most of these objectives specifically relating to the management of their datacenter infrastructure [18].

Resource management in cloud computing incorporates compute, storage, and net-

work management. Unlike compute and storage management, network management in cloud environments is in its early stages. Managing a network in a cloud includes enabling proper means for Virtual Resource (VR) (e.g. compute and storage) connection and communication. Typically, in terms of Virtual Machines (VMs) with their interfaces, these components can be created, destroyed, or migrated at any time, in a very dynamic way. Because network management is still very basic in cloud environments, most cloud management platforms rely on external management systems for any network-layer configuration (e.g., DHCP servers, manual VLAN establishment, NAT or forwarding rules on IP tables) [19] [20]. Reliance on external management constitutes the first drawback that will be examined in this dissertation. This involves the rather elementary support for network management in cloud management platforms. Researchers have tried to overcome this drawback by allowing multiple controllers in the management systems [21] [22]. A combined management of network and cloud resources has the potential to be a promising solution for this challenge. In addition, exploiting the key performance indicators of the network resources as a new metric in the management process helps in an efficient and agile resource allocation operation [23].

The second challenge is lack of flexibility, in terms of both requirement expressiveness and interactions among management components. For requirement expressiveness, some modern applications, particularly highly distributed and network-intensive ones that could benefit from cloud environments, are inhibited from exploiting existing systems because of inadequate support to specify their strict requirements. Other systems required a customer with a higher level of experience to deal with the system. On the other side, improper management of component interactions impacts the overall performance of the management system when insufficient shared information exists or an unsynchronized event occurs. Proper modeling for the entire domain can be a potential solution for this challenge, where the desired model controls the level of abstraction and the shared information.

The third observed challenge in current management systems is the scalability issue.

Traditional monitoring and management systems are typically centralized. These approaches will not scale well to potentially millions of management objects in cloud systems. In addition, a centralized approach escalates new limitations that result in a single-point-of-failure issue. New approaches are urgently needed that are better distributed and have scalability properties allowing monitoring and management systems to easily scale up or scale down to elastically meet cloud requirements. In another existing challenge, enabling cooperation among multiple service providers, an adaptation layer is urgently needed to manage the compatibility between different CSPs in various administrative domains. This adaptation layer must also control the interoperability between management systems in the case of federated or distributed cloud systems. The following section also focuses on resource allocation as an integrated evolving part of resource management.

1.3 Resource Allocation Challenges

Cloud computing, as an attractive cost-efficient model for individual users and enterprises, faces enormous challenges with regard to allocating resources efficiently in a reasonable time. Resource allocation is an integrated part of many evolving management problems of datacenters [24]. Because of its importance and effectiveness, resource allocation tunes up the efficiency of the management system. Unresolved key issues regarding the process of resource allocation also affect resource management. Through the service provisioning life cycle, a set of challenges arises when developing a resource allocation system.

The first observed challenge is the ***Resource Modeling and Description Challenge***. An important point when allocating resources for serving incoming requests is how the resources are modeled, particularly when the cloud computing environment is comprised of diverse, heterogeneous resources. There are different abstraction levels of service that a cloud can provide for customers, and many parameters that can be optimized during the allocation process. The modeling and description of the resources should consider a

suitable level of abstraction that allows the resource allocation system to work properly. Moreover, the richness of information in describing the resource model impacts the optimization problem, in the sense that more details can give more flexibility and allow for a better usage of resources and vice versa. This concept is called the granularity of the resource's description model. Balance in the model information is required, particularly if the description model is used not only by service providers for advertising their service offers, but also by customers to specify their requirements [25]. The integration of network resources in this description is essential since cloud computing is expected to employ a wide variety of heterogeneous networking systems. Furthermore, describing QoS attributes of network resources and measuring QoS parameters of network services are challenging problems, particularly in dynamic large-scale networks for public cloud service provisioning [25]. Proper cloud resource modeling (resource description) enables efficient management of the cloud infrastructure and solves the flexibility challenge. This modeling is essential to perform all the management operations in the cloud system.

The second observed challenge is ***Resource Discovery***. This process involves the identification and localization of the most appropriate cloud resources from all the available sources, in order to comply with incoming customer requests. Resource discovery and monitoring constitute a core concept through which efficient resource allocation is achieved in a cloud environment. This concept is achieved by using a discovery framework that maintains the customers' demands within specified constraints. Resource monitoring, on the other hand, plays an effective role in ensuring that cloud resources are continuously available to the clients on demand, anytime, and anywhere. In addition, these resources must perform well without any performance degradation in the managed datacenters. This requires resource monitoring to run as a continuous process, and assists in the overall resource optimization process of the cloud environment. In the literature, some research has addressed that challenge by borrowing the discovery and selection mechanisms from preceding paradigms, such as grid computing. Other research has proposed a framework

for a scenario based on a network virtualization environment [26].

The final challenge concerns ***Resource Selection and Optimization***. Once appropriate and cost-effective resources are identified as candidate resources, the ideal resource should be selected. The process of resource selection endeavors to find a resource that completely fulfills all the client requirements in a timely manner while optimizing infrastructure usage. However, the selection of an ideal resource from a list of discovered resources is very complex and is influenced by different aspects of a cloud computing setup. It is clear that selecting solutions from a set of available ones is not a trivial task, due to the dynamicity of the environment and the fluctuation in customers' demands. Service providers are challenged in optimizing resource selections with different customers' requirements, as well as the service providers' objectives. Therefore, service providers should employ various optimization algorithms to select efficient resources that will provide an optimum utilization for the cloud infrastructure resources while maintaining the customers' SLA [25].

All of these challenges affect the performance of the resource allocation process, the resources utilization level, and consequently, the overall management performance. A comprehensive solution for resource allocation challenges is fundamental to any CSP. Such a solution should cover many aspects for the service providers and the customers, with regard to efficient infrastructure provisioning, utilization of the managed resources, and customer's SLA satisfaction. In this dissertation, a comprehensive management system is introduced to assist CSPs and solve the problems of IaaS resource allocation respecting customers' QoS requirements.

1.4 Motivation

In a cloud computing system, computing resources need to be allocated and scheduled in such a way that CSPs achieve high resource utilization, and customers meet their applications' performance requirements with minimum expenditure. This will be called the

"*Cloud Resource Management problem*" in this dissertation. Due to the diversity of cloud resources and ever-changing management needs, resource management has received much attention over the past number of decades. Most of the existing management systems aim at achieving maximum profits for CSPs while maintaining the customer's SLA with low penalties. At the same time, customers struggle to respect their strict SLA and to guarantee performance requirements for their provisioned services at lower costs. The fundamental idea is to support such CSPs and customers' requirements by providing solutions that can adapt to varying management conditions in the working environment, as well as to fluctuations in customers' demands. Thus, an efficient, well-designed resource management system for cloud IaaS providers is urgently needed. Such a system guarantees the achievement of a maximum satisfaction level for both service providers and customers.

Furthermore, existing management systems have certain limitations on network management, such as reliance on external network management (e.g. DHCP servers and NAT) [19] [20]. In addition, existing solutions to this limitation use multiple controllers in the management system. This imposes a common drawback in most of the existing systems by defining a separate controller for each type of services, e.g., Wlres and Nova (OpenStack). Existence of multiple controllers for network and cloud resources ensures differentiation in resource management and incurs more negotiation and coordination between these controllers. This impacts the QoS of the provisioned services as well as the efficiency of the management system that is used.

Moreover, today's existing management systems are further complicated by a lack of flexibility in requirement expressiveness, in the sense that customers need a minimum experience level to deal with these systems. A critical issue that still exists in most existing management systems is the centralized approach in managing their assets. Centralized management is sometimes followed in cases when distributed management is obviously more efficient since the service provider spans a wide coverage area. This approach inhibits the scalability and extensibility of these management systems as the managed datacenters'

scales grow and span the globe.

Despite the efforts of all existing systems towards efficient management, resource allocation as a crucial task in most existing management systems still suffers from computational complexity and solution optimality. A survey of the current literature indicates a focus on efficient resource allocation but many challenges arise. These challenges are due to the diversity and heterogeneity of the available resources. Consequently, additional challenges exist concerning resource discovery and selection, as well as allocation optimization. Also, the existence of multiple controllers adds more coordination overheads to controllers for allocating the resources.

The aforementioned limitations of current resource allocation and management approaches exemplify a set of strong motivations for developing the framework of this thesis. To address these challenges, an efficient management framework following a layered approach was designed. This management framework is characterized by the adoption of conceptual modeling with combined management. We argue that the proposed design addresses the challenges caused by diverse heterogeneous resources, the multi-tenant service model, and fluctuating customer demands. Furthermore, this management framework manages competition in the cloud market among multiple providers.

The proposed management framework of this dissertation has the ability to manage a pool of heterogeneous resources, perform resource allocation, maintain customer's SLA, monitor QoS attributes, and utilize datacenter resources efficiently. These are the minimum requirements for cloud management systems. Further, the proposed framework controls different management objectives as defined by the CSP. Working through the proposed framework, a set of resource allocation solutions can define a suitable mechanism to discover, manage, and allocate the most appropriate resource efficiently. Focus was placed on the design to realize the integrity between management components as well as the flexibility in managing the resources. The aim was to provide a set of solutions that addressed the resource allocation problem in cloud datacenters. These solutions guarantee efficient

resource utilization, optimality of the obtained solutions, CSP profit maximization, and customer's SLA satisfaction. Structuring the management framework in modules and layers ensures scalability and flexibility in the operation of the framework, thus allowing the working solutions to perform properly.

1.5 Dissertation Overview

To build an automatic cloud resource management system, this dissertation has focused on studying the limitations of similar existing systems. This focus was carried out while addressing flexibility, scalability, manageability, interoperability, availability, cost-efficiency and profitability, and datacenter utilization levels. Also, attention is given to the performance of these systems within the proposed framework at different scales, and in different scenarios, including single and multiple datacenters.

Furthermore, resource management and allocation issues are addressed. The resource allocation problem is considered from the providers' perspective and the customer's viewpoint, as well. In this framework, a unified treatment of all the resources and combined management for network and cloud resources is achieved in order to cope with various heterogeneous resources. Furthermore, the unified treatment with the combined management improves the datacenter utilization level, and guarantees higher customer satisfaction levels. The proposed framework takes advantage of the main characteristics of the cloud in addition to other characteristics that include provisioning of a robust and reliable IaaS service, achieving a good reputation for the CSPs, and performing scalability, extensibility, and adaptability of the management system in a timely fashion.

The objectives of the research work presented can be summarized as follows:

- Provide IaaS QoS assured service; maintain customers' SLA, and provide an efficient IaaS service.

- Provide a set of working solutions that address the resource allocation problem respecting customers' QoS requirements.
- Handle diverse resources; conceptual modeling for the cloud resources; achieving a unified treatment of all the available resources.
- Provide flexible management; existence of the generalized virtual resource repository with the combined management of cloud and network resources; eliminate many coordination and communication issues between multiple controllers that incur additional traffic.
- Perform scalability and flexibility; building the management system in a component-based layered architecture which allows the seamless integration of its components and integration with external modules as well. The working models and algorithms cope with this scalability without performance degradation.
- Improve DC utilization; improving the DC utilization level allows the acceptance of more requests, thus preventing the resource fragmentation problem.
- Maximize CSP profits; the main goal of the CSP is to maximize his profit while being constrained by the customers' QoS requirements. Both participants' objectives are considered in the designed framework.

1.6 Summary of Contributions

The goal of this dissertation is to investigate new strategies and techniques for the purpose of designing an efficient cloud resource management framework. An attempt is made to solve existing cloud resource management problems by addressing various aspects from the perspectives of both service providers and customers. The key pillars of this work are the unified VR model, the VRP, and combined management. The proposed framework uses a layered architecture for cloud resource management mainly relying on these supporting

concepts. In addition, a set of resource allocation solutions is proposed while working through this framework towards achieving the desired management objectives. The major contributions of this dissertation can be summarized as follows:

- **Two-Phase Ontology-Based Resource Allocation Approach for IaaS Cloud Service (2P-IaaS) [1]**

  The first contribution of this work is a resource allocation framework that uses composition as a technique. This study illustrates that providers can allocate resources more efficiently by having customers specify a few parameters of their infrastructure requests without exposing greater detail. The proposed composition approach relies on a unified ontology model for a VR; this model is populated to form the VRP as a general repository that hosts diverse and heterogeneous resources. Furthermore, the working composition algorithm adapts fundamental concepts in discovering and composing the required infrastructure; semantic similarity, closeness centrality, and biased random walk techniques. In addition, the algorithm exploits the ontology model associating reasoning capability in reducing the scenario search space and performing resource classifications. Adopting the ontology offerings along with the fundamental concepts not only finds the best solution within the scenario search space, but also proves the concept of applying ontology as an efficient resource allocation approach in cloud computing. In addition, using ontology-based modeling with associated capabilities has confirmed the applicability of the composition technique as a resource allocation solution. The experimental results show that this ontology-based resource allocation scheme achieves a higher utilization for the managed datacenters and also performs efficiently.

- **MILP-Based Approach for Efficient Cloud IaaS Resource Allocation (MILP-2P-IaaS) [2]**

  The second contribution is an optimal solution for the resource allocation problem

based on a mathematical modeling formulation in MILP. An overhauled MILP model is proposed that considers heterogeneity while keeping the simplicity of the current composition methodology. The proposed model formulates the integration of semantic similarity and closeness centrality concepts as a two-phased Mixed Integer Linear Program (MILP-2P-IaaS): (i) mapping of hosting resources, and (ii) connectivity composition. A unified combined manager; the *composer* manipulates this integration for efficient resource allocation. Since the quality of the solution that was obtained using the heuristic algorithm defined in the ontology-based model lacks optimality, the proposed mathematical model guarantees an optimal solution for the resource allocation problem.

- **A Cost-Efficient QoS-Aware Model for Cloud IaaS Resource Allocation in Large Datacenters (RA-IaaS-CG) [3]**

  With the growing scale of datacenters and fluctuation in customers' demand patterns, the ontology-based model suffers from a long-computation-time issue with increasing resources, and an efficiency degradation which requires a solution. Furthermore, the ontology-based solution does not guarantee an optimal solution for the resource allocation problem; it is only adequate for a near-optimal solution in acceptable running time, which may affect the quality of the solution as well. Although the MILP-2P-IaaS solution is optimal, it is based on an MILP formulation. Indeed, solving a MILP with a huge number of resources and requests is known to be NP-Hard, and thus, large-scale instances and models are often computationally intractable. Thus, this research proposes a cost-efficient model acquainted with QoS requirements as a solution to IaaS resource allocation problems in large datacenters (RA-IaaS-CG). This solution makes use of large-scale optimization tools and proposes a CG formulation for the IaaS resource allocation problems. The proposed solution scales well in large datacenters and realizes a cost-efficient model for CSPs, in the sense that it minimizes the cost of VI deployment with efficient datacenter resource usage respecting

the customers' QoS requirements. In addition, this model solves problems of the previous solutions.

- **A Distributed Economics-based Framework for Efficient IaaS provisioning in Geo-Datacenters**

  The evolution of many recent applications that are deployed on large-scale infrastructure, has posed additional challenges for CSPs regarding resource outages and the exponential growth of IaaS demands for hosting emergent applications. However, large datacenters are energy inefficient in hosting the increasing number of managed resources to cope with the fast growth of IaaS requests. Scaling large datacenters incurs additional OPEX on CSPs making them economically inefficient. Furthermore, customers are still unsatisfied with the QoS of the provisioned services and they still experience latency. Distributed datacenters appeared as an economical alternative to the high-energy consumption and costly established datacenters. In addition, deploying datacenters in geographically-dispersed regions favors the new applications to get more benefits regarding their stability and reliability, and to become closer to the end-users. Thus, this dissertation proposes a distributed framework for IaaS resource allocation in Geo-datacenters. The proposed framework design is based on a regional concept, where geographically dispersed datacenters are grouped into regions based on their proximity to each other. A study on Inter/Intra-bandwidth impacts has been conducted to evaluate design effectiveness and regional granularities. Furthermore, two decentralized resource allocation solutions are defined to work through this framework: hierarchical and distributed approaches. The proposed resource allocation solutions address the related challenges of centralized approaches regarding computational complexity and scalability issues. Moreover, we further substantiate the performance and the scalability of the proposed approaches by adapting the recent cost-efficient model (RA-IaaS-CG) to work at the desired regional level. The adapted model entitled (RCG-IaaS) extends (RA-IaaS-CG) with additional con-

straints concerning regional prices and the proximities of the regions. The proposed architecture along with the working solutions confirms the scalability and reduced computational complexity, thereby, improving customers' satisfaction.

1.7 Organization of the Thesis

The remainder of this dissertation is organized into the following chapters:

- Chapter 2: Gives an introduction to the cloud computing paradigm, followed by a definition of cloud computing, a list of cloud computing characteristics, service models, and deployment models. An elaboration on cloud computing enabling technologies is presented, particularly, the virtualization technology. Furthermore, an illustration on datacenters and their different topologies is introduced. There is a brief description of datacenter resource allocation and optimization-related work. Also, a discussion on datacenter energy consumption has been carried out. Finally, a quick survey was done on similar existing commercial and open-source cloud computing systems.

- Chapter 3: Some limitations and challenges of existing management systems are highlighted. Based on these limitations, the layered architecture for the proposed framework is presented. Initially, the proposed system objectives are given. Key enabling technologies that constitute the cloud computing environment are illustrated and the presented architecture is explained layer by layer. Two scenarios are defined that show how a group of users interact with one or more providers when acquiring their resources. Finally, the IaaS provisioning life cycle is presented to illustrate the interaction between different components in the framework and to define the operational steps on the presented architecture.

- Chapter 4: An ontology-based resource allocation approach is presented for cloud IaaS based on a designed composition algorithm in two phases. An elaboration is

given on the conceptual modeling and the proposed VR ontology model. The most useful techniques that work through that approach are explained, including; semantic similarity, closeness centrality, virtual resource pool, and random walk techniques. The execution steps of the designed composition algorithm are briefly explained. In addition, an illustrative example is introduced that explains the approach. Finally, the effectiveness of the proposed approach is evaluated in terms of resource utilization and acceptance ratio against existing benchmarks, although, with some existing limitations.

- Chapter 5: Two mathematical formulations are presented for the resource allocation in IaaS seeking optimality. The devised models ensure the management of cloud resource and network resource convergence, and their unified treatment. In the first model, the problem was first introduced as a multi-objective mathematical modeling in MILP formulation. Then the solution was provided using mathematical programming in IBM CPLEX solver. In the second model, a cost-efficient model considering customer's QoS requirements was devised; this model acts as a large-scale solution for the resource allocation problem in large datacenters that makes use of a Column Generation formulation. This model is presented to cope with the scalability of the provided solution and ensures the optimality of the obtained solution. A performance evaluation has been carried out between the presented models and other popular heuristic solutions. It is observed that the proposed models outperform other solutions.

- Chapter 6: A distributed framework for IaaS provisioning is presented based on regions. The presented framework is well-suited to a set of applications that exploit distributed architectures. In addition, two decentralized resource allocation approaches are presented, hierarchical and distributed approaches. The approaches presented overcome the potential limitations of similar centralized approaches: com-

putational complexity and scalability issues. Moreover, the cost-efficient model has been extended to work on a regional scale considering additional constraints regarding the distributed infrastructure. Finally, a performance evaluation has been carried out to assess the proposed decentralized solutions and to investigate the distributed infrastructure design.

- Chapter 7: A summary is given on the work of this research, including the contributions presented, and there is a discussion of possible directions for future research on this topic.

Chapter 2

Background

With the fast growth in computing and storage technologies and the rapid development of smart devices, people's dependencies on the IT field to accomplish their work has been increased. Furthermore, broad coverage of high speed Internet connectivity technologies has assisted in realizing this dependence. Consequently, the wide spread of intensive applications such as multimedia entertainment applications (Content Delivery Network (CDN) and Video-on-Demand (VoD)) create a challenge for both Internet Service Providers (ISPs) and Service Providers (SPs) to deliver their services at an acceptable level for their customers. Many computing paradigms such as grid computing and distributed processing, utility computing, and pervasive computing exist to deliver customers' intensive applications in a proper way maintaining QoS and Quality of Experience (QoE) satisfactions levels. Competition for low prices with high quality and the increasing challenges in service provisioning have stimulated the emergence of the cloud computing paradigm.

Recently, moving to the cloud computing paradigm became the ultimate solution for the SPs to save their CAPEX instead of upgrading their IT infrastructures. The work presented in this dissertation builds upon and extends such technology to provide an efficient solution that adaptively and dynamically automates cloud service provisioning. This chapter provides the reader with a quick background overview of the emergence of the cloud computing paradigm and its associated key enabling technologies. This chapter also provides readers with an overview of the active research initiatives in the areas of cloud resource allocation and optimization. Finally, the chapter summarizes the similar existing cloud management systems.

2.1 Introduction to Cloud Computing

The evolution of recent concepts like "as-a-service" has formed the future vision of the Internet. With the advent of these recent concepts, cloud computing has gained significant interest in both academia and industry, and allowed users to access applications and Internet services over any devices from any place. The essence of cloud computing appears to be using virtualization techniques to share resources between customers. This evolving paradigm has a significant effect on the existing IT paradigms and the business market from economic perspectives. Cloud computing has promised to deliver its services on a utility basis following the pay-as-you-go pricing model like other utilities such as gas and electricity. This new model explains the significant economic shift in the current IT and service provisioning field. Nowadays, many enterprises are moving their work to the cloud environment to exploit its cost-efficient benefits and compete in the service provisioning market.

Cloud computing can be described as a large-scale and distributed computing paradigm in which a pool of virtualized and managed computing, storage, network resources, platforms, and software services are delivered on demand to the customers through the Internet [30]. Cloud computing in its infancy attempts to overcome all the limitations that exist in the preceding paradigms and it provides potential benefits to service provisioning. These benefits include resource sharing, on-demand service provisioning, and a pay-as-you-go pricing model, all of which characteristics differentiate cloud computing from existing IT paradigms. Cloud computing has appeared as an attractive computing paradigm that benefits both the business owners in reducing their costs, and the customers in removing the burden for planning and provisioning from their concern [31]. A formal definition introduced by the National Institute of Standards and Technology (NIST) [32], states, "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared Pool of configurable computing resources (e.g., networks, servers, storage, applica-

tions, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".

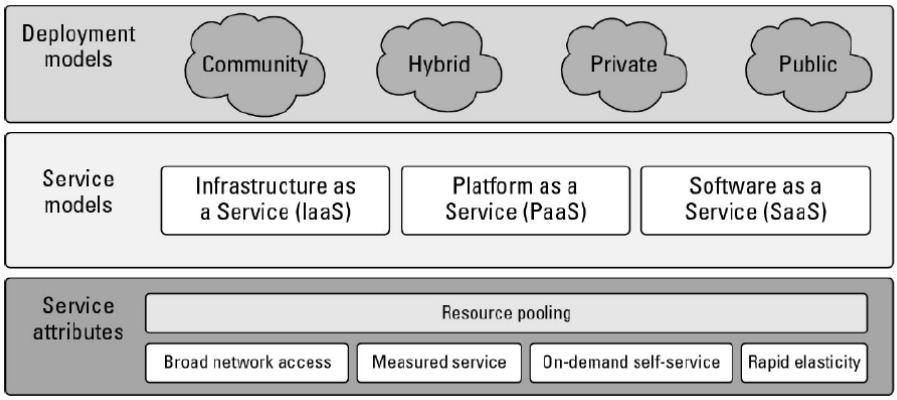


Figure 2.1: Cloud computing essential characteristics, service models, and deployment models [32]

NIST introduced the standard cloud computing model with five essential characteristics, three service models, and four deployment models as shown in Figure 2.1 as follows:

Essential Characteristics:

1. On-demand self-service: the customer can ask for any service, and get this service provisioned on a utility basis, without any human intervention by the CSP.

2. Broad network access: all the cloud services are available over the Internet and accessed through standard methods that can be used on various platforms (e.g., mobile phones, tablets, and laptops).

3. Resource pooling: the CSP provides customers the requested resources through a shared pool supporting the multi-tenant model. This resource pool contains diverse physical and virtual resources that are dynamically allocated and de-allocated according to customer demand.

4. Rapid elasticity: the provided cloud service capabilities can be elastically provisioned and released; they are automatically scaled-in and out without human intervention

and according to the customer requirements.

5. Measured service: CSPs provide metering capabilities that could monitor, control and optimize the service usage, and charge the customer on a utility basis (i.e., a pay-as-you-go pricing approach).

Service Models:

1. SaaS: In this model the customer can use the cloud provider's applications running on a cloud infrastructure. These applications are accessible from various devices. The customer does not have the ability to manage or control or monitor the underlying infrastructure represented by the network, servers, and operating systems.

2. PaaS: The customer adopts the programming languages, libraries, and tools supported and provided by the CSP to deploy his applications onto the cloud infrastructure. The customer does not manage or control the underlying cloud infrastructure including servers, storage, and network; he only has control over the deployed applications.

3. IaaS: The customer is able to use fundamental computing resources including processing, storage, and networks, where the customer can deploy and run arbitrary software including operating systems and applications. The customer is not able to control the cloud infrastructure but has control over the provisioned resources, operating systems, storage, and deployed applications, and limited control over some networking components.

Deployment Models:

1. Private cloud: The cloud infrastructure is owned, managed, and operated by a single organization or a third party for exclusive use; it may be located on or off premises.

2. Public cloud: the cloud infrastructure is provisioned for open use by the general public. This infrastructure may be owned and operated by a business or government organization. It exists on the premises of the CSP.

3. Community cloud: The cloud infrastructure is established for exclusive use by a particular community of consumers from single or multiple organizations that have shared concerns. It may be owned and operated by one or more of the organizations in the community. Also, community cloud infrastructure can be located on or off the premises.

4. Hybrid cloud: This deployment model is a composite of two or more of the previously mentioned models (private, community, or public). These cloud infrastructures are bound together by standardized or proprietary technology that enables data and application portability.

2.2 Cloud Computing Enabling Technologies

2.2.1 Virtualization Technology

A key enabling technology in cloud computing is virtualization technology. Virtualization is the process of emulating the hardware or software environment that appears to a user. The emulated environment is exposed to the user as a complete instance of computing resources, or as a logical association of these resources. This environment gives an advantage over the original configuration [18]. The generated emulated environment is a new virtual view of the resources, denoted by a VM.

Virtualization helps in hardware consolidation when several servers are underutilized, and VMs can be provisioned as needed, endowing an organization with reductions in hardware cost. Additionally, VMs can be migrated from one physical location to another in

a seamless way, unlike traditional computing hardware that is difficult to move once deployed. The main objectives of virtualization are: reducing administration efforts, increasing hardware utilization, and creating flexible and adaptive infrastructure. Furthermore, the virtualized infrastructure can be easily maintained and scaled accordingly to resource demand. Several types of virtualization exist including hardware, software, desktop, memory, storage, and network. Hardware virtualization is of interest for the current study [1].

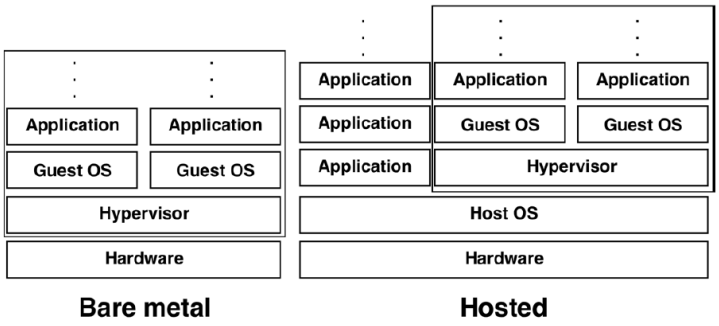


Figure 2.2: Full virtualization architectures [1]

Hypervisors

A basis for system virtualization is the Virtual Machine Manager (VMM), or simply, the hypervisor. VMM handles the creation and the execution of multiple VMs on a single common physical machine. Thus, VMM multiplexes a single physical resource onto multiple virtual resources [33]. Each VM has its operating system, guest OS, and applications that run inside such a VM and it uses its virtual resources such as a virtual CPU, virtual memory, virtual disks and a virtual network card. The hypervisor also provides isolation of guests from each other, protection and security, and the distribution/ management of resources among different running OSs. In addition, it allows independent start-up, reboot and shutdown for any of the VMs.

A well-known hypervisor, Kernel-based Virtual Machine (KVM) is a full virtualization

solution for Linux on x86 hardware containing virtualization extensions. KVM consists of a kernel module, *kvm.ko*, which provides the core virtualization infrastructure. KVM is a lightweight VMM that runs as a kernel module inside the Linux kernel. KVM consumes few resources, decreases the services for interrupts, and eliminates context switching between the driver domain and the hypervisor. Hence, KVM is a good choice for constructing architectures for cloud computing environments [33]. Xen is another popular open-source VMM on a single x86-based physical platform hypervisor. Xen offers a powerful, efficient, and secure feature set for virtualization of x86, x86_64, IA64, ARM, and other CPU architectures [34]. Xen supports paravirtualization that requires changes in the OS to interact with the VMM.

The broad definition of Virtualization refers to the creation of a virtual, as opposed to tangible, version of an object. This definition can be applied to many contexts. Previously, virtualization technology has been elaborated on from the perspective of the physical machine to extend the applicable scope of virtualization to distributed systems and cloud computing that contains multiple resources. Resource virtualization refers to the isolation or the combination of part or all of a computing device's hardware resources for different or shared purposes respectively. The concept of resource virtualization is presented as a key enabler part of virtual resource management in IaaS providers. Resource virtualization in cloud computing includes the following types:

- Server Virtualization: Provides isolated access to server resources while hiding the underlying implementation details of the hardware from the end-user to increase the utilization and improve the management.

- Storage Virtualization: Involves pooling of physical storage devices from multiple networked devices, presenting the appearance of a unified storage device that is managed from a central location.

- Network Virtualization (NV): Refers to a method of multiplexing the use of network

devices while isolating traffic and disguising the true complexity of the underlying network topology.

The main enabling component of Server Virtualization in cloud computing is the hypervisor. Storage Virtualization is realized through the virtual disk while NV has recently been achieved through Software-Defined-Network (SDN) technology [35]. Hereafter, an elaboration on NV is given, the fundamental pillar in the cloud computing paradigm.

2.2.2 Network Virtualization

The Internet has been stunningly successful over the past three decades in supporting plenty of distributed applications and a wide variety of network technologies. However, Internet popularity has become the biggest barrier to further growth. Due to the multi-provider nature of the Internet, adopting a new architecture or modification of the existing one requires consensus among competing stakeholders. As a result, alterations in the Internet architecture have become restricted to simple incremental updates. Thus, the deployment of new network technologies has become increasingly difficult [36] [37]. In that context, the existence of virtualization technology offers a promising solution for developing new network technologies.

Recently, NV appeared as a long-term solution to the existing Internet ossification problem and has become an integral part of the next-generation networking paradigm [38] [39]. In NV, multiple Virtual Networks (VNs) are allowed to coexist over the same physical infrastructure in a seamless manner. NV provides flexibility, promotes diversity, and promises security and easy manageability. The popular NV architecture was introduced by 4WARD in [40] [41]. In a Network Virtualization Environment (NVE), each service provider is able to use an arbitrary network topology, routing or forwarding functions, as well as customized control protocols that are independent of the underlying physical network and other coexisting virtual networks [38]. An individual VN consists of a subset of

the substrate network resources including virtual nodes and virtual links.

NV is defined by decoupling the roles of traditional ISPs into two independent entities [37] [42]: an Infrastructure Provider (InP), who manages the physical infrastructure, and an SP, who creates VNs by collecting resources from multiple InPs and offers end-to-end services. Such an environment will proliferate the deployment of coexisting heterogeneous network architectures free from the existing Internet's inherent limitations. Figure 2.3 presents the NV main business entities. InP can be decoupled into Virtual Network Operator (VNO), Virtual Network Provider (VNP), and InP [2].

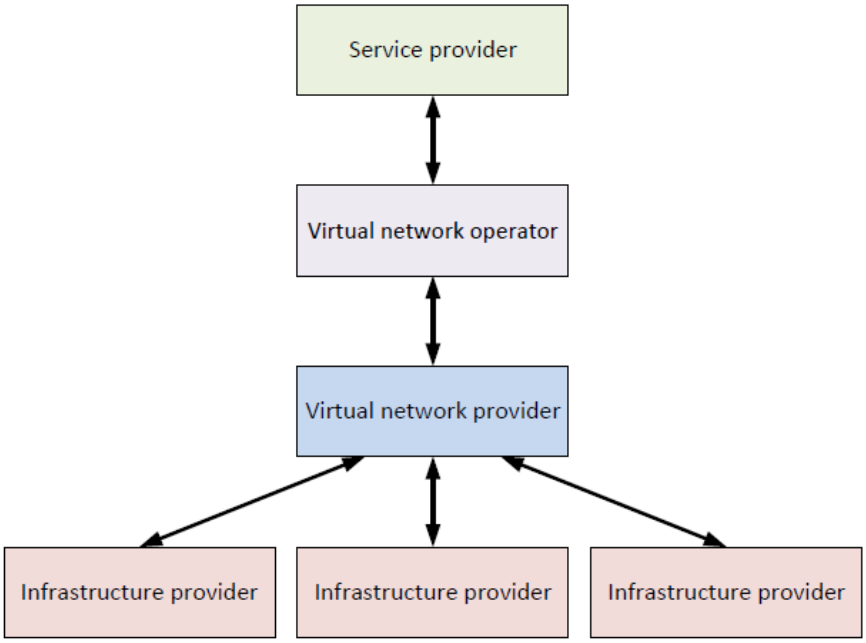


Figure 2.3: Main business roles in Network Virtualization [2]

InPs try to allocate resources to each VN request efficiently in order to ensure a high utilization of their Substrate Network (SN) resources. The VN embedding process involves mapping of virtual nodes/links to substrate nodes/paths in an efficient manner that ensures a high and balanced utilization of SN [43] [44]. A significant challenge for VN embedding approaches is to provision the network resources among hosted VNs dynamically in order to adapt the dynamic variations in the hosted applications (e.g., [45] [46]). Moreover, with

today's advances in cloud applications, NV should be able to run multiple homogeneous customized VNs considering different performance objectives.

IT organizations have realized the advantages, the significance, and the efficiency of VNs compared to the traditional setup. Such organizations are contemplating measures to migrate to this new paradigm [47]. Although researchers argue about the significance of VNs, they agree on the advantages of VNs over the rigidity of traditional networks [48]. Different studies have been carried out to define the promising characteristics of an ideal virtual network, resulting in the following list of traits which represent the promising characteristics that should exist in an ideal VN [48]:

1. Flexibility: The ability of SPs to implement diverse network topologies that satisfy their business needs is a primary requirement for any virtual network framework. SPs must have the capability to define their network characteristics without facing any hardware constraints.

2. Manageability: VNs adopt a segmented architecture where the physical layer is separated from the network layer. This working environment is much easier to maintain. In an ideal VN, SPs should have complete control to manage their VNs.

3. Scalability: A Virtual Network Embedding (VNE) must be scalable. InPs should provide robust hardware infrastructure to accommodate multiple virtual networks without any trade-offs in performance and QoS [49].

4. Isolation and Stability: The multiple VNs' operations in hardware layers should be operated independently. A standalone VN should be able to avoid interference from or dependence on the neighboring networks, thus, ensuring clients' privacy. For stability, an ideal virtual environment should be resilient such that in a network breakdown scenario, other VNs should be able to recover independently and get back to the last stable state.

5. Support for legacy network setup: While implementing a VN, it is necessary to consider the integration of the established traditional setup and the new virtual architecture. Supporting a legacy network can be critical from different perspectives such as stability and disruption prevention.

2.3 Cloud Datacenters

Previously, in traditional IT paradigms, SPs have used dedicated clusters of commodity computers and storage to run their applications and store their data. Recently, with the growing demands for high computational power and the development of new intensive applications, a bottleneck problem occurs. This problem lays in the inter-node communication bandwidth between cluster nodes. This emerging problem is ill-suited for cloud computing applications; it forces ISPs to look for a solution. The cost-efficient choice to build a communication paradigm for large-scale clusters is to use commodity switches and routers to interconnect cluster machines, following a well-known architecture form what is called a Datacenter (DC). We can describe the datacenter as a place where a large number of servers are installed along with associated networking equipment to provide computing services to a large number of clients [4].

CSPs are employing datacenters as the main way of hosting their resources and services. Popular service providers, e.g. Google, Facebook, Yahoo, and Amazon, store a large amount of data and provide services by adopting datacenters. The infrastructure facilities required for operating large-scale datacenters, e.g., cooling systems and power systems, are quite large and require a substantial investment as well as operating expenditures. The communication infrastructure that is used in a datacenter constitutes the datacenter network. This communication infrastructure is described by the network topology, the routing/switching equipment, and the implemented protocols (e.g., Ethernet and IP). Figure 2.4 presents the conventional datacenter network topology. This topology is organized

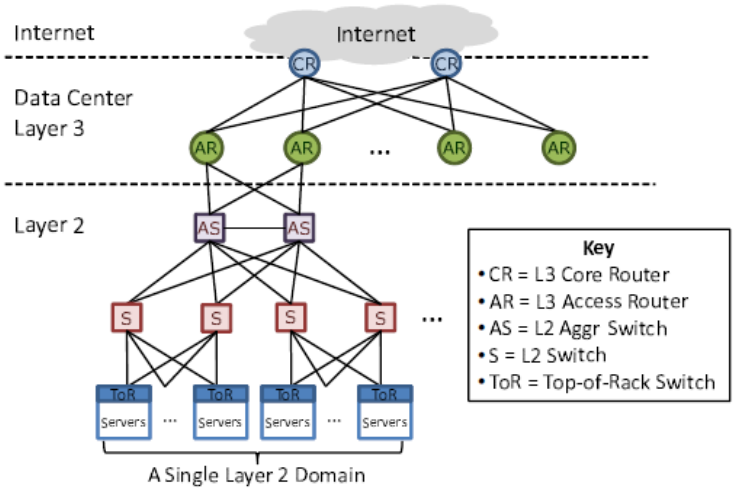


Figure 2.4: Cloud datacenter architecture [3]

into three layers: the access layer, the aggregation layer, and the core layer. The Top-of-Rack (ToR) switches in the access layer provide connectivity to the servers positioned on every rack. Each Aggregation switch (Agg) in the aggregation layer forwards traffic from ToR switches to the core layer. Every ToR switch is connected to multiple Agg switches for redundancy. The core layer provides the connectivity between aggregation switches and core routers (Cor) connected to the Internet.

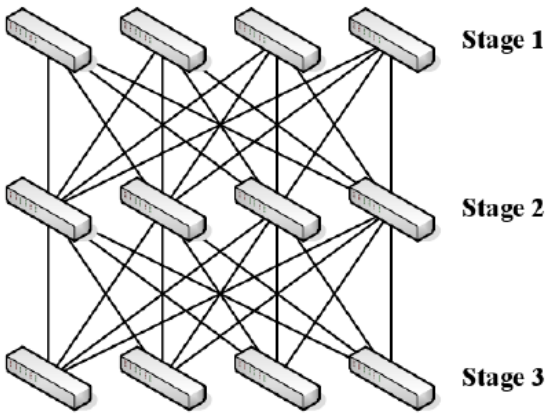


Figure 2.5: Clos topology [4]

Different topologies have been defined to describe the datacenter communication infrastructure. *CLOS* topology is one of these topologies, where the topology architecture

is built up from multiple levels of switches [50]. Each switch in a level is connected to all switches on the next level; these connections provide extensive path diversity. Figure 2.5 shows an example of a three-stage *Clos* topology. Furthermore, *Fat-tree* topology [51] is a particular type of *Clos* topology; it is organized into a tree-like structure, as shown in Figure 2.6. *Fat-tree* topology is comprised of k-port switches containing k pods; each pod has two layers, aggregation and edge. Each layer hosts $k/2$ switches. Each layer has $(k/2)^2$ core switches with one port connected to each pod. The ith port of any core switch is connected to pod i so that the following ports in the aggregation layer of each pod are connected to core switches on $k/2$ steps. Each edge switch is directly connected to $k/2$ end hosts; each of the remaining $k/2$ ports of an edge switch is connected to $k/2$ ports of an aggregation switch [52].

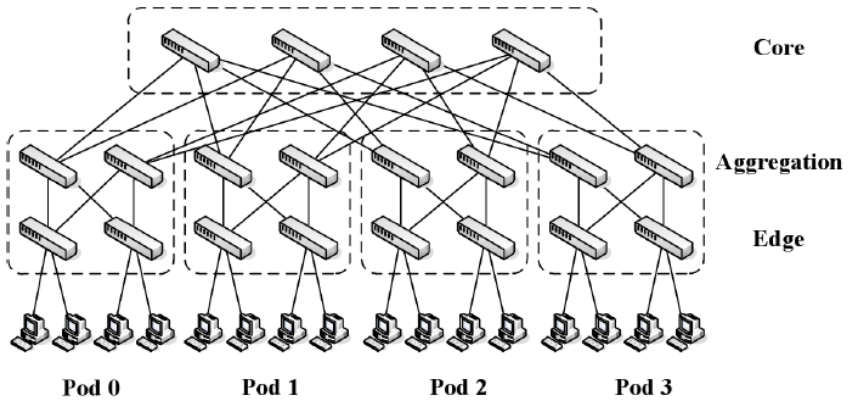


Figure 2.6: Fat-tree topology (k = 4) [4]

Datacenter topologies are not limited to well-known topologies; a number of alternative topologies have emerged such as BCube [53], which is a datacenter network architecture based on hyper-cube topology and randomized small-world topologies [54]. An interesting comparison of recent datacenter network topologies can be found in [55]. In all cases, the main goal is to design a scalable topology without performance degradation; the designed topology should significantly impact the network performance and fault tolerance [56].

Datacenter's traffic model can be divided into two broad categories based on the type

of the hosted applications: north/south and east/west communication. North/south traffic corresponds to the communication between a source and a destination host where one of them is located outside the cloud platform such as user-facing applications (e.g., web services) typically exchanges data with users. In contrast, east/west traffic is the communication in which both ends are located inside the cloud such as MapReduce that requires coordination among its VMs, generating east/west communication. Moreover, applications such as searching and parallel computing require multiple servers to collaborate, and this increases traffic between servers. Traditional datacenters mainly provide access for external users, so most traffic moves in a north-south direction. The fast-growing demands for cloud computing services have a great impact on the datacenter traffic model. In the sense that, traditional networking models have bottlenecks and impose limitations on the datacenter growing. Thus, datacenter networks must change to meet the needs of cloud computing datacenters and cope with the fast growing cloud services. Furthermore, cloud datacenters perform dynamic VM migration that creates complexity and higher east-west traffic volume. This makes traditional datacenter networks are unable to meet the demands of this traffic model. Therefore, traditional datacenter network models, e.g., *CLOS* and *Tree-like* models should be explored, particularly, if studies indicate that the east-west traffic represents 75% of the current traffic volume and is growing on a large scale [57].

2.3.1 Datacenter virtualization

Virtualization promises to change the way that datacenters operate, by breaking the bond between the physical servers and the shared resources granted to the customers. Virtualization can be used to slice a single physical host into one or more VMs that allow resource sharing. This availability can be useful in a hosting environment where the requested applications do not need the full power of a single server. In such a case, virtualization provides an easy way to isolate and partition server resources. The adoption of Server Virtualization, Storage Virtualization, and Network Virtualization in datacenters utilizes

the datacenters' resources efficiently and overcomes many issues in traditional datacenters, e.g., resource fragmentation.

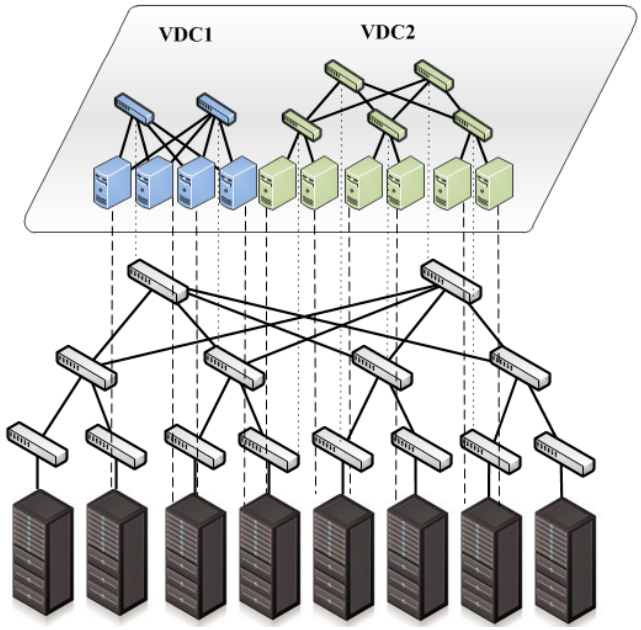


Figure 2.7: Virtualization of a datacenter [4]

Currently, attention is directed towards enabling the virtualization technology in a datacenter's network [4]. Network virtualization is a natural extension that implements service differentiation via QoS policies and segregates traffic for performance isolation, thus permitting high-level traffic engineering and deployment of custom network addressing schemes as well as networking protocols[18]. In the literature, sufficient attention has been paid to datacenter network virtualization; for example, SecondNet [5] mainly focused on providing bandwidth guarantees among multiple VMs in a multi-tenant Virtual Datacenter (VDC). As an example of VDC, Figure 2.7 represents VDC1 and VDC2 as instances of a virtualized datacenter after applying virtualization techniques on a traditional datacenter.

As shown in figure 2.8, SecondNet defined a single entity that is responsible for deploying VDC based on a requirement matrix, this matrix represents the requested bandwidth between each VM pairs. The SecondNet bandwidth guarantee is achieved through provisioning of three types of basic services: (1) type 0 represents a high priority end-to-end

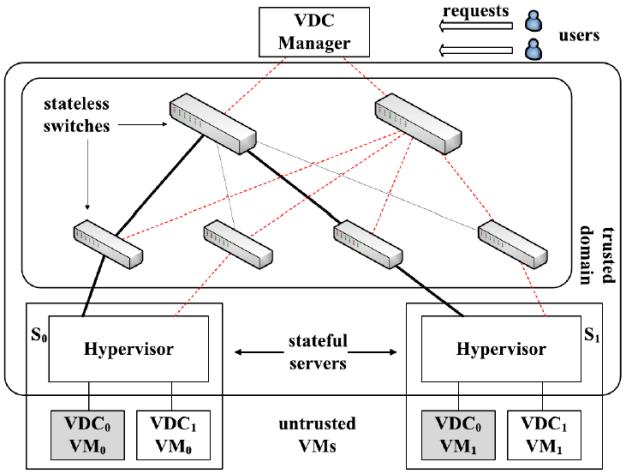


Figure 2.8: SecondNet Architecture [5]

guaranteed service, (2) type 1 defines a better service than best-effort service where bandwidth is guaranteed, and (3) type 2 that represents a best-effort service. Furthermore, SecondNet used a forwarding scheme called port-switching source routing (PSSR), this modified scheme forwards packets using port numbers instead of MAC addresses. However, the physical topology of a network in SecondNet constitutes the main limitation that impacts its performance, i.e., BCube network achieves high network utilization while VL2 and fat-tree networks cannot achieve it. Furthermore, SecondNet does not consider other performance characteristics such as latency that are important to customers.

Rodrigues *et al.* [58] addressed the problem of network performance isolation and proposed "Gatekeeper". Gatekeeper focuses on achieving a high-bandwidth utilization by providing guaranteed bandwidth among VMs in a multiple-tenant datacenter. Furthermore, the authors argue that a proper solution for network performance isolation should possess scalability in terms of the number of VMs and predictability in terms of the network performance. Furthermore, authors are seeking a robust solution against malicious behavior of customers and be flexible concerning the performance guarantees at different levels. Gatekeeper tackled the strict bandwidth guarantee by defining rate limiters for each VM pair. Like many other existing schemes, Gatekeeper does not consider other performance

metrics such as latency. Furthermore, Gatekeeper is still under development, it needs complete implementation and more realistic experiments for comprehensive evaluation to truly assess the effectiveness of Gatekeeper in real cloud environments.

CloudNaaS [21] is a virtual network architecture that offers efficient infrastructure for deploying and managing enterprise applications in clouds. In particular, the supported infrastructure constitutes a set of primitives that achieve the enterprise applications requirements including application-specific address spaces, middlebox traversal, network broadcasting, VM grouping, and bandwidth reservation. CloudNaaS adopts OpenFlow forwarding scheme in achieving its requirements (e.g., middlebox traversal). To deploy an application in CloudNaaS, it goes through several steps. First, the end-user specifies the desired application network requirements to the cloud controller using a predefined network policy language primitives. The network controller then translates the network requirements into a communication matrix. Using the derived matrix, the cloud controller determines the placement of VMs and generates the required network-level rules that will be installed on switches. Currently, CloudNaaS places VMs by considering the communication locality in defining a modified greedy bin-packing heuristic. Moreover, CloudNaaS supports handling online failures and changes in the network policy specification, this online mechanisms can be achieved by re-provisioning the VDCs. One limitation of CloudNaaS is, the occurrence of traffic congestion due to limiting the traffic to a few paths, this limiting traffic may leads to poor network utilization. In addition, the trade-off between scalability and network utilization is still a challenging problem for CloudNaaS.

Although CSPs offer on-demand computing resources to customers through allocating VMs in datacenters, they do not guarantee the performance of the provisioned network resources. The mismatch between the customers' desired performance and the actually achieved performance impacts application performance hosted in datacenters that leads to a management challenge. Furthermore, unpredictable network performance impacts negatively the application productivity and so customer satisfaction is affected. These

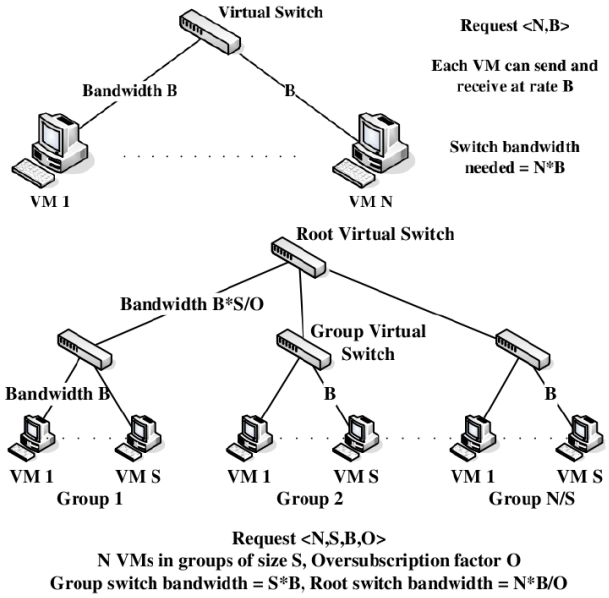


Figure 2.9: Oktopus proposed Abstractions [6]

factors lead to revenue losses. Oktopus [6] presents two virtual network abstractions (a virtual cluster and a virtual oversubscribed cluster) as shown in Figure 2.9. This implementation aimed to control the balance between the providers' revenue and the cost of the performance guarantees offered to customers. Oktopus not only increases application performance, but also offers InPs a better flexibility. Furthermore, Oktopus allows customers to find a balance between higher performance and lower cost. These abstractions suit a wide range of applications. A customer can choose the appropriate abstraction for his desired application, also he can choose the degree of the oversubscription of the VN based on the application's communication pattern (e.g., user-facing web-applications, data intensive applications). Oktopus allocates the resources to the VDC using a greedy algorithm that works for tree-like physical network topologies. Working in tree-like network topologies constitutes the main limitation of Oktopus. Thus, an open challenge still exists regards the implementation of Oktopus abstractions for other network topologies.

Recently, the emerged software-defined concept addresses control and management challenges known in traditional platforms by hiding their complexities from the end users. This is achieved by isolating the data plane from the control plane, i.e., control of the datacenter

is fully automated by software. This concept has been applied to datacenter infrastructure (storage, security, compute and network) that defines all the needs of the datacenters by adopting virtualization techniques. Software-defined datacenters are considered as the evolutionary result of virtualization and cloud computing technologies. This is in contrast to traditional datacenters where the infrastructure is typically defined by hardware and devices.Thus, all the datacenter elements are virtualized and delivered as a service including networking, storage, and CPU. In that case, there are three core components of the software-defined datacenter: network virtualization, server virtualization, and storage virtualization. Also, it may be called May also be called software-defined datacenter (SDD) or virtual datacenter [59].

2.3.2 Resource Allocation in Cloud Datacenters

Resource allocation, as a crucial evolving part in many datacenter management systems, has been studied in the literature with sufficient proposals. Most of these proposals have employed different techniques in performing various approaches in resource allocation including initial placement for VMs and static and dynamic resource allocation. Initial placement of VMs on Physical Machines (PMs) has had plentiful attention in research. The placement of VMs onto PMs is similar to the vector bin-packing problem, which can be used to model static resource allocation problems. The vector bin-packing problem and its variants are NP-hard problems [60]. Thus, heuristic algorithms are frequently proposed and evaluated as promising solutions for this type of problems, e.g., Jung *et al.* [61], Gupta *et al.* [62], and Li *et al.* [63]. Most proposed heuristics are based on greedy algorithms that use simple rules, such as First Fit Decreasing (FFD) and Best Fit Decreasing (BFD).

Meng *et al.* [64] proposed a VM placement solution in which multiple VMs are consolidated and provisioned together, to exploit the statistical multiplexing among individual VMs' workload patterns. Addressing the placement of a set of VMs involves additional

constraints. Shi *et al.* [65] considered a range of constraints that are related to full deployment, anti-colocation and security. Jayasinghe *et al.* [66] considered three types of constraints: demand constraints, placement constraints, and communication constraints, and used approximation algorithms to find a solution.

Following the initial placement, VMs can be rescaled and relocated as the customer's requirements change and also as management policies change on the CSP side as well. VM migration is one of the powerful tools that achieves the reallocation of VMs [67]; the running VMs are paused, serialized and moved to a different PM; then they are once again scheduled for execution. Live migration offers many benefits, including improving fault management and simplifying system maintenance. From the resource management perspective, live migration enables global scheduling objectives such as balancing the load across PMs and consolidating VMs in a minimal number of PMs. Many studies have assessed the overhead of VM live migration, e.g., [68, 67, 69, 70], and many proposals have been introduced to optimize the migration process, e.g., [71, 72, 73, 74, 75].

Sharma *et al.* [76] proposed a set of techniques for VM rescaling, replication, and live migration. The problems are formulated as Integer Linear Programming problems and greedy heuristic solutions are proposed. Zhang *et al.* [77] addressed the control of VM migrations in scenarios where the CSP intentionally over-commits their PM resources to VMs. The proposed technique minimizes the number of VM migrations while migrating VMs with strong utilization correlation to different PMs to minimize the risk of future PM overload. Xu and Li [78] proposed a novel formulation of the VM live migration problem where the VM live migration problem was represented in terms of the Stable Marriage problem [79]. The authors argued that Stable Marriage is more appropriate than optimization-based formulations. They proposed a polynomial time algorithm that generates a stable egalitarian match between VMs and PMs and balances the VM performance with migration overhead.

Extensive use of live VM migration and consolidation significantly contributes in per-

forming dynamic resource allocation. In [80], Bobroff. *et al.* introduced a management algorithm that performs dynamic resource allocation in virtualized environments. The algorithm proactively adapted to changes in demand and migrated VMs between physical hosts thus providing SLA guarantees. Furthermore, to minimize the number of PMs required to support a workload, the adopted algorithm combined forecasting techniques and a bin-packing heuristic. This algorithm is based on Measure-Forecast-Remap (MFR), which starts by collecting historical data, forecasting the future demand, and then remapping VMs to PMs. Moreover, a Minimum Cost Maximum Flow (MCMF) algorithm for VM placement in clouds has been proposed in [81]. The proposed approach employed a time-series model to forecast future requests, thus exhibiting excellent performance and scalability, and solving the challenges which occurred due to dynamic workloads and flow variations. Furthermore, the authors proposed an exact modified Bin-Packing formulation as a benchmark for the MCMF algorithm. The latter formulation was combined with a prediction mechanism to handle the dynamic variations.

Intuitively, as the network significantly impacts the resource allocation performance, it is beneficial to place VMs that interact with each other nearby in the datacenter network topology. Meng *et al.* [82] addressed the traffic rate aggregations at switches in the datacenter, while Hu *et al.* [83] discussed the network-aware approach from the perspective of a customer who has leased a group of VMs constrained by inter-VM bandwidth limits. Similar approaches to minimizing datacenter traffic have been proposed by Jayasinghe *et al.* [66], Shrivastra *et al.* [84], and Wen *et al.* [85].

Moving to large-scale and distributed datacenters, Zhang *et al.* [86] investigated dynamic VM placement in Geo-distributed clouds. The authors' main goal was to minimize the hosting cost while achieving the Service Level Objectives (SLOs) under various demand patterns and resource costs. Their proposed solution was based on control and game-theoretic models. Alicherry *et al.* [87] extended the network-aware VM placement problem to Geo-distributed clouds. They proposed algorithms that select on which data-

center to place a VM, and within this datacenter, which PM to host this VM.

In general, the resource allocation literature possesses much research work that has been proposed to achieve efficient management with different objectives, e.g., network-aware [88], context-aware, and SLA-aware. The main objectives of datacenter owners are the efficient utilization of the managed resources, high availability of resources with operations running on a 24/7 basis, fault tolerance, and security maintenance.

2.3.3 Resource Allocation Optimization-related Work

Obviously, recent research proposals have employed different techniques for resource allocation optimization motivated by a set of aims and objectives, e.g., SLA-aware [89], Context-aware [90], Network-aware [87], Cost-aware [76], and most of the proposals have targeted the energy-aware [91] objective. Many proposals have aimed to optimize the resource allocation process using different tools including linear programming, heuristics, meta-heuristics, evolutionary algorithms, queuing models, and statistical methods. Therefore, resource allocation can be an efficient technique for further management objectives. In [92], resource allocation algorithms have been presented with the objective of minimizing the infrastructure cost and SLA violations. Further, the algorithms have considered both the customers' QoS parameters in terms of response time and infrastructure parameters in terms of service initiation time. In [93], the authors proposed an Integer Linear Programming (ILP) formulation for the problem of assigning SaaS customer workflows components to multiple IaaS providers, their proposal considering the SLA at two levels: the first level with its customers, and the second level with its IaaS providers with the objective of minimizing SLA violations. Thus, two heuristics algorithms have been proposed to solve the relaxed version of the presented ILP and obtain feasible integer solutions.

Furthermore, in [94] Yusoh *et al.* presented evolutionary algorithms to perform an initial placement and to solve resource optimization problems for composite SaaS. Both

problems are formulated as combinatorial optimization problems with the objective of improving the execution time of SaaS services, and utilizing the resource usage as well. In [95] the same authors proposed a penalty-based grouping genetic algorithm for composing multiple SaaS components. The authors aimed to minimize the used resources by adopting clustering techniques considering specific constraints.

Bodík *et al.* in [96] presented a comprehensive analysis of a large-scale web application including its communication patterns. Based on the conducted analysis, the authors proposed and evaluated a novel optimization framework. The proposed framework exploits the skewness observed in the communication patterns in achieving high fault tolerance and reducing bandwidth usage in the network core. Thus, the proposed framework captured the trade-off between reducing the used bandwidth and improving the application's fault tolerance.

Regarding the profit maximization objective, Feng *et al.* in [97] focused on how a CSP can maximize revenues through SLA-based resource allocation. Thus, the authors proposed a queuing-theory-based mathematical formulation for the resource allocation problem that considered various QoS parameters including the available resources, the requests arrival rate, the service time, and the adopted pricing mechanisms. They aimed to maximize the revenue by serving each customer as an M/M/1 model and finding how many servers should be assigned for each service instance. Furthermore, Hong Xu and Baochun Li in [98] addressed the dynamic pricing problem in an IaaS cloud by proposing a revenue maximization framework. In the proposed framework, the revenue maximization problem has been formulated as a stochastic dynamic program. Also, their framework provided the optimality conditions and significant structural results of the optimal pricing policies. Furthermore, the model has been extended to include general non-homogeneous demands. Instead of the dominant static pricing strategy, their model took into consideration that prices are charged per instance per time unit.

2.3.4 Datacenter energy consumption

In current cloud computing systems, the energy consumption of underutilized resources accounts for a substantial amount of the actual energy use. Inherently, an efficient resource allocation strategy that considers resource utilization should increase the system's energy efficiency. Recent studies reported that energy consumption scales linearly with resource utilization. This fact highlights the significant need for techniques to improve the resource utilization and, in turn, reduce energy consumption in the system. Datacenter power is consumed by servers, network devices, power distribution equipment, cooling infrastructure, and other supporting infrastructure [18]. As a result, there is a significant increase in the overall cost of datacenter operations. Energy usage minimization has become an important objective for CSPs. Over the past few years, there have been several attempts to reduce energy consumption in datacenters. These attempts included developing low-power components to improve hardware energy efficiency, developing techniques for energy-aware resource management, deploying more efficient cooling systems, and exploiting the geographical areas with appropriate climatic conditions for deploying datacenters.

Energy efficiency is an emerging research challenge that was recently addressed by several researchers. For example, Khan and Ahmad in [99] used game theoretical methodologies to optimize system performance and energy consumption simultaneously. Since then, many research works have used similar models and approaches. These approaches addressed a mix of research problems related to large-scale computing systems, such as energy proportionality, memory-aware computations, data-intensive computations, energy-efficiency, grid scheduling, and green networking ([100], [101], [102], [103], [104]). Cloud computing and green computing paradigms are closely related to each other and have gained more attention than before. Advances in hardware technologies such as low-power CPUs, solid-state drives, and energy-efficient computer monitors have relieved the energy issue to a certain degree.

VM consolidation [105] and VM migration are popular techniques that perform an efficient energy management. The existing VM consolidation approaches, e.g., [106], [107], [108], and [109] are used in datacenters to reduce the under-utilization of physical machines and to optimize their energy efficiency. VM consolidation has mainly relied on live VM migration [67] to consolidate VMs periodically, so that some of the unloaded machines can be terminated. Also, VM live migration provides the ability to adapt resource allocation dynamically. Dynamic resource allocation can be achieved by moving VMs to new physical machines with sufficient resources when the demand increases, or by consolidating VMs on a smaller number of physical machines to minimize energy usage.

The existence of large virtualized datacenters as a response to the rapid growth in computational power demand is driven by applications that have recently emerged. Such datacenters possess high operating costs due to enormous amounts of electrical energy consumption. In that context, many solutions have been proposed in the literature aimed at minimizing energy consumption in these datacenters. In [110], heuristic solutions are presented for dynamic VM consolidation, these solutions aimed to minimize energy consumption considering SLAs, particularly CPU performance. These adaptive heuristics are based on an analysis of the historical data concerns VMs' resource usage. Their approach uses live VM migration to migrate the VM with the minimum migration time between hosts to minimize power consumption.

Goudarzi *et al.* [111] aimed to minimize the total power and migration costs in cloud computing systems under performance-related SLAs in a probabilistic sense; more specifically, upper bounds on response times for serving client requests. The authors solved this optimization problem using a heuristic algorithm based on convex optimization and dynamic programming. In [111], Goudarzi *et al.* proposed a flexible energy-aware framework that considered the energy-aware allocation/deallocation problem of VMs. The proposed optimizer relies on Constraint Programming (CP) to compute a configuration of VMs that minimize the power consumption. Their approach also uses branching heuristics to

instantiate variables that guide the solver to a near-optimal solution.

Mastroianni *et al.* in [112] considered the energy-related costs issue in datacenters. Thus, they presented *ecoCloud* which is a self-organizing and adaptive approach to VMs' consolidation. Based on local information, probabilistic processes are responsible for decisions making on VM assignments and migrations. Also, Zhen Xiao *et al.* in [113] presented a resource management system for cloud services which is used to allocate datacenter resources dynamically based on application demands. Their work includes the design, implementation, and the evaluation of the proposed system. Further, their proposal optimized the number of servers used in order to support green computing. In their algorithms, they combined VMs with different resource characteristics to physical resources by adopting the skewness metric.

Cloud environments consist of large-scale distributed datacenters across multiple geographical regions. The cost to operate these huge environments is typically dominated by electrical energy bills. Recently, attention was directed to workload management in Geo-distributed clouds as a solution for CSP cost reduction, and also, CSP benefiting from lower energy prices in dispersed locations. Public cloud environments have been realized via resources housed in datacenters that are dispersed geographically and managed by a single organization. Research work including [114], [115], [116], [117], [118], and [86] has proposed algorithms designed to reduce overall costs. In these proposals, datacenters were deployed geographically where the electricity prices are cheap or they have scheduled their workloads for the times with the lowest electricity prices. Qian and Rabinovich [119] addressed the combined problem of provisioning application instances across a Geo-distributed cloud and distributing customer requests to these instances using routing policies. They proposed a novel demand clustering approach that scales to realistic system sizes.

2.4 Cloud Management Systems State-of-the-Art

This section will introduce existing commercial cloud management products and the open-source systems that have been developed. In the academic field, many open-source cloud management systems have been developed in order to build private, public, and hybrid clouds, such as OpenNebula [8], Eucalyptus [9], OpenStack [8], and CloudStack [20].

2.4.1 Open-source Cloud Management Systems

This section examines the main features of the existing cloud platform solutions and presents an elaboration on the support of cloud and network resource control and management. There is also an analysis from the flexibility point of view of how the applications' required specifications are supported in existing cloud management systems and how the operator can influence the allocation of a resource on such platforms. In addition, some major open source platforms that are currently available are discussed, e.g., CloudStack, OpenNebula [8], Eucalyptus [9], and OpenStack [8].

CloudStack

CloudStack [7] is a mature open source IaaS solution that is governed by the Apache Software Foundation [120] and currently supported by Citrix. CloudStack has been designed to consider scalability and centralized management in the design. Supporting major hypervisors on that system differentiates this system from other existing systems. CloudStack architecture is strictly a hierarchical structure that enables the solution to scale to many thousands of physical machines from a single management interface. Figure 2.10 represents the architecture of CloudStack. This architecture is mainly based on three tiers:

- Zone: The largest organizational unit within CloudStack, typically a datacenter, will contain a single Zone. This enables geographical zoning of resources and it allows

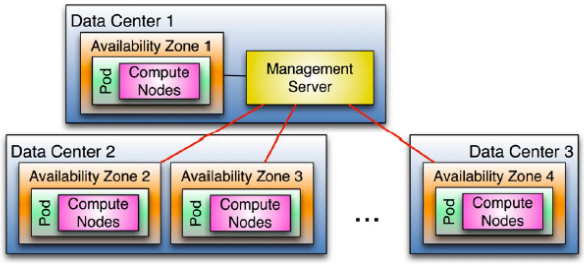


Figure 2.10: CloudStack architecture [7]

data to be placed at specific locations to comply with an institution's policies. A Zone consists of at least one "Pod" and Secondary Storage component.

- Secondary Storage: Is used to store VM templates and enable data replication between Zones, providing a common storage platform throughout a Cloud. The components make use of the Network File System (NFS) protocol to ensure network access by any host within a Zone.

- Pod: Is hardware configured to form a cluster of resources. A Pod is typically comprised of a rack of servers and shared networking hardware. Pods are not visible to end users.

- Cluster: Is a group of host machines running identical hypervisors. Each Cluster has a dedicated Primary Storage device in which VM instances are hosted. A Cluster provides high availability and load-balancing features.

- Primary Storage: Is built using high-performance dedicated hardware to accommodate concurrent access by multiple VM instances. The component is designed to provision an instance with storage using standard compliant iSCSI and NFS protocols.

CloudStack was developed in Java and provides a Management Server component backed with a database to store the cloud state. This enables the management of resources via a web interface, command-line interface, or REST-based API.

OpenStack

OpenStack [8] is a collection of open source components to deliver public and private clouds. These components currently include three main projects: OpenStack Compute (Nova), OpenStack Networking (Quantum), OpenStack Object Storage (Swift), and OpenStack Image Service (Glance). Furthermore, there are OpenStack Block Storage (Cinder) and OpenStack Dashboard (Horizon). Figure 2.11 presents OpenStack component architecture

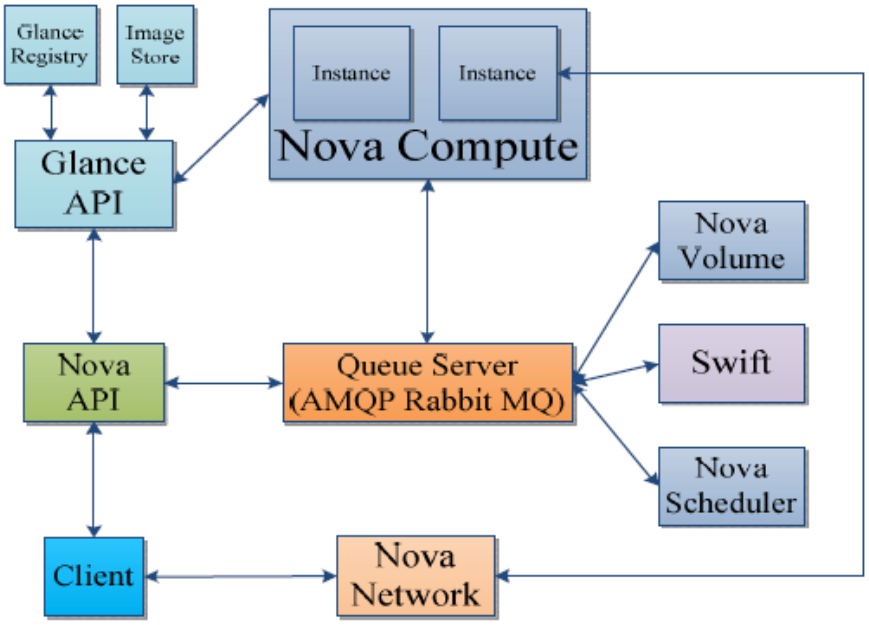


Figure 2.11: Part of OpenStack component architecture [8]

with the most important component highlighted, NOVA. NOVA is designed to provision and manage large networks of VMs. Also, NOVA handles creating a redundant and scalable cloud computing platform. It is considered to be the computing fabric controller for the OpenStack Cloud. NOVA has six components including NOVA-API, Message Queue, NOVA-Compute, NOVA-Network, NOVA-Volume, and NOVA-Scheduler. NOVA can be viewed as the main management platform in OpenStack; it controls compute resources, networking, authorization and scalability. The architectural components follow a shared-nothing and messaging-based approach. In this approach, each component or each group

of components can be installed on any server. Queue Server is in the middle of the architecture in order to enable the communication between the cloud controller and other components via AMQP. All the communications are using what is called asynchronous eventually consistent communication, to transport messages. This mean of communication allows a callback to be triggered once a response is received and prevents users' actions from being stuck in a waiting state for a long time. Swift is a scalable object storage system including a Proxy Server, an Object Server, an Account Server, a Container Server, and the Ring. It is a long-term storage system for permanent types of static data that can be retrieved, leveraged and updated. Swift's main functions and features are secure storage of large numbers and sizes of objects, data redundancy, archival capabilities, and media streaming. Glance provides discovery, registration, and delivery services for virtual disk images.

Eucalyptus

The next critically-appraised IaaS Cloud architecture is Eucalyptus [9], an IaaS system that aims to create an open-source infrastructure. This infrastructure is architected specifically to support cloud computing research and infrastructure development. Eucalyptus implements an IaaS private cloud that is accessible via an API compatible with Amazon EC2 and Amazon S3. The architecture of the system is simple, flexible, and modular in a hierarchical design that reflects common resource management environments found in many academic settings [121]. Users can start, control, access, and terminate the entire VM using Amazon EC2's interfaces. Consequently, Eucalyptus users who are familiar with Amazon EC2 interfaces can easily interact with the system using the same tools and interfaces. Each high-level system component is implemented as a stand-alone web service. Using web service implementations brings benefits to the system. First, each web service exposes a well-defined API represented in the form of a WSDL document containing both the defined operations that the service can perform and input/output data structures.

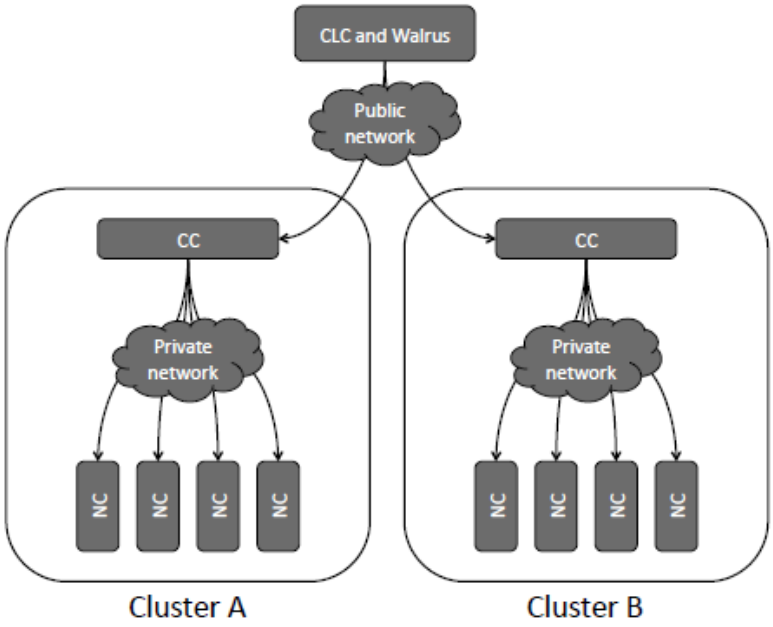


Figure 2.12: Eucalyptus main architecture [9]

Second, existing web-service features can be leveraged for secure communication between components. There are four high-level components represented in figure 2.12, each with its web-service interface, that compose Eucalyptus installation [121]:

- Node Controller: controls the execution, performs inspection, and terminates VM instances on the hosting machines.
- Cluster Controller: gathers information and schedules VM execution on specific Node Controllers. Also, manages the VN instance.
- Storage Controller (Walrus): is a storage service that implements Amazon's S3 interface. Walrus provides a mechanism for storing and accessing VM images and the users' data.
- Cloud Controller: the entry point into the cloud for users and administrators. This Controller queries Node Controllers for information about the available resources,

makes high-level scheduling decisions, and implements their decisions by making requests to Cluster Controllers.

One of the primary benefits of using Eucalyptus over other IaaS implementations is that it provides users Amazon EC2 (compute) and S3 (storage) service APIs, but has also limited novelty within the project.

CloudNaaS

Benson *et al.* [21] presented CloudNaaS, a system that allows cloud customers to deploy virtual infrastructure augmented with a set of network functions such as virtual network isolation, custom addressing, service differentiation, and flexible positioning of middleboxes. CloudNaaS leverages techniques such as software-defined networking to provide flexible and fine-grained control of the network in an environment as large and dynamic as a cloud.

OpenNebula

OpenNebula [10] was introduced as a research project in 2005; it is much earlier than OpenStack [8]. OpenNebula has become a prominent implementation of an open-source IaaS

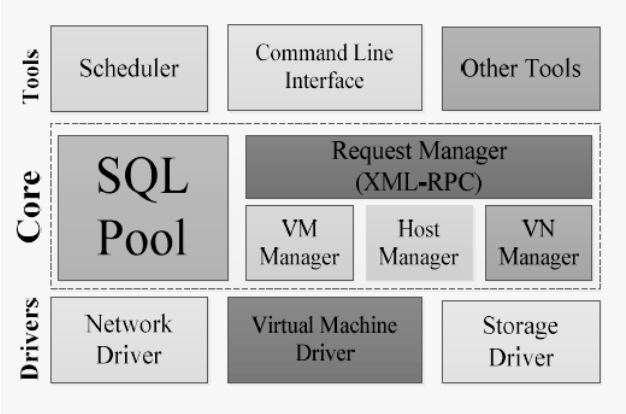


Figure 2.13: OpenNebula component architecture [10]

framework. This framework aimed to provide a flexible, open, scalable and comprehensive management layer to manage datacenters and simplify its operation [8]. OpenNebula architecture has been designed in a modular structure that allows the extensions to be added by external developers easily and has been tested on large scales with many thousands of nodes. The flexible and modular architecture makes it easier than OpenStack to integrate multiple storage disks, network infrastructures, and hypervisor technologies. Figure 2.13 introduced OpenNebula architecture which includes three layers: Drivers, Core and Tools. The core of this architecture is comprised of a number of components that control and monitor resources like VMs, VNs, storage, and PMs. These components are:

- Request Manager: This component is responsible for handling customer requests. The component exposes an XML-RPC interface with some methods for managing resources.
- VM Manager: This component manages and monitors VM resources and the operations it provides are abstracted from the underlying Hypervisor technology used.
- VN Manager: This component handles the assignment of IP addresses, enabling the dynamic creation of VNs by tracking IP leases.
- Host Manager: The monitoring of PMs is managed by this component via suitable drivers that can be extended to monitor any host attribute.
- SQL Pool: A persistence database for storing OpenNebula internal data structures that represent the state of a resource pool. This component enables reliability in the case of a failure or system restart.
- Scheduler: Resource scheduling is performed by the scheduler component. This component is independent of the rest of the architecture and communicates with the core components using XML-based remote procedure calls. This enables decoupling

so that the scheduling algorithms and mechanisms used can be tailored or changed to a specific provider's needs.

2.4.2 Commercial Cloud Management Systems

Understanding specific problems related to the QoS in commercial clouds is a difficult task, since the provided services are not transparent, and the customer has no idea of the underlying implementation. This case is similar to the SOA approach, where only the functionality of service is exposed by a defined interface. The majority of major commercial cloud vendors provide best-effort QoS. This provides a primary motivation to researchers to address the QoS in the context of cloud computing service provisioning and resource allocation, furthering the adoption of the cloud paradigm. Also, due to the closed-source nature of these commercial clouds, limitations are presented concerning interoperability and flexibility. Such limitations attract the attention of research to investigate the development of cloud standards that enable sharing of cloud resources outside administrative and organizational boundaries. These standards could also encompass QoS specifications that realize standard interfaces for communications and descriptive languages to define cloud services. This section contains a brief discussion of commercial adopters of cloud technology, including the services and software products provided.

1. Amazon

   Amazon is the first company that provided cloud infrastructure services, via its Amazon Web Service [122] products in early 2006. Amazon provided a PaaS architecture on a pay-per-use financial model. The architecture is marketed as two individual products: the Amazon Elastic Compute Cloud (Amazon EC2); and the Amazon Simple Storage Service (Amazon S3). These products provide a set of well-defined APIs that have become widely adopted as standards in many open source cloud architectures, such as Eucalyptus [9] and OpenNebula [8]. These projects provide in-

terfaces compatible with Amazon's services to enable on-demand scale-out of service workloads to supplement local resources and satisfy peak or fluctuating demands.

2. Google

   Another contender positioning themselves as a provider of cloud services is Google. Google provides SaaS via its Google Apps software and a PaaS via its Google App Engine [123]. The Google App Engine provides the architecture that Google Apps run on and promises transparent scalability on a pay-as-you-go charging model. The Google App Engine is limited to a subset of Python APIs and provides a proprietary data storage query language that limits its applications.

3. Microsoft

   Microsoft provides cloud services using the Azure Services Platform [124], a PaaS operating system, which integrates many of Microsoft's current proprietary software packages into one, via a middleware layer. This integration can be utilized by licensed cloud vendors and is marketed as an all-in-one cloud software solution.

2.5 Summary

Although cloud computing has only recently appeared, it has evolved quickly. Cloud computing has gained much attention in the industry; many enterprises moved their business to a cloud computing environment to take advantage of its proprietary features and to reduce their establishment cost. CSPs aim to increase their revenue and to provision their service in a dynamic and adaptive way to fulfill user's custom requests. This chapter introduced a brief background for cloud computing as an emerging paradigm, including a definition, service models, and deployment models. Furthermore, cloud computing enabling technologies have been discussed briefly. Related work on resource allocation and optimization in datacenters has been presented. Also, energy-consumption-related work was also discussed. Comprehensive research has been conducted on similar existing cloud

systems for both commercial and open-source systems. The limitations and drawbacks in similar existing systems were discussed to enumerate the associated challenges and define our scope for potential research work.

Chapter 3

IaaS Resource Management Framework Architecture

The previous chapter introduced some related work that exists in regard to cloud computing resource management systems, in general. The need for a generic, flexible, efficient, scalable, and reliable cloud management system through which both service providers and customers acquire maximum satisfaction was introduced. CSPs are seeking profit maximization and optimum resource utilization, while customers are aiming for low cost and efficient cloud service with acceptable QoS. This chapter gives an overview of the proposed Cloud IaaS Resource Management Framework for that purpose. The framework is designed in a layered architecture that assists CSPs in (1) managing their infrastructure resources efficiently, (2) maximizing their revenue, and (3) provisioning an efficient IaaS while respecting customers' QoS requirements. This architecture with the associated modeling and algorithms aims to automate and simplify the cloud IaaS resource management process.

This chapter proceeds as follows. Section 3.1 presents a general overview that discusses the emergence of cloud computing. Section 3.2 illustrates some of the limitations of similar existing systems. The design objectives of the layered architecture are discussed in section 3.3, while section 3.4 gives a detailed elaboration on the proposed architecture's different layers. In Section 3.5 a breakdown illustration is given that describes the different components of the management layer and the deployment use case scenarios that explain the proposed framework's operational parts. Finally, in Section 3.6 the chapter concludes with a brief summary.

3.1 Overview

The emergence of cloud computing as a new cost-effective computing paradigm, stimulates many enterprises to move their work to the cloud. Cloud computing benefits CSPs and customers as well. Cloud computing is credited with lowering the enterprise expenditure on the establishment of infrastructure. This expenditure reduction minimizes CSPs' costs for the provisioned services and so increases their revenue. Furthermore, CSPs aim to deliver efficient cloud services for their customers in a seamless way with QoS guarantees, while customers are always wanting to pay less money for their requested services. All these guarantees are limited and recorded in predefined SLAs between CSPs and their customers. The SLA serves as a contract between the customer and service provider that arrange this relationship and guarantees the QoS level of the provisioned service. Cloud computing achieves the customers'/CSPs' cost requirements through the pay-as-you-go characteristic and a utilitarian basis. Managing such multi-objective requirements still challenges service providers. The framework resulting from this research aims to help CSPs overcome their management challenges, increase their profits, and use their resources efficiently, while respecting their customers' SLA. Furthermore, this framework reassures customers of a flexible interface, lower cost, and efficient cloud services.

IaaS as the fundamental layer in the cloud service model has gained more attention recently in academic research because of its compatibility with many real-life applications and services. New computational paradigms like MapReduce have been developed to exploit the cloud infrastructure and its characteristics. In addition, more scientific and resource-intensive applications that have acquired this infrastructure have emerged. Consequently, the management challenges have grown and become more complex and costly. In order to cope with the growing market requirements, an efficient cloud computing management system is urgently required to assist CSPs in managing their resources efficiently.

In this dissertation, our interest is directed towards the IaaS cloud service model since it

is the fundamental layer of the service model, which provides the necessary infrastructures for hosting platforms and software. The following limitations and challenges of similar existing systems contributed to the design of the proposed architectural framework, such that these limitations were overcome and its design objectives were achieved.

3.2 Existing Management Systems Limitations and Challenges

Last few years, several open-source cloud management frameworks that provide IaaS services have been developed such as OpenNebula [125], Nimbus [126], CloudNaaS [21], OpenStack [8], Eucalyptus [9], and GEYSERS [22], existence of such management frameworks facilitate the creation of private clouds for different enterprises. When studying these systems, the following limitations are observed:

- Flexibility: Existing systems received a complete IaaS request from the customers, either by selecting a saved predefined configuration or by precisely specifying all the request components. This is carried out by using developed scripts that include the connection topology of the customer's request components, e.g. hosting N-tier web applications like in CloudNaaS [21]. These developed scripts allow a CSP to achieve its operation easily. Other systems have developed user interfaces that are unfamiliar to unskilled customers (e.g. OpenStack and OpenNebula [8]). Such systems expect a certain level of experience.

- Diversity/heterogeneity of the managed resources: Existing management systems repositories are comprised of diverse resources including compute, storage, and network resources. The enormous number of managed resources supplied from different manufacturers adds additional challenges in efficient management. Moreover, the integration of network resource manipulations in such systems is still in its early stages and does not provide a mature product, e.g., OpenNebula [8] supports creating VNs based on VLANs, and OpenStack Quantum component [8] provides an

API for managing the network connectivity between other OpenStack services. Existing management systems address homogenous resources, but GEYSERS [22] has addressed non-homogeneity and has built an information model for their resources.

- Scalability: Given the growing demand in computing power, existing systems need to be scalable with the increasing number of resources and able to continue their operation despite system component failures. However, centralized architecture is the dominant architecture in most of the existing systems; all these frameworks possess a high degree of centralization that impedes the system tolerance toward different component failures. For example, the scalability of OpenNebula, Nimbus and OpenStack is limited by their node/cloud controller. Moreover, since there is no replication support exists, a failure of the front-end node impacts VM management severely. Similarly, the same drawbacks are still exist in Eucalyptus for each non fault-tolerant cluster controller [127]. As the number of IaaS requests increases and as the number of managed datacenters increases, a scalability issue occurs. Lack of scalability in such systems' architectures directly impacts the existing system performance and the CSP's revenue.

- Multiple Controllers: Since network management in existing management systems is still in its early stages, it was noticed that most of the existing systems have two controllers in their design; a network controller and a cloud resource controller, e.g., a module of OpenStack called Quantum [8] is responsible for managing the VNs in OpenStack-based cloud platforms. Also, CloudNaaS [21] defined a separate controller for network management. As a consequence of the two controllers, a differentiation occurs in treating different cloud resources (compute, storage, and network). This differentiation in treatment with the diversity and heterogeneity of the managed resources makes the management operation very difficult and significantly impacts the efficiency of the provisioned services and the reliability of the cloud systems.

As a result, systems may collapse even at small scales of operation. Furthermore, these systems require additional communication and negotiation between different controllers to synchronize their operations. These overheads occur at different scales of management and affect the performance of the management framework operations.

In addition to the observed limitations in existing systems, CSPs continually keep an eye on how their systems guarantee the required QoS for customers. Service providers are always challenged by: (1) the efficient management of huge numbers of diverse and heterogeneous resources regarding those resources' utilization and energy consumption, (2) satisfying and maintaining the SLA with the customers without penalties, and (3) improving their reputation and increasing their profits. While customers are continually worried about the cost of the provisioned services, they are also looking for a flexible system that respects their skills. Throughout this dissertation, we aim to design an efficient management framework that overcomes the aforementioned limitations and challenges. Furthermore, we aim to provide a set of resource allocation models and algorithms that solve any issues arising from these limitations with the aim of satisfying both the customers and the CSPs.

3.3 Proposed System Objectives

For an efficient architecture that overcomes the limitations of similar existing systems and overcomes most of the existing challenges, CSP frameworks should take the objectives of different stakeholders' into consideration. We consider the following requirements for the architectural design of the proposed system:

- Flexibility: The proposed architecture design addresses the experience level of both CSPs and customers. From the CSP's management side, the modularity and layering of the architecture provides the CSP with flexibility to manage and maintain its framework efficiently. From the customer utilization side, the desired framework allows the customers to customize their requests, and easily express their requirements

through conceptual modeling. In addition, the proposed framework employs tools that allow manipulations of the uncompleted requests. This dissertation's framework is quite sufficient for unskilled users and requires no experience.

- Unified and Efficient management: It is required that the architecture be managed by a single controller, particularly for diverse and heterogeneous resources. Existing systems define multiple controllers for different resources (i.e., individual network controllers). The proposed architecture includes unified management that solves many problems regarding communication and negotiation overhead. This unified management is achieved by defining a combined controller that unifies the allocation of cloud resources and network resources evenly, alleviating the burden of coordination, communication, and synchronization between different controllers inside the managed datacenter. For efficient management, the combined controller employs a set of efficient resource allocation approaches with different models and algorithms. This set of working solutions positively impacts the effectiveness of the designed architecture. An efficient management architecture should allow the implementation of all the possible scenarios without using closed proprietary technologies. In this sense, different designed resource allocations solutions can be tested through our framework.

- Scalability and Extensibility: At the design level, the management framework is designed using a modular layered architecture with components on each layer, allowing the extensibility of the architecture at any layer by integrating more layers or adding more components. As a consequence, the modular architecture provides the ability to cooperate with other CSPs. At the working solutions level, the proposed architecture allows CSPs to test and integrate different resource allocation approaches that work at different scales. Furthermore, the designed working models and algorithms employ tools and techniques that support a growing scale of both the managed resources and the infrastructure requests. Furthermore, the working solutions address coordi-

nation and communication among multiple infrastructures in distributed deployment scenarios.

To sum up, the proposed framework is designed using a layered architecture that allows the extensibility of the framework. The proposed framework aims to: (1) assist CSPs in efficient management of their assets by utilizing a unified representation and a combined controller, ensuring the uniform treatment of cloud resources, (2) guarantee the performance of the provisioned infrastructures and ensure the customer's SLA is satisfied through employing efficient modeling and resource allocation algorithms, and (3) realize the cloud computing cost-efficiency characteristic, by employing efficient economical models that guarantee monetary benefits for both CSPs and customers.

3.4 The Proposed Framework Architecture

The evolution of recent cloud computing applications that require IaaS has raised a challenge for CSPs of how to provide a robust and efficient IaaS for these applications. The desired service is comprised of matched QoS cloud resources for the application components with a network QoS guarantee between these components. The IaaS service provisioning process passes through many steps in an effort to perform efficiently. Optimizing the IaaS service provisioning life cycle to achieve the desired objectives involves an enormous amount of work. In this section, prior to the provisioning of a QoS-assured service, the architecture of the proposed cloud IaaS resource management framework is presented. The system architecture is designed in such a way that it allows the allocation of resources to achieve its management objectives; these objectives are specified by the CSP by taking into account the benefits to both the CSP and to the customers. As shown in Figure 3.1, the proposed architecture is arranged in a layered architecture named from bottom-up, as follows:

1. Physical Layer

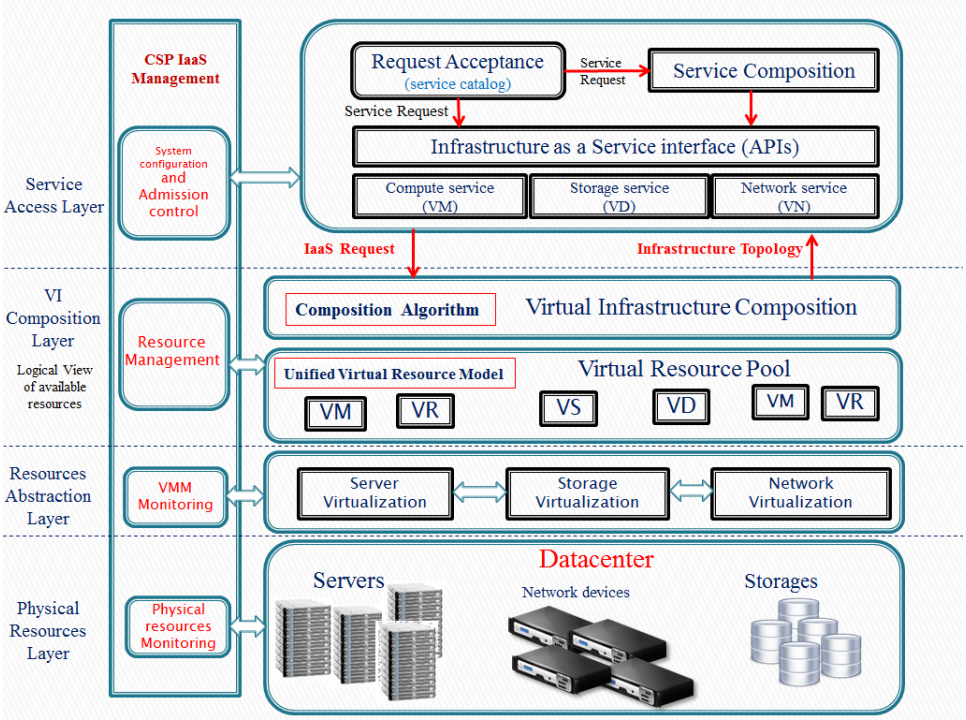


Figure 3.1: Proposed layered architecture

2. Virtual Abstraction Layer

3. Virtual Infrastructure Composition Layer

4. Service Access Layer

5. Management Layer

In this framework, we define two stakeholders, the customers who are interested in renting VRs in order to deploy their applications and the CSP who owns these resources and is interested in provisioning them to many customers in the most cost-effective way.

3.4.1 Bottom-up Approach

In order to maintain the SLA contract with customers without any violations, the bottom-up approach was used in the architectural design [128]. This strategy reduced the risk that SLAs could be compromised, as the possible SLAs are always bound by the currently

available resources at the physical level, the lowest layer of the architecture. Furthermore, this approach removes the complexity of the negotiation and constant recalculation of possibly available resources at the different layers. In addition, this approach also moves the complexity of the allocation approaches away from the physical resources, such that the provisioning time for a virtual infrastructure is low. Other approaches also exist, e.g., the top-down strategy. However, top-down moves the complexity of the advanced provisioning algorithms to the physical layer. There is a higher risk of the physical resource layer not being capable of responding rapidly to demands, or not being able to resolve the mapping of VRs to physical resources. Furthermore, the CSP cannot offer guarantees to the customers as there is no prior knowledge of the lower level status.

In our design, the bottom-up strategy is adopted, where a pool of VRs is prepared in advance before customers explore the services and submit their requests. As a result, the lower level status is known prior to receiving requests. By following the bottom-up approach, the complexity of management moves away from the lower layers to the upper layers. Also the abstraction level is increased. Consequently, a trade-off exists between the complexity of management and the abstraction level. In accordance to that trade-off, the most important layer in this framework is the layer where the complexity is manageable and the abstraction level is sufficient for efficient IaaS provisioning. These features are achieved in the ***Virtual Infrastructure Composition Layer***.

3.4.2 Physical Layer

With the proliferation of cloud computing as a new IT paradigm, CSPs rely on datacenters as repositories for diverse resources and services. Currently, datacenters have been deployed as the mainstream hosting platform for cloud resources and services [129]. Thus, datacenters have experienced a shift towards exposing their capabilities as a service. In our architecture, the physical layer is described as a datacenter facility that is comprised

of physical resources like servers, storage and network devices. This datacenter is owned and managed by the service provider, and the number of available resources depends on the size of the datacenter. The most important component in the datacenter is the set of network devices (e.g., switches, routers, and cables) and its communications infrastructure, as shown in Figure 2.4 [4]. In addition, datacenters contain power plants and cooling systems. The underlying physical layer can be organized and managed in different ways: it can be a single datacenter or multiple datacenters. As the number of datacenters increases, the complexity of the resource allocation and management increases. Managing multiple-datacenter cases adds more management challenges for the service providers and additional constraints for the applied resource allocation solutions.

Deployment of traditional datacenters becomes a source of disturbance and suffering for CSPs. Because of the resource fragmentation problem in traditional datacenters, the problem of insufficient resources occurs. Consequently, the poor resource utilization problem arises in these datacenters. Virtualization techniques solve the issue of poor resource utilization and problem of fragmentation in traditional datacenters and create the virtualized datacenter. Virtualization techniques in traditional datacenters lead to a flood of diverse VRs which introduces new management challenges to CSPs in managing their repositories. Conceptual representation of these diverse resources can be a solution that allows for proper management.

3.4.3 Resource Abstraction Layer

In addition to the poor resource utilization introduced in traditional datacenters, the physical resources at the lowest level are the most complex to operate and share among multiple users. The physical layer is comprised of heterogeneous diverse resources, where several existing hardware details do not need to be visible to users, so a level of abstraction is needed. The most significant enabling technology that differentiates cloud computing from preced-

ing paradigms is using the virtualization technique. The existence of the virtualization technology, e.g. VMware and KVM, plays a vital role that drives cloud computing and makes it the key enabling technology for cloud computing. Virtualization techniques apply statistical multiplexing in realizing the resource sharing and resource isolation that helps in giving the impression of infinite resources and achieving the on-demand cloud characteristic. Establishing a virtualized datacenter ensures the resource isolation and a performance guarantee for customer requests.

In this layer, different virtualization techniques for cloud resources and network resources are applied. These techniques are used in preparing the datacenter resources, cloud resources and the network resources for the upper layer operations, the virtual infrastructure composition layer. The main functionality of the resource abstraction layer is to convert the physical resources to VRs that possess all the features of the physical resources. This conversion is achieved by partition and aggregation techniques in virtualization [130], and it is applied for cloud resources and network resources as well. The abstraction level of the created VRs is defined following a unified resource model and the granularities of the abstracted resources respecting the predefined classes of services [131].

3.4.4 Virtual Infrastructure Composition Layer

This layer is the core layer in this architecture, and it mainly works on resource-level manipulation. Collecting all the created VRs from the abstraction layer into a repository announces the creation of the pre-prepared virtual resource pool. This layer is considered to be the most important layer in the architecture. The trade-off between the complexity of management and the abstraction level is treated at this layer. A VR is an intermediate abstraction level between the physical resources on the lower layers and the service API on the upper layers. These VRs can be easily manipulated and managed at this layer. Working at the resource level gives more flexibility in resource manipulation. As the abstraction

increases while moving up, it is better to manipulate VRs than physical resources. Also, for efficient resource management, manipulating resources at the virtual level contributes to better management and gives more flexibility in treating resources.

Accordingly, in this framework design, attention is given to coping with CSPs' challenges regarding the diversity and heterogeneity of the available resources. As a result, a unified ontology model for the datacenter's diverse resources is proposed. This conceptual modeling allows the application of the associated ontology capability to support efficient resource management. Indeed, the advantage of this ontology model is to ensure the flexibility and the simplicity to extend new concepts and properties for datacenter resources and supports the usage of reasoning engines as well. In this layer, the defined unified VR model abstracts the physical layer resources. We consider in that model sufficient information from physical resources to ensure management efficiency. The generated VRs are then collected to create the virtual resource pool.

In this layer, the allocation and de-allocation of VRs on demand is achieved by employing several resource allocation approaches with different aims and objectives. Based on the deployment scenario, the generated virtual resource pool can represent the VRs of a single datacenter or multiple datacenters in a region according to the cloud management framework coverage area.

This layer is comprised of two main modules, the first being the virtual resource pool, which represents the conceptual modeling of all the available resources in the managed datacenter stored in an ontology repository. Over the virtual resource pool, the second component exists, which is the virtual infrastructure composition module. The design of this layer is mainly based on a composition technique. Proposing composition as a main technique in the resource management framework stems from the successful applicability of such techniques in web services and VNs [132]. Furthermore, growing complexity and customizable customers' requests encourage the usage of the composition strategy in IaaS provisioning. Service composition is the process of providing a composite service by orches-

trating a set of existing services in a way that satisfies the user requirements. The aim is to acquire the benefits of the composition that gives more flexibility and stability in resource allocation since the customer requirement is customized without topological information. It is worth mentioning that the nature of the cloud service differs from existing services like web services, but there may be some common features. The composition strategy was applied in the designed architecture at the resource level, while exploiting different techniques that are compatible with the virtual resource pool modeling, the datacenter topology, and the customer's QoS requirements.

3.4.5 Service Access Layer

This layer is the upper layer in the proposed architecture exposed to the customers' IaaS requests; it is concerned with the direct interaction with the customers, receiving and provisioning services. It contains a function module for SLA negotiations with customers; the SLA is defined mainly as the customers' basic requirements represented as QoS requirements. The required functionalities in this layer are the service publishing catalog, the admission control service, and the request translation service. This layer is responsible for publishing the CSP services in a catalog. The customer explores the published catalog and submits his request based on the published services.

Describing an IaaS request on this layer can differ from one customer to another due to the different level of details each customer requests and the customer's skill level. For example, a customer may specify separately each virtual link between two virtual nodes and its required attributes, or he may apply for a certain topology like mesh or star with the attributes for all the virtual links [132]. Therefore, a generic IaaS request model is required on this layer in order to represent the customer's requirements so that many IaaS requests with different levels of details can be represented using this generic model. To this extent, two types of customers can be defined. (1) An expert customer who represents

an enterprise: This type of customer asks for IaaS infrastructure to deploy the enterprise's applications, or uses another service provider who acts as a broker that uses the IaaS services and rents it to end users. (2) An individual user, who asks for an IaaS service to run an application that follows a MapReduce computational model, a scientific application or a N-tier application. In the latter case the customer is not required to be experienced. For both types of customers, a customized request is allowed. The composition technique used in the lower layer is compatible with both customer types and serves their customized requests efficiently. After composition, the composed virtual infrastructure from the lower layer is encapsulated into IaaS service and exported back to the customers. This is carried out in the form of the service's API using web service technologies.

3.4.6 Management Layer

The cloud management vertical layer provides configuration, monitoring, and administration of the cloud IaaS framework to keep the whole system in a stable, reliable, and efficient state. The management layer is responsible for synchronizing the execution of different components in the architecture, managing the interface between layers and different modules, and orchestrating the architecture modules' operations. In addition, this layer is responsible for different tasks concerning the resource allocation operations including resource allocation management (enforcement of quotas), scheduling, monitoring, load balancing, and resource usage collection. This is all carried out without causing much overhead in the underlying system. All these responsibilities add more challenges to the framework in order to achieve its desired goals, e.g., guarantee the availability, reliability, scalability, and the efficiency of the provisioned infrastructures. Furthermore, this management layer aims to achieve QoS-assurance and fault-tolerance requirements for each running service. This assurance involves all the services that are running across geographically distributed datacenters in cloud computing environments.

Figure 3.2 represents the conceptual view of the management functions in the proposed architecture. The management layer acts as the decision-making module for the CSP.

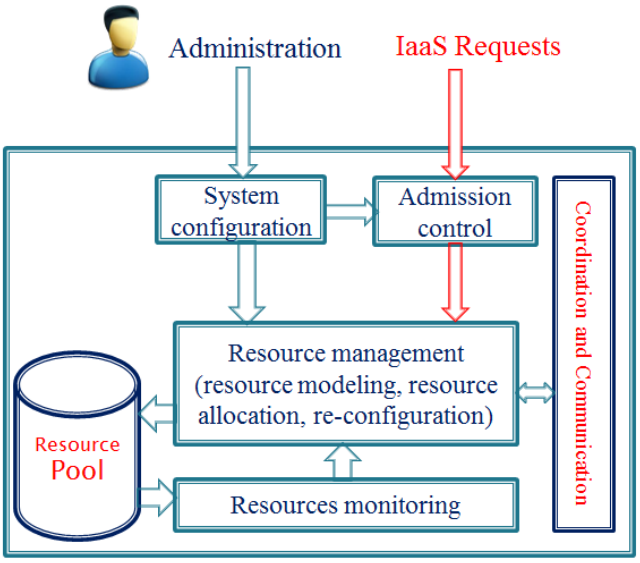


Figure 3.2: Functional elements in the conceptual management framework and the interactions between them

This figure summarizes the proposed functionalities for the management layer and presents the main components working on this layer. Details of the main modules follow:

- System configuration.

  This module provides a multi-objective management facility for the administrators. These multi-objectives cover the stakeholders' requirements from that design for both customers and CSPs. In this module, the CSP defines, in a timely manner, the set of objectives that make the system work efficiently. Different objectives can be applied in this management module. Furthermore, these multi-objectives are controlled and managed in a prioritized way. A CSP may choose to apply different objectives during different operational conditions, or optimize them in an arbitrary consistent combination [18]. For example, it can be used for implementing green computing, increasing the revenue, and reducing the cost of the provisioned services. Also, this administrative task contains threshold settings for many system parameters. Throughout this

dissertation, elaboration will be provided on different objectives imposed by the CSP regarding the resource allocation with various settings applied on the design.

- Admission control.

  This module realizes the defined administrative objectives on the received IaaS requests, as well as on different modules in the management layer. Based on defined administrative requirements, the admission control module automatically accepts the customers' requests that match the desired requirements.

- Resource management.

  This module defines a set of management functionalities concerning resource modeling, resource allocation, and the reconfiguration functionality. These module functionalities are responsible for managing the unified VR model which includes create, add/remove concepts and relations to the model. Furthermore, these module functionalities are responsible for managing the virtual resource pool by publishing more VRs and updating the status of VRs periodically. The re-configuration management functionality is concerned with handling the unpredictable changes captured by the resource monitoring module. Also, it reacts to any QoS degradation for the provisioned services in order to maintain customers SLA and keep the system in a stable state. The resource management module acts based on the administrative policies received from the system configuration module.

- Resource Monitoring.

  This module is continuously running and responsible for monitoring the virtual and physical resources available in the pool, and it triggers the re-configuration module when a performance degradation is detected, or any failure occurs. This module acts as a sensor for the re-configuration module. The resource monitoring along with the re-configuration module, realizes the system fault tolerance and confirms its reliability.

- Coordination/Communication:

  This module is responsible for managing the coordination and communication with others datacenters' management modules in a multi-datacenter deployment scenario. This module works properly to realize the case of cloud federation or distributed cloud resource management, through a designed coordination algorithm. Through the cooperation/coordination module and the management function interfaces, the CSP exchanges management information for decentralized, adaptive and efficient resource allocation in distributed datacenters.

As a conclusion for the proposed management layer, the management architecture is aimed at addressing a number of challenges arising from the heterogeneity of the available resources. Furthermore, unpredictable changes occurred in the managed datacenter states and rapid fluctuations in customer requests. Conceptually, the management functions can be instantiated in any layer of the proposed architecture. Also, these management functions consider the management operations in both single- and multi-datacenter cases. The CSP can implement these functions according to its desired management objectives and with respect to the managed infrastructure capabilities. In this dissertation, we aim to propose a management framework with a set of functions that are efficient with regard to their overhead, scalable with respect to the number of managed entities, and adaptive as well as robust, to all types of changes in the managed cloud. Furthermore, interactions between the management functions realize collaborative management solutions that to a high degree autonomously adapt to the dynamic conditions in the cloud systems.

3.5 System Components Interactions

The details of each layer in the proposed architecture are presented as well as the main components and their functionality in the management layer. The followed layering approach in the proposed framework design facilitates the interactions between the framework

layers, i.e., each layer interacts with the above and the lower layers no crossover is allowed during the IaaS provisioning life cycle. Furthermore, this role is also applied to the operation of the vertical management layer. Each component in the management layer is responsible for managing the corresponding layer in its level of the architecture, e.g., at the Physical Resource layer, the physical resource monitoring module is in charge of monitoring the datacenter's physical resources performances. Also, it is not allowed to perform a crossover management between the management layers' components. As part of the presented architecture, a use-case model is defined that describes some common functions for cloud environments. The defined use-case model provides the interactions between the architecture components and elaborates the operational steps of the framework execution in different scenarios.

3.5.1 System Use Case Scenario

The goal for this section is to define a general use-case that can cover as many scenarios as possible, and can be implemented in a standardized modular way to ensure interoperability, ease of integration, and portability of the proposed architecture. The defined use-case describes how a group of users may interact with one or more cloud computing systems, acquiring their resources to achieve specific goals. Use cases can help to identify possible flaws in the system by analyzing their activities. Also, use cases aim to identify actors that will interact with the system. An actor can be either an individual person with different experience levels or an enterprise. Prior to going through the use cases, the main stakeholders in the proposed architecture will be defined. This work defines two business roles: (1) the IaaS service provider who owns the datacenter, and is responsible for establishing the virtual resource pool, collecting the incoming requests, and configuring the cloud environment and (2) the cloud customer who explores the provider service catalog and submits an IaaS request.

The Standards Acceleration to Jumpstart Adoption of Cloud Computing (SAJACC) project at the NIST [133] and the Cloud Computing Use Case (CCUC) Discussion Group [134] define common use case scenarios for cloud computing. Based on the defined use-cases in NIST and CCUC, general functions are compiled that can be applied to all cloud services and the use-case that fits the proposed architecture is described. The focus in this section is directed towards designing a general use-case scenario that is concerned with deployment in a single cloud scenario, represented as a single-datacenter, and the multiple cloud scenario, represented as multiple datacenters. As a general use-case description, the following deployment case definitions are considered, as adapted from NIST:

- Deployment Case 1: In the centralized deployment cases, there is one CSP under consideration at a time. This provider is serving multiple cloud customers. Each customer has a simple client-provider interaction with the service provider.
- Deployment Case 2: In the distributed deployment cases, a single CSP deploys distributed datacenters and a single customer has an application that may be distributed across two or more datacenters and administrative regions, simultaneously. While the customer has a simple client-provider interaction with the provider, more complicated coordination may be required between both the customer and provider and also between the different administrative regions themselves.

3.5.2 IaaS Deployment Scenarios

Single-datacenter scenario

Based on the first use-case deployment model, the single-datacenter scenario is built. The pivotal actor in this scenario is the IaaS provider. Prior to when the provisioning process starts, two predefined preparation steps are stated, the virtual resource pool preparation and the service catalog advertisements. The customer explores the service provider's service

catalog to submit his request. The proposed system allows the customer to customize his request in the sense that he can use one of the published services in the catalog or customize a request that fits his needs. After submitting his request, the admission control module checks the eligibility of that customer to submit a request, performs identity checks, and the possibility of satisfying his constraints in the request. Also, the readiness of the system to accept the requests is considered. If it is a published atomic service, the service access layer responds immediately to that request and provisions the service directly to the customer. If it is a customized service, it will be forwarded to the next layer virtual infrastructure composition layer for composition. The monitoring module in the management framework keeps monitoring the physical resources and the VRs in order to trap any failures or performance degradation and acts immediately before the provision infrastructure is affected. In this way, the proposed architecture maintains the signed SLA items properly.

Multiple-datacenter scenario

The second deployment scenario defines multiple datacenters under the administration of a single CSP who owns and deploys these datacenters. Popular service providers such as Google and Amazon deploy multiple datacenters in different regions for many business considerations. The working environment on the multiple-datacenter scenario is comprised of datacenters that span geographical regions. The customer in his region explores the published service catalog from the service provider, and has the ability to define his requirement and customize his service based on his hosted application's requirements and budget. Implicitly, a customer request is forwarded to the nearest datacenter in his location unless he expresses his preferences for specific location. The received datacenter starts to allocate the infrastructure resources that are sufficient for the customer's requirement in regional datacenters. If the regional datacenters are not able to satisfy the customer request, an iteration of the negotiation phase starts among different regions to satisfy the customer

request efficiently. The monitoring module continuously monitors the physical resources and VRs in order to trap any failures and respond immediately before performance degradation occurs. Throughout this research, the proposed framework's models and algorithms will be evaluated with regard to their possible deployment scenarios as presented in this clause. This is carried out to determine the proposed models and algorithms efficiencies in serving IaaS requests and to decide what level they facilitate the management and assist service providers while respecting the customers' QoS requirements.

3.5.3 IaaS Provisioning Life Cycle

In order to complete the overall view of the working environment, the operational steps for the proposed framework will be defined in the provisioning life cycle. It is apparent that this framework serves a wide range of applications that require an infrastructure to run. Examples include: compute-intensive applications [135], data-intensive application [136], scientific applications, and applications that use the MapReduce model [137]. Through the defined scenarios, two types of requests are considered: a simple user request where, for example, the customer asks for an infrastructure service to host his N-tier application with QoS requirements; this type of application is considered as a short-term application. The second type of request is an enterprise application that requires a heavy, long-time infrastructure to host its service, such as a social networking, network gaming, or multimedia application, with highly specific QoS requirements. Figure 3.3 demonstrates the IaaS service provisioning lifecycle in a workflow chart, where the interaction between the proposed framework components is elaborated in an operational state.

The presented workflow chart is comprised of six phases, namely, (1) IaaS Request/Admission control, where the service provider receives the customers' IaaS requests and applies the predefined administrative policies as an admission control operation; (2)VI Composition: This phase includes the core operations in serving IaaS requests as represented in the re-

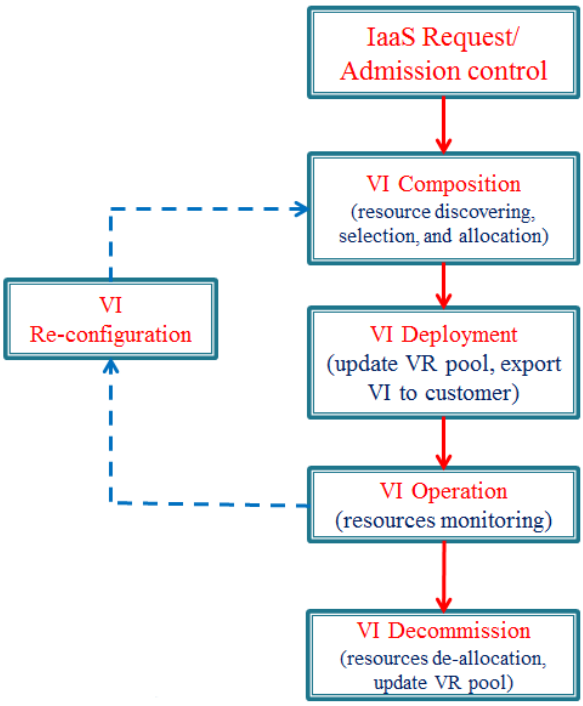


Figure 3.3: IaaS provisioning workflow chart

source discovery, resource selection, and resource allocation processes; (3) VI Deployment: is the after-composition phase, where the service provider provides the customer's VI as a service through APIs and updates the entire virtual resource pool; this update operation reflects the available resources for the next allocation periods; (4) VI Operation: This phase acts as an after- provisioning service, in which the service provider performs monitoring operations for the provisioned resources, as well as for the free ones, in order to keep the system in a stable state; (5) VI Decommission: This phase starts operation whenever the customer finishes using the provisioned VI, performs the VI resources de-allocation, and updates the virtual resource pool that reflects the available resources and (6) VI Re-configuration: This phase rarely operates; it is triggered by the monitoring phase indicating instability of the system state and directly responds to the received alerts from the monitoring phase in order to put the system back in a stable state. The operational rate of the VI Re-configuration phase reflects the system reliability and the efficiency of

the implemented algorithms and models in the proposed framework.

3.6 Summary

This chapter presents the layered architecture of the proposed cloud IaaS resource management framework in the cloud computing context. The benefits of the proposed work lie in applying a set of concepts and technologies in the design phase. Furthermore, considering multiple stakeholders' objectives and constraints in the design assists service providers in performing an efficient management. Building the architecture in a layered approach ensures the efficiency of the proposed management framework with regard to its extensibility and interoperability. Also, it provides the ability to test any developed resource allocation algorithms and to add any useful management modules for assessments. The multi-objective management module simplifies the management for CSPs and covers most of their needs. Moreover, the proposed framework design possesses salient features for CSPs such as scalability, fault-tolerance, and extensibility. This architecture provides features to customers as well.

Chapter 4

Two-Phase Ontology-based Resource Allocation Approach for IaaS Cloud Service

The Resource Allocation problem is still a critical problem for CSPs, and this type of problem has been continuously evolving from previous computing paradigms such as clustering and grid computing to the most recent paradigm, cloud computing. Finding an efficient solution is quite challenging and often computationally too expensive. Several new techniques that rely on heuristic approaches have been contemplated. Heuristic solutions are popular for their reasonable computational time and the approximate, near-accurate provided solution. In the previous chapter, a layered architecture for a cloud IaaS service provider was introduced. The architecture mainly relied on a VI composition layer as the core layer on the system, particularly, the composition algorithm.

This chapter presents an efficient resource allocation approach that adopts the composition algorithm as a technique. The presented composition approach relies on a unified ontology model for a VR; this model is populated to form a generalized repository that hosts diverse and heterogeneous resources. Exploiting the created repository, a two-phase resource allocation algorithm is proposed. This algorithm adopts two fundamental concepts in discovering and composing the required infrastructure: semantic similarity and closeness centrality. The proposed approach includes ontology-based modeling along with the designed algorithm confirming the applicability of the composition technique as an efficient resource allocation solution in a cloud computing context.

This chapter proceeds as follows. A brief introduction to the proposed approach is presented in section 4.1. Section 4.2 introduces an overview of the resource-based modeling state-of-the-art and the composition strategy. Section 4.3 introduces the key foundation concepts used in the proposed approach: virtual resource pool, semantic similarity, closeness centrality, and a random walk technique. The proposed approach is described in Section 4.4. In Section 4.5, the simulation setup and performance evaluation results of the proposed resource allocation approach are presented. Section 4.6 explains the presented approach limitations. Finally, Section 4.7 concludes the chapter.

4.1 Introduction

Cloud computing delivers its IaaS service in the form of integrating computation, storage and network resources in a virtualized environment. Incorporating network resources motivates the current research to give more attention to the network impacts in the provisioned IaaS service. A number of previous proposals have addressed resource allocation for IaaS services with bandwidth guarantees as the main obstacle to accessing datacenter-based cloud resources. These proposals focused on challenges adapted from VN provisioning. However, less focus has been put on the utilization of hosting resources, i.e. computing and storage. Oktopus [6] proposed a simple VN abstraction called the Virtual Cluster (VC) model where all VMs were connected to a virtual switch with links of fixed bandwidth and a performance guarantee. VC allocation was carried out by a designed greedy allocation algorithm. SecondNet [5] is another work that focuses on providing bandwidth guarantees among multiple VMs in a multi-tenant virtualized datacenter. This work proposed a VDC as the unit of resource allocation. VDC allocation was carried out using the bipartite graph and min-cost network flow methods.

Shortcomings with these proposals may cause (i) a high blocking of IaaS requests and (ii) a less efficient use of datacenter resources, which may impact the provider's revenue.

In the proposed approach, the suitability of adopting composition techniques stems from the customized privilege given to the customers in submitting their requests. In addition, the complexity of the submitted IaaS requests calls for an efficient composition strategy for service provisioning where atomic services were insufficient in achieving customers requests. The proposed approach uses a two-phase composition approach (2P-IaaS) as a technique to allocate resources for IaaS requests, namely: (i) hosting resources mapping, and (ii) connectivity composition. This approach exploits conceptual modeling and employs semantic similarity [138] jointly with closeness centrality [139] in order to evaluate, assign and compose the topologies of accepted IaaS requests efficiently.

4.2 Resource Modeling State-of-the-Art

As the datacenter is the mainstream in hosting services and resources for CSPs, there is an urgent need to manipulate a datacenter efficiently. Despite the appearance of virtualization techniques as a solution for poor datacenter utilization, current server virtualization technologies that are used on datacenters are alone insufficient in addressing the problem of resource limitations. Therefore, applying virtualization techniques on datacenter network resources improves the resource utilization level and provides solutions for most of the datacenters' management challenges. In order to maximize the benefits with virtualized datacenters, representing datacenter resources conceptually can be a solution that allows for improved datacenter utilization and proper management. Conceptual modeling is a promising solution for datacenters' diverse and heterogeneous resource manipulations. In addition, the conceptual representation allows for inference operations to proceed through the virtualized datacenter for efficient management.

4.2.1 Resource Modeling

Modeling the datacenter's individual resources is the first step in building an information model for the desired system. The designed model should describe both the physical infrastructure and its virtualization aspects. A great effort has been made to model the cloud resources, e.g., compute, storage, and network resources. Inspired by the VN problem, modeling the cloud computing resources involves VN modeling and cloud resource modeling [26]. The pioneering work in defining an information model for network resources was done by the developers of the Network Description Language (NDL) [140] in creating a model for describing optical transport networks. The defined NDL ontology model used RDF language. In [141] ORCA extended NDL to define a more powerful model using Ontology Web Language (OWL), naming it NDL-OWL.1.0

Later, an extensible network description schema called the Network Mark-up Language (NML) that is based on the NDL [140] scheme is proposed for possible standardization within the Open Grid Forum (OGF). The NML specification includes only a primitive definition of virtual links and nodes and does not provide finalized schemes and properties of VRs [26]. Obviously, the aforementioned efforts in resource modeling are network-oriented models, in the sense that NML and NDL modeled the network components only. Incorporating the concept of virtualization in the modeling schema has been carried out in the Infrastructure and Network Description Language (INDL) [142] that appeared in *GEYSERS EU-FP7* project and the *NOVI* federation platform. INDL aims to capture the concept of virtualization in computing infrastructures and to describe the storage and computing capabilities of the resources. It is worth mentioning that the INDL ontology is built upon the NML ontology.

GEYSERS adopted INDL and extended it in order to define its information model. Figure 4.1 represents *GEYSERS's* main information model with classes and service hierarchical components. This model incorporates a description for a resource in a computing

infrastructure. INDL appears as a promising solution that considers the cloud resources in terms of VMs. However, it still differentiates between the cloud resources and network resources in some concepts and properties. This differentiation stimulates this work to define a unified model for the VR, i.e., virtual machine, virtual disk, and virtual switch are uniformly defined as a VR. The adoption of resource modeling is motivated by the concept

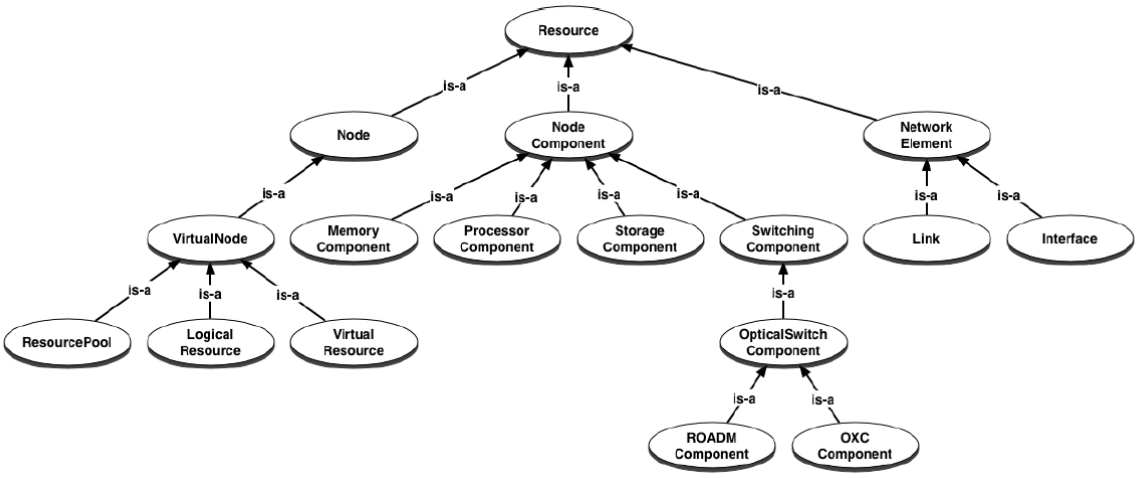


Figure 4.1: Main IMF model for INDL [11]

that different framework components need a common understanding of how resources are represented and how they interact with each other. In addition, the presence of diverse and heterogeneous resources in the proposed approach encourages the use of a simple, sharable, and extensible approach for representing knowledge in a cloud computing environment.

Consequently, the proposed framework is based on the existence of an ontology that represents syntactic and semantic information of resources, unifies the view of the available resources and unifies their treatments. This set of findings motivates our proposed ontology model. The popularity of using ontology in representing many real-world domains stems from ontology's capability in representing the syntactic and semantic information in a machine-readable format. In order to guarantee efficient resource modeling in the desired framework, the use of OWL [143] [144] has been adopted to model the proposed unified ontology for VR within the presented system. OWL describes modeled data by using sets

of classes and properties, and provides the flexibility and extensibility necessary within pervasive environments. Moreover, the proposed ontology is designed by extending the INDL capabilities and focusing on the connectivity between the resources that realize the composition methodology efficiently.

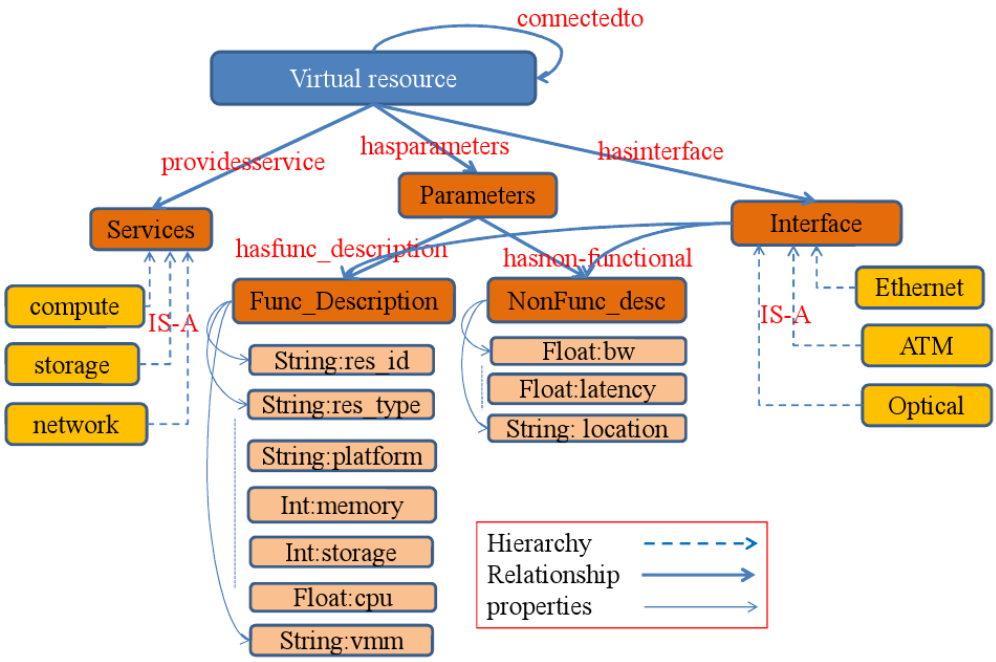


Figure 4.2: Proposed VR ontology

Figure 4.2 presents the proposed VR ontology model. This model represents three main concepts of the VR; *services*, *parameters*, and *interface* concepts. The *services* concept describes the type of services that can be performed by a VR, such as compute, storage, and network. The *parameters* concept describes the VR's functional and non-functional attributes. Finally, the *interface* concept presents the type of interface this resource has, e.g. Ethernet or ATM. One of the significant properties of the proposed model is the *Connectedto* property between VRs. This property represents the connectivity between

VRs in the pool. Phase-2 of the proposed approach mainly relies on this property in its operation. Applying virtualization techniques on the physical resources of the datacenter following the proposed VR model creates the VR repository pool. In addition, the physical network connectivity is represented in that repository by the VR *connectedto* property.

By defining a unified model for all heterogeneous and diverse datacenter resources, the first step in the proposed approach is achieved. The existence of the populated VRs in the virtual resource repository constitutes the search space where resources are utilized by the composition algorithm. This space is characterized by a large number of resources. Moreover, adopting ontology-based modeling in our approach allows the use of a reasoning capability associated with the defined model. However, the degree of expressivity [145] of the proposed ontology model impacts its operation efficiency. Therefore, we considered in our design the expressiveness characteristic in building the desired ontology model, in the sense that the number of classes, properties, and existential rules are limited so as to overcome high computational complexity. Furthermore, we address in designing the unified VR model, the trade-off between the expressivity and its abstraction such that it achieves high performance for the proposed approach. Based on [145] in classifying different expressivity levels, the proposed model is characterized as a simple OWL-Lite with low load and query times as complexity metrics. This classification ensures the efficiency of the developed ontology regarding memory consumption and processing time overheads. Following the expressivity level specification, selecting an appropriate reasoning engine manipulates the ontology model efficiently and impacts its performance. The reasoning engine manipulation is mainly a resource classification operation that forms a reduced search space. Thus, adopting an adequate reasoning engine ensures the proposed model consistency, performs resource classification, and prevents any repetition or conflict in the relationships among the model components [145].

In that context, a tableau-based algorithm reasoner is employed, e.g., Pellet, with the support of DL-based query languages, e.g., SPARQL as a semantic web query language.

SPARQL (and its extensions) is the most widely-used query language for the Semantic Web and the only language that is a W3C Recommendation [146]. Furthermore, it is fully supported in several implementations. Some research work has been conducted on the semantic approach as a cloud resource allocation technique and using ontology-based modeling for the cloud resources, e.g., [147], [148]. For a large pool of VRs, using a reasoning engine performs QoS class classification properly and reduces the search space for resource discovery. In the proposed approach, a reasoning engine is put in operation during the execution of the proposed approach. Moreover, reasoning engine functionalities are achieved by executing SPARQL query language that queries the ontology model efficiently. Adoption of the reasoning engine in the background while the composition is working achieved automatic classification and update for the VRP.It is worth mentioning that SPARQL query language provides more efficient reasoning mechanisms than other existing semantic query languages. Exploiting the reasoning feature in SPARQL has many advantages: (1) it handles the scalability issues for reducing the search space, (2) it handles the problems incurred due to incomplete requests from the customers, and (3) it simplifies the resource discovery complexity [149].

4.2.2 Composition Strategy

Some similar research work has adopted web service composition methodologies to compose SaaS and PaaS services, where a new web service is composed of a set of existing web services to accomplish the requested task. These research efforts follow the abstract service composition concept of combining existing services into a new service in order to fulfill the request. However, the main challenge is how to choose which services will satisfy the user requirements, especially if there are many candidate services with the same functionality but different QoS attributes. There are many methods and models employed to solve this problem [150], most of them treating web service composition as an AI planning problem and using biologically inspired algorithms such as Ant Colony Optimization (ACO), the

Genetic Algorithm (GA) [151], and the Evolutionary Algorithm (EA). In addition, there exist research proposals that address the composition strategy utilizing the multi-agent framework and agent corporations, e.g., [152], [153], and [138]. All these possible solutions help in building an autonomous, scalable, adaptable, and reliable service provisioning solution [154].

For IaaS service, many attempts have been performed to realize composite network cloud infrastructure services [155] [156], particularly when the network is involved in most of the recent applications and cloud services. Since network QoS has a significant impact on the provisioned cloud service performance, network and cloud services are best modeled and analyzed as a composite service. This recommendation has been embraced in the early stage by defining a unified model for VRs. The following key concepts form the foundation of our approach and offer a promising basis for network and cloud infrastructure composition.

4.3 Key Foundation Concepts

Four key foundation concepts that build up the proposed ontology-based resource allocation approach are introduced. A VRP representation is created when the datacenter's diverse resources, e.g. compute, storage, and network are represented following the predefined unified VR model then collected in that pool. The approach presented here defines a combined controller named the *composer* that exploits the two main concepts, semantic similarity and closeness centrality, in order to accomplish its task. Furthermore, the random walk technique is elaborated as a driver technique for the virtual infrastructure composition.

4.3.1 Virtual Resource Pool (VRP)

The datacenter as a massive resource repository has become an efficient and promising infrastructure for supporting the growth of requested resources and cloud services. A virtualized datacenter is created by applying virtualization techniques to physical resources, including server, storage, and network. In addition, the virtualized datacenter provides a solution for better flexible resource management. Virtualization techniques with a unified VR model have been used as a solution that serves different demand patterns with diverse and heterogeneous resources. As a result, a virtualized unified resource repository, the VRP, is created. This pool generalizes the representation of all the available resources in the datacenter by adopting the unified VR model presented in Figure 4.2.

Furthermore, the creation of the pool with different QoS classes leads to variant granularities of VRs in the pool. The VRP is reshaped periodically after each successful resource allocation iteration. In addition, the granularity of different VRs, with periodic updating, copes with the fluctuations in IaaS demand patterns and implies the dynamic characteristics of the proposed resource allocation approach. Reliance on a preexisting VRP supports the presented approach in many ways. It (1) contributes to the approach being different from existing resource allocation techniques that work directly on the physical layer, (2) ensures uniform treatment of the resources and the convergence management of networking and computing resources as well, (3) reduces the dependency of the presented approach components on resource types, and (4) bridges gaps that may occur due to the diversity and heterogeneity of resources.

4.3.2 Semantic Similarity Evaluation

Semantic approaches have been proposed in the past as a means for enhancing web service search mechanisms [157] [158] [159] [160]. This was later utilized to cluster peers with similar content to become part of the same Semantic Overlay Network (SON). Furthermore,

such clustering reduced communication costs and increased searching query accuracy levels. However successful composition processes are highly dependent on the mechanism for filtering and selecting of services. As such, the dynamic composition of services requires an understanding of the capabilities and compatibilities of services. Full automation of composition is still an object of ongoing research, while partial automation of composition has had more success in the past. With semantic web technology full automation has become more feasible. Through semantics, information is given a well-defined meaning thus better enabling collaboration between computers and people.

Extensive research has been conducted in regard to semantic manipulations in web services [161] [162] [163]. Autonomic composition of web services involves the need to support some degree of semantic evaluation of such services. Ontologies play an important role in the semantic-driven service composition. In the context of semantic cloud computing, ontology has been defined as representing cloud concepts, services, and their semantic relations. In order to enhance the opportunity for discovering appropriate cloud services, a similarity evaluation method was devised. This similarity evaluation method uses a semantic similarity metric that plays a vital role in evaluating the matching between two concepts. Similarity in its general form is divided into five categories [164] [165]: Exact, Subset, Subsume, Intersect, and Disjoint. The presented resource allocation approach exploits the efficiency that is provided by this evaluation method to assess the matching between cloud resources. Semantic similarity ensures that a matching result occurs and leads to an increase in resource allocation accuracy. Similarity evaluation is designed to increase the chance of finding relevant alternatives of a resource. The degree of similarity is normalized, ranging from 1 for an exact match to 0 for disjoint [138].

Three kinds of reasoning methods are devised to achieve semantic similarity measurement [138] over a cloud ontology: similarity reasoning, compatibility reasoning, and numerical reasoning.

- Similarity reasoning: determines the similarity between two concepts in the ontology by counting common reachable nodes; it represents the degree of commonality between concepts, i.e., how much concept x and concept y share in common. For example the similarity between two concepts x and y can be determined by counting their common reachable nodes as follows:

$$Sim(x,y) = \rho \frac{|\alpha(x) \cap \alpha(y)|}{|\alpha(x)|} + (1-\rho)\frac{|\alpha(x) \cap \alpha(y)|}{|\alpha(y)|} \tag{4.1}$$

  Where $\rho \in [0,1]$

- Compatibility reasoning: determines the similarity between two sibling concepts in the cloud ontology based on their label values, e.g., determining the compatibility between two different versions of a software. These sibling nodes will have a high degree of similarity, but differ only in terms of their chronological ordering. For example the similarity between two concepts x and y can be determined as follows:

$$\hat{Sim}(x,y) = Sim(x,y) + \frac{0.8^{|c_x - c_y|}}{10} \tag{4.2}$$

  Compatibility reasoning formula consists of two parts: (1) measuring the degree of similarity between x and y, and (2) differentiating between x and y in terms of their chronological ordering. Where c_x and c_y are the label values of x and y respectively, that represent the chronological orderings of different versions of a software.

- Numerical reasoning: determines the similarity between two numeric concepts based on their label values. The similarity between two numeric values in same domain can be calculated as following formula:

$$Sim(a,b,c) = 1 - \left| \frac{a-b}{Max_c - Min_c} \right| \tag{4.3}$$

  where a and b are numeric values and c is the concept. Max_c and Min_c are the

upper and lower limits of concept c values, respectively.

It is worth mentioning that, the proposed unified ontology represents the VR specifications (e.g., CPU, mem, and storage) which are numeric values. Thus, similarity and compatibility reasoning are most fit the evaluation of cloud SaaS and PaaS ontologies, since they have many concepts that are built in a hierarchical level and have a chronological order. However, in our research concerning IaaS service, Numerical reasoning is the most appropriate method, since all the data properties representing the unified VR model are numerical values. Moreover, the proposed unified VR model contains few concepts without chronological order. Furthermore, the running reasoning engine exploits the semantic represented by *Connectedto* property in manipulating the proposed approach phases. A comprehensive example that illustrates the reasoning methods mentioned above is presented in the supplemental materials in [138].

As an example that illustrates the numerical similarity measurement, it is assumed that the customer submits a request for a resource with the following attributes: $CPU = 1.2GHz$, $memory = 512MB$, $storage = 750GB$. Meanwhile, different VR granularities exist in the pool and are specified by the CSP. For example, assume that there exists a VR in the pool with the following attributes: $CPU = 2GHz$, $memory = 1GB$, $storage = 750GB$. The upper/lower limits of the attribute values in any published VR are constrained by the actual physical server capacity in the datacenter. These limits are defined as Max_c and Min_c in the formula (e.g. $Max_c = 4$ and $Min_c = 1$ for CPU). The similarity evaluation is carried out between the requested resource and a sample VR in the pool (e.g. VM as a compute resource) and is calculated as follows:

$$Sim(2, 1.2, CPU) = 1 - \left| \frac{2 - 1.2}{4 - 1} \right| = 0.733 \tag{4.4}$$

This equation illustrates that the similarity between the requested resource and this VM is 0.733 for the CPU attribute. As well, with regard to memory, storage, and network

attributes, the same formula is applied. Semantic similarity measurement evaluates the matching between resources and is designed to increase the opportunity of finding a solution. It has been observed that semantic similarity always ensures a matching degree that varies from VR to another.

4.3.3 Closeness Centrality Evaluation

In graph theory and network analysis scope, several measures exist to calculate the centrality of a node within a graph (e.g., degree, betweenness, closeness, and eigenvector centrality) [139]. Closeness centrality is one of the centrality measures that is widely used in complex network analysis; it uses the distance between nodes to measure the centrality of a node and to evaluate the importance of a node in the network graph. Closeness centrality of a node is evaluated by measuring the distance between the desired node and all other nodes in the network and this measurement reflects the node importance within the network. If the node is in the center of the network it has a high value, while small value nodes are edge nodes [139].

The well-known closeness centrality definition states, "The closeness of a node n_i is the reciprocal of the sum of geodesic distances to all other nodes reachable from it" [139], and it is mathematically represented as follows:

$$C_c(n_i) = \frac{1}{\sum_{j=1}^{N} d(i,j)} \tag{4.5}$$

This formula scores the closeness centrality of a node n_i with respect to all nodes in a network of size N, where $d(i,j)$ is the geodesic distance between node n_i and node n_j in the desired network. It is worth mentioning that, in our approach, the defined distance in the closeness centrality metric is measured by counting the number of hops between each two nodes.

For efficient resource allocation, the presented approach utilizes the guidance of cen-

trality analysis and adapts the closeness centrality measurement, since closeness centrality measures the distances between VRs in the pool and ensures the proximity of the hosting resources for the same request. The proposed approach adapts the closeness centrality in both phases; Phase-1 applies the locality-aware principle in the form of measuring the closeness of one candidate VR to other candidate VRs in the same request as in Eq. (4.5). As well, Phase-2 utilizes the guidance of closeness centrality in terms of the shortest path between the mapped resources in the solution space for an efficient connectivity path composition. Thus, using closeness centrality ensures the proximity of the discovered resources, confirms the correctness of the discovery direction, utilizes the resources efficiently (conserving bandwidth), and prevents resource consumption during the composition [139].

In this research, we argue that the integration of semantic similarity and closeness centrality enriches the presented approach such that it maximizes the throughput of the IaaS provider and optimizes the datacenter resource utilization. Also, the adoption of these concepts achieves good results in the performance evaluation and confirms the presented approach as a promising solution for datacenter poor-resource utilization.

4.3.4 Biased Random Walk Technique

Many techniques have been implemented to discover a path in a physical network. The most popular technique is the flooding technique. However, flooding the network with advertisements searching for a particular resource consumes the network's resources and is ill-suited for VR discovery. Instead, a Random Walk (RW) technique [166] is adopted. This technique utilizes fewer resources than flooding during the network discovery process. Also, this technique fits the received incomplete information about the connecting topology of the requested VI. RW explores the intermediate VRs step by step while the composition proceeds.

In order to compose the connectivity paths efficiently, an adapted Biased Random Walk

(BRW) is used. By using a BRW, a significant reduction in discovery time and excessive resource consumption over conventional RW is observed. In BRW, a bias metric (bm) is defined as the sum of the aforementioned evaluation measurements, i.e., semantic similarity and closeness centrality. This metric controls the discovery process, in the sense that it biases the next hop VR discovery and guides the connectivity composition operation.

4.4 2-Phase Composition as a Resource Allocation Technique

In this section, a Two-Phase IaaS resource allocation approach (2P-IaaS) is proposed: (i) the hosting resources mapping phase, and (ii) the connectivity composition phase. The proposed two-phase IaaS resource allocation approach is characterized by:

- Modeling datacenter individual resources using a unified VR ontology model followed by creating the virtual resource pool.
- Uniform treatment of all the available VRs in the pool.
- Using an efficient 2P-IaaS resource allocation algorithm that performs reasoning on the ontology model and implements semantic similarity with closeness centrality as evaluation measurement techniques.
- Exploiting a network discovery technique that explores IaaS request network topology guided by the semantic representation of the unified ontology.

The proposed approach defines two business entities: (1) the IaaS service provider who owns the datacenter, and is responsible for establishing the VRP, and (2) the cloud customer who explores the provider service catalog and submits an IaaS request. The centralized entity, the *composer*, is responsible for receiving the customer IaaS request for infrastructure and performing the composition. A received IaaS request is modeled in an abstract way as shown in Figure 4.3. In this figure, a set of VRs is connected to a

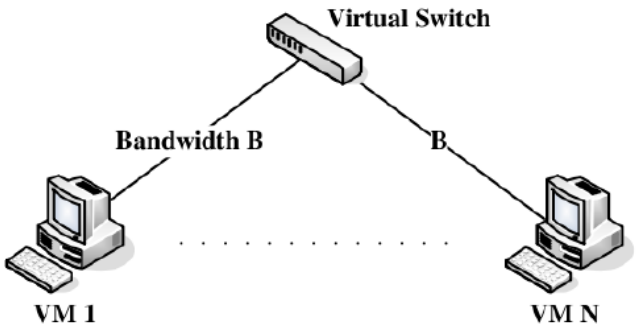


Figure 4.3: IaaS request Abstract Output [6]

virtual switch using a bidirectional link with QoS constraints. This abstraction is modeled as $IaaS_{Req} = (\text{NR}es, \text{C}on)$, where NR$es$ denotes a set of VRs and Con represents the set of connectivity QoS constraints among these resources [6]. The proposed composition algorithm 2P-IaaS phases execute cooperatively. In Phase-1; the hosting resource mapping phase starts to discover and map the requested resources from the virtual pool. In Phase-2; the connectivity composition phase starts immediately after the completion of the first phase. In addition, the hosting resources mapping phase interferes with the composition phase in discovering and mapping the intermediate networking resources.

4.4.1 Phase-1: The Hosting Resource Mapping Phase

In Phase-1, the *composer* is concerned with VR discovery from the VRP for the submitted requests. Figure 4.4 provides a simplified view of Phase-1 execution. The operational steps start by the customer submitting his request. After the request submission (step 1), $Res_i, i \in (1..N)$, the *composer* extracts the requested resource's parameters ($Res_{(i)fun}$). In order to discover the requested resource in the VR pool, the *composer* starts to query the VR pool using a formulated SPARQL with the extracted parameters (step 2). As a result of SPARQL execution, a reduced search space is created. This search space contains the candidate resources for the requested resource (step 3). To increase the candidate resource selection accuracy, the *composer* performs semantic similarity and closeness centrality eval-

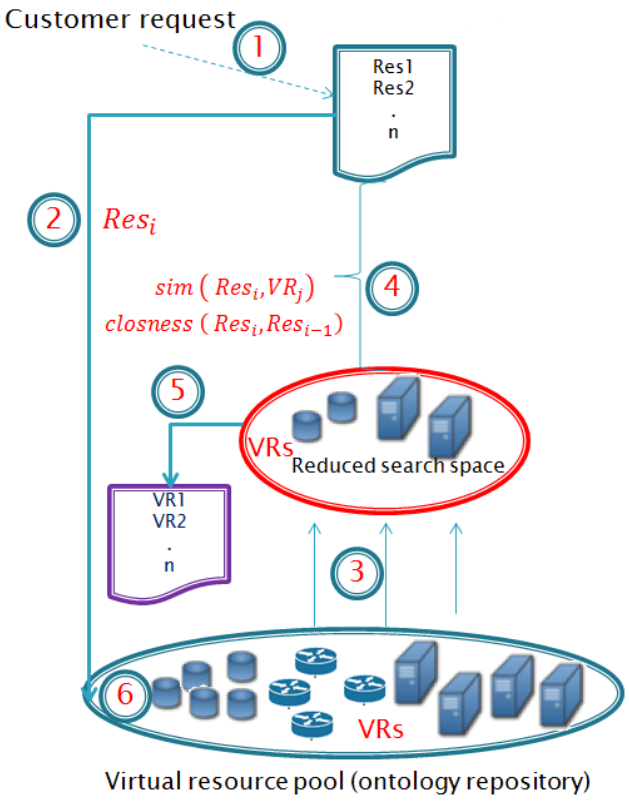


Figure 4.4: Phase-1: VR discovery steps

uations on the reduced search space. The *composer* evaluates the similarity between the requested resource (Res_i) and the reduced (VRs) by applying $sim(Res_{(i)func}, VR_{(j)func})$, $sim(Res_{(i)non-f}, VR_{(j)non-f})$ evaluations (step 4). A ranked score for each candidate VR is calculated by aggregating its similarity scores as follows:

$$RankScore_{Res_i - VR_j} = sim(Res_{(i)func}, VR_{(j)func}) + sim(Res_{(i)non-f}, VR_{(j)non-f}), i \in N, j \in M. \tag{4.6}$$

Where N is the number of requested resources in one request and M is the number of reduced VRs candidate for selection for each requested resource. The highest ranked score represents the best-fit (VR) for the requested (Res). Then, the successfully discovered resources are saved in (VR*list*), and the discovery phase for the next requested VR commences. This process continues until all the requested resources in NRes are discovered. After completing the discovery operation, the output of Phase-1, the successfully discov-

ered resources that are stored in the VR*list* are fed to phase-2 to finalize the composition (step 5). The reasoning engine operations appeared in different steps during the execution of Phase-1. Firstly, the SPARQL query is formulated based on the submitted request from the customer then executed by the reasoning engine. Thus, the reasoning engine, in that case, accepts any submitted forms, i.e., it accepts customized requests with missing information and manipulates the ontology efficiently to obtain accurate solutions. Secondly, the reduced resource pool is created as a result of executing the SPARQL query, so classification and reduction in the search space are achieved. Finally, the *composer* executes the reasoning engine to update the pool and check the model consistency after a successful composition. Algorithm 3 provides the detailed steps for the 2P-IaaS operation. As soon as Phase-1 has completed its work, Phase-2 starts.

4.4.2 Phase-2: The Connectivity Composition Phase

Phase-2 is responsible for composing the connectivity among the discovered resources that are stored in VR*list*, while respecting the IaaS request QoS constraints C*on*. This phase involves VR discovery for intermediate resources as well. Figure 4.5 explains the Phase-2 execution steps in sequence. Initially, VR composition receives the discovered VRs list that represents the infrastructure virtual resources (IVR). The connectivity composition phase operates recursively for each pair of discovered VRs in the VRL*ist*. The *composer* defines IVR_s as a source node and IVR_d as a destination node (steps 1 and 2). Starting from IVR_s moving towards IVR_d, the *composer* consults the VR pool ontology database using SPARQL and the *connectedto* property of the source IVR_s (step 3A). As a result, a reduced search space is created containing a set of networked resources (VRs) that are directly connected to the source node IVR_s and can join the connectivity path (step 3B). The *composer* starts to exploit the semantic similarity measurements to evaluate the degree of matching between the reduced search space and the requested QoS specifications.

Algorithm 1 2P-IaaS Algorithm Pseudo code

Input: IaaSRequest $IaaS_{Req} = (\text{NR}es, \text{C}on)$
Output: Virtual topology (connecting NRes and satisfying Con
Steps:
Phase 1: Hosting resources mapping
Input: IaaSRequest $IaaS_{Req} = (\text{NR}es, \text{C}on)$
Output: VRList (successfully discovered NRes)
for each Res_i in NRes **do**
  Extract ResP$aram_i$
  VR= Discover(Res_i,ResP$aram_i$,Res_{i-1},ResP$aram_{i-1}$)
  if (VR is NULL) **then**
    R$ejected++$
    Break
  else
    VRL$ist \leftarrow$ VR
  end if
end for
Phase 2: Connectivity Composition
Input: VRList
Output: Topology connecting VRList
if $size$(VRList) > 0 **then**
  for each pair (VR$_i$,VR$_{i+1}$) in VRList **do**
    $temp\_vr \leftarrow$ VR$_i$
    while ($temp\_vr \neq$ VR$_{i+1}$) **do**
      VR$_i \leftarrow temp_vr$
      $temp\_vr$=Discover(VR$_i$,$Connectedto$,VR$_{i+1}$, P$aram_{i+1}$)
      if ($temp\_vr$ is NULL) **then**
        R$ejected++$
        Break
      else
        C$onnectivity$P$ath \leftarrow temp_vr$
      end if
    end while
    T$opology \leftarrow$ C$onnectivity$Path
  end for
end if

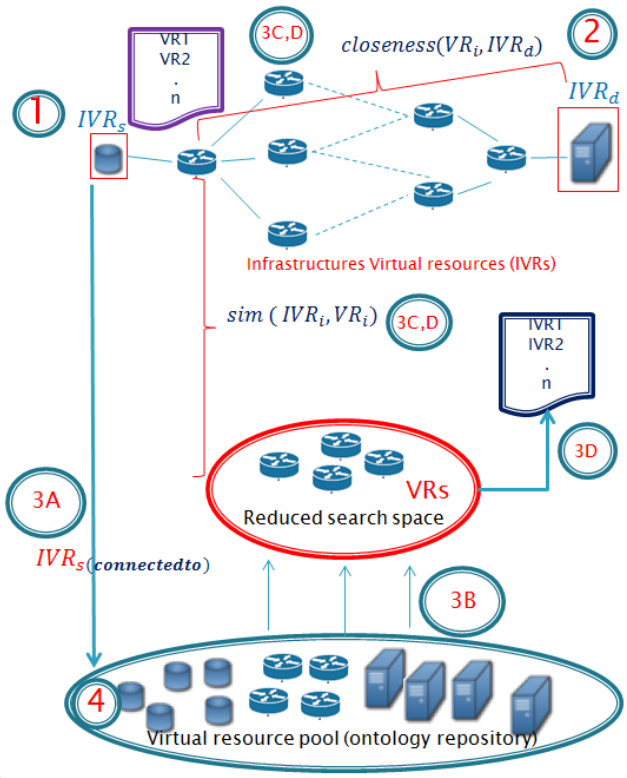


Figure 4.5: Phase-2: Connectivity composition steps

A set of similarity measurements that guide the composition process are carried out as follows: evaluate the semantic similarity between the discovered VR (IVR_s) and the reduced search space VRs parameters, e.g., $sim(IVR_{fun}, VR_{fun})$, and evaluate the closeness of the reduced search space VRs towards the defined destination node (IVR_d), e.g., $closeness(VR, IVR_d)$ (steps 3C and D). The *composer* utilizes the BRW technique recursively to discover intermediate nodes. In each discovery iteration, a *RankScore* is calculated as the sum of the similarity evaluation and the closeness evaluation score. And thus, the highest-score VR is chosen and joins the connectivity path. The process continue by feeding the highest rank score VR to the next iteration in the connectivity composition. This cycle is repeated for each intermediate VR discovery until (IVR_d) is reached. Finally, the *composer* executes the reasoning engine to update the pool declaring a successful composition has been achieved (step 4). The successfully composed connectivity path is stored in the topology which is the output of the 2P-IaaS algorithm.

Algorithm 2 Discover Function Pseudo code

1: **Discover** (Res_i, $Param_i$,Res_j, $Param_j$)
2: $ResultSet \leftarrow Consult\_Ontology DB(Res_i, Param_i)$
3: **while** ($ResultSet$ is NULL) and ($Ntrials \neq 0$) **do**
4: Relax $Param_i$
5: $Ntrials - -$
6: **end while**
7: **if** $ResultSet$ is NULL **then**
8: $Ntrials \leftarrow T$
9: **Return** NULL
10: **else**
11: **for each** VR in $ResultSet$ **do**
12: $Eval_1 \leftarrow Sim(Res_{func}, VR_{func})$
13: $Eval_2 \leftarrow Sim(Res_{non-func}, VR_{non-func})$
14: $Eval_3 \leftarrow Eval\_Closeness(VR, Res_j)$
15: $RankScore_{Res-VR} \leftarrow Eval_1 + Eval_2 + Eval_3$
16: $RankScores_{VR} \leftarrow RankScore_{Res-VR}$
17: **end for**
18: $R\_ResultSet \leftarrow Rank(RankScores)$
19: $VR \leftarrow Highest(R\_ResultSet)$
20: **Return** VR
21: **end if**

Algorithm 2 elaborates the *Discover* function. This function is used in both phases with different parameters and objectives. In Phase-1, the *Discover* function performs the semantic similarity and closeness centrality evaluations between the requested resource and the VR pool utilizing the requested resource parameters. In Phase-2, the *Discover* function is carried out between a successfully discovered resource and all the directly connected resources utilizing the *Connectedto* property while respecting the *QoS* requirement *Con*. In both phases, the *Discover* function consults the ontology model using SPARQL query searching for VRs in the pool. IF the returned result set from the resource pool consultation is empty, it means that there are no available resources, the algorithm starts to relax the query to get a result set for evaluation. This relaxation process repeated *Ntrials* for each query with *NULL* result set. Otherwise, the request is subsequently counted as a rejected request. As such, the accumulated evaluation result represented by *RankScore* is interpreted differently in the two phases. In Phase-1, *RankScore* reflects the degree of matching

and the proximity of the discovered VRs to each other, while in Phase-2, *RankScore* acts as the (bm) that guides the BRW technique. Furthermore, the relaxation achieved in Phase-2 while discovering the inter-networking resources act as a backtracking steps in performing the connectivity composition efficiently. Finally, the *RankScore* is ranked and the highest match value is returned as the *Discover* result. In Phase-2, the reasoning engine role appeared in both the composition algorithm and the discovery function as well through consulting the VRP looking for networking resources; this consultation resulted in a reduced search space, so, classification and reduced resource pool is achieved by the reasoning operation. Moreover, discovering the connectivity using the *Connectedto* property is achieved by exploiting reasoning capabilities in performing inference on the defined ontology. The merit of using ontology reasoning is exploiting the SPARQL customized formulation in consulting the ontology and inference new connectivity information. Also, the periodic update for the resource pool after the successful composition is achieved automatically by the reasoning engine. The *composer* indicates the success of the IaaS resource allocation process when it successfully passes the two phases; otherwise, the IaaS request is marked as rejected.

4.4.3 Illustrative Composition Example

Before we start elaborating the proposed algorithm performance evaluation, we present an illustrative example. This example explains the fundamental steps in the algorithm. Assume that we have a VI request as shown in figure 4.6. This request incorporates end-points represented as compute (VM) and storage (VDisk) with a network connecting topology among these endpoints. Furthermore, for each end-point, the customer specifies QoS requirements in terms of CPU, Storage, and memory for end-points. Also, QoS for network connectivity has been specified in terms of the required bandwidth and end-to-end delay. As shown in the figure, no topology specifications are available, and this missing information constitutes the novelty and power which exist in our proposed approach. It is

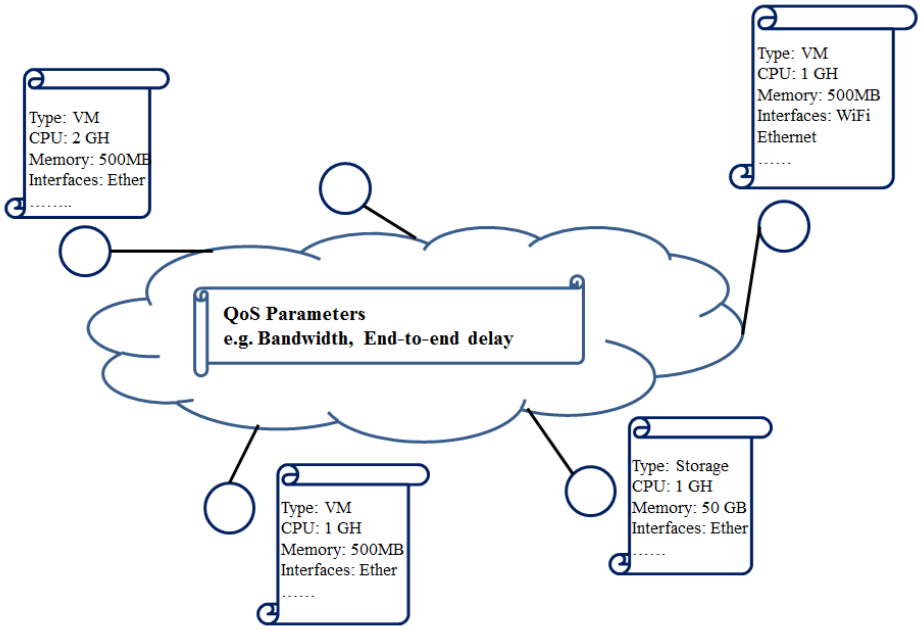


Figure 4.6: Example on IaaS request adapted from [12]

required to allocate the appropriate resources for end-points and to compose the connecting topology among these end-points while respecting the desired QoS. Moreover, figure 4.7 represents the desired output after executing the composition algorithm. In more detail, this figure shows the generated traffic pattern between the allocated end-points. In similar composition algorithms for VN embedding [132], the system allows the customers to specify the topology of the connections e.g. bus, star, or mesh topology. However, the composition algorithm's superiority was shown by its discovering the connectivity topology step-by-step without prior knowledge of the underlying physical network connectivity and without knowing the requested topology. This characteristic confirms the applicability of our approach to many network topologies.

In our example, the composition algorithm starts by discovering the end-points in Phase-1. Then, in Phase-2, the algorithm proceeds with composing the connectivity between each pair of the discovered end-points avoiding any repetitions in link allocation. The algorithm ends up with the traffic pattern shown in figure 4.7. It is worth mentioning

that the output topology follows the underlying datacenter physical topology, e.g., fat-tree topology, although there is missing information regarding the physical network topology and the requested topology, the composition algorithm performed the connectivity composition operations accurately and blindly, only adopting the *Connectedto* property associated with each VR in the pool.

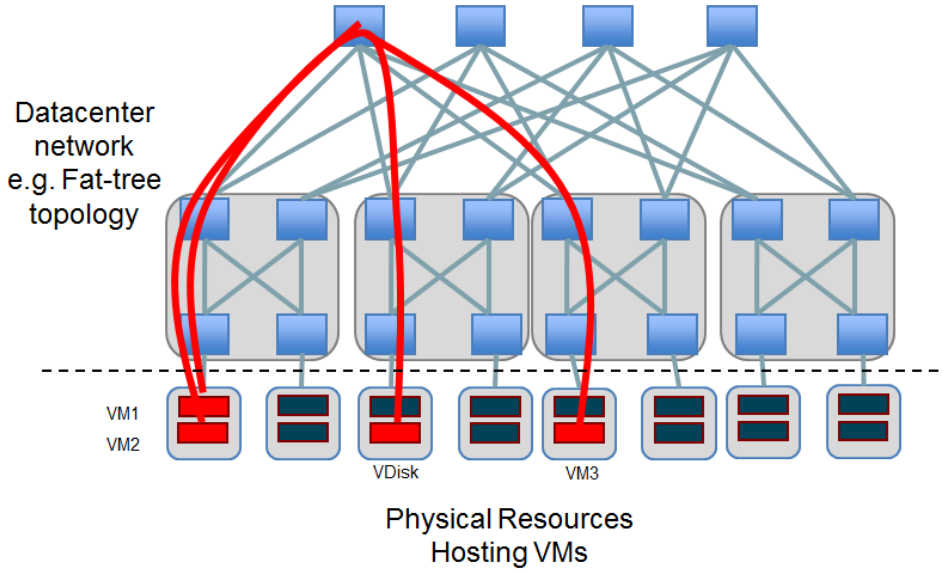


Figure 4.7: Desired output from the composition

4.4.4 Composition Algorithm Complexity

Regarding the complexity analysis of the proposed approach, consider the situation where the management system received M requests. For each request, consider n requested resources. Phase-1 requires $\mathcal{O}(n).\mathcal{O}(Discovery)$ to discover n resources, and Phase-2 requires $\mathcal{O}(m.n-1).\mathcal{O}(Discovery)$ to compose the connectivity paths for a request where m is the maximum number of intermediate VRs in a path. Furthermore, assume that we have a VRP with v resources, and we conducted t trials for discovering each requested resource n. The main function in both phases, the *Discover* function, requires at the worst case $\mathcal{O}(t.v)$ to consult the ontology repository for candidates resources of size c and $\mathcal{O}(c^2)$ to

sort and rank the selected candidate resources. Also, the ontology load and query time are taken into consideration while dimensioning the search space. Thus, the *Discover* function complexity is represented by $\mathcal{O}(t.v.c^2)$.

Obviously, the complexity of the *Discover* function grows with the increasing size of the VRP and number of trials. Therefore, a scalability issue threatens the proposed solution applicability. To sum up, the 2P-IaaS composition algorithm mainly relies on Phase-1; this phase utilizes reasoning in performing classification and reducing the search space. Also, Phase-1 is responsible for discovering and assigning resources such that it utilizes the resources efficiently. The integration of semantic similarity with closeness centrality realizes the locality-aware principle that alleviates the occurrence of blocking events [6] [5].

4.5 Performance Evaluation and Numerical Results

As proof of concept, to assess the efficiency of the 2P-IaaS approach presented, extensive simulations were carried out. In the following section, the setting of the simulation environment will be presented, along with a description of the experiments, the performance evaluation metrics used, and then the results obtained.

4.5.1 Experiment Environment

Consider a small-scale datacenter with 320 physical machines, 16 top-of-rack switches, 8 aggregate switches, and 4 core switches. Physical resources are connected using *Clos* topology which provides extensive path diversity [4]. Each physical machine has 4 CPU cores, 8 GB of memory, 1 TB of storage and contains a 1 *Gbps* network adapter. The inter-switch bandwidth on *Clos* topology is 1 *Gbps*.

The presented ontology model was developed in OWL-2 language as a standard ontology language proposed by W3C and using protege [167]. The Jena API was adopted as a

framework that allows both SPARQL query execution and reasoning operations. IaaS requests were generated randomly following Poisson distribution by varying the arrival rate from four IaaS requests per 100 time units to eight IaaS requests per 100 time units. The lifetime duration of the IaaS requests followed an exponential distribution with an average of a 5000 time-unit window. QoS requirements of the submitted IaaS request were randomly specified following a uniform distribution among three defined QoS classes: gratis, middle, and production [131]. The experimental assessment was carried out through the CloudSim framework [168] as a discrete event simulator.

The on-line simulation methodology was followed in serving the incoming IaaS requests in the conducted simulation, such that the incoming requests were arranged in a FIFO queue according to their arrival time. Moreover, the performance evaluation of the proposed ontology-based resource allocation approach was carried out against SecondNet [5] and Oktopus [6] as benchmarks.

4.5.2 Performance Evaluation Metrics

In order to compare 2P-IaaS to SecondNet [5] and Oktopus [6] benchmarks, the following performance metrics were measured:

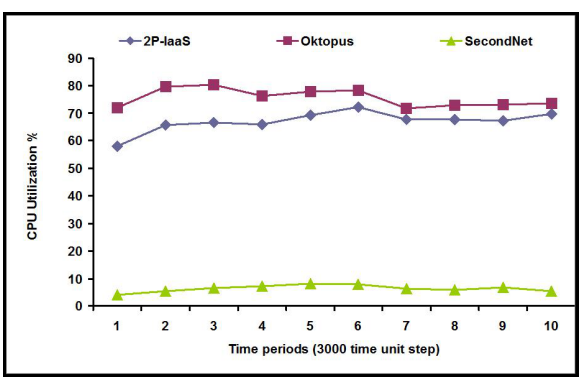
- Acceptance ratio: measured as the ratio of accepted IaaS requests to their overall submitted requests in a given period.
- Datacenter resource utilization: measured as the ratio between the resources used(CPU, memory, storage, and bandwidth) and their overall capacity in the datacenter at a certain point in time.

Table 4.1: 2P-IaaS Composition: IaaS Request Acceptance

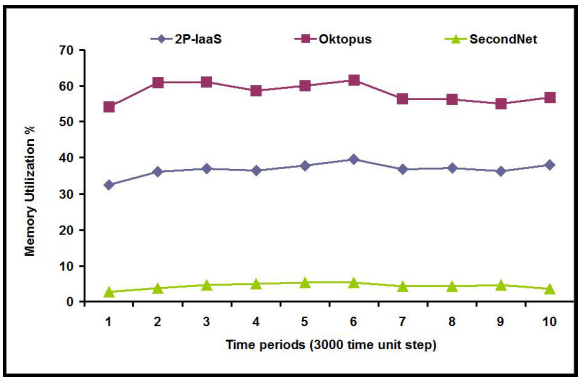
		Number of IaaS Requests			
IaaS Serving Approach		500	600	700	800
2P-IaaS	# Accepted IaaS	437	521	577	601
	Acceptance Ratio	87%	87%	82%	75%
Oktopus	# Accepted IaaS	437	495	493	536
	Acceptance Ratio	87%	83%	70%	67%
SecondNet	# Accepted IaaS	49	57	74	80
	Acceptance Ratio	10%	10%	11%	10%

4.5.3 Evaluation Results

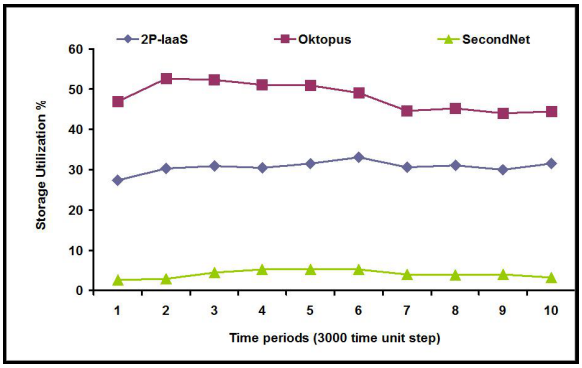
Through extensive experiments, the IaaS request acceptance ratio was evaluated among the aforementioned benchmarks. The proposed algorithm was evaluated with an increasing number of IaaS requests, from 500 to 800 requests. Table 4.1 shows the acceptance ratios for a different set of mixed QoS class IaaS requests. As expected, the proposed 2P-IaaS outperformed benchmarks Oktopus and SecondNet. It is clear that at the lowest number of received requests, i.e., 500, the acceptance ratio for the three algorithms are close to each other. However, with increasing numbers of received requests, the proposed 2P-IaaS is superior to the others. It is worth mentioning that the achieved relaxation in both phases contributes significantly in our approach acceptance ratio among the benchmarks, however at the expense of the algorithm execution time. The SecondNet VDC resource allocation approach is mainly based on a bipartite graph algorithm with min-cost flow to map VMs onto physical servers. Further, SecondNet adopted a Breadth-First Search (BFS) as the shortest-path algorithm for links. The resulting clique topology in SecondNet possesses dense connectivity that makes it difficult for the provider to multiplex multiple IaaS requests on the underlying network infrastructure and this result in a lower acceptance ratio. As a result, a less efficient cloud resource usage is observed as shown in the Figures 4.8a to 4.8d. The other benchmark, Oktopus, relied on a virtual cluster greedy-allocation approach illustrating an IaaS request acceptance ratio closer to 2P-IaaS. However, with



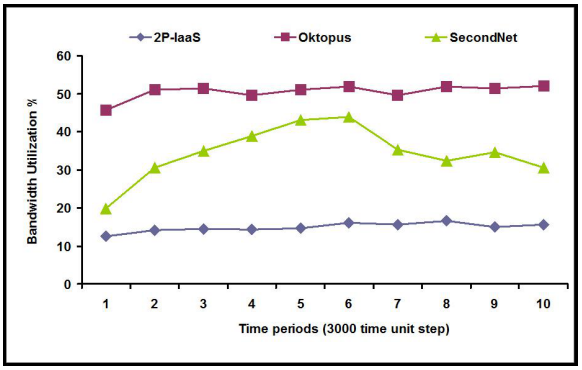
(a) CPU utilization



(b) Memory utilization



(c) Storage utilization



(d) Bandwidth utilization

Figure 4.8: 2P-IaaS Composition Resource Utilization

increasing IaaS requests, a greedy heuristic algorithm designed just for satisfying capacity requirements may not be efficient.

Moreover, figures (4.8a, 4.8b, 4.8c and 4.8d) confirm the efficient resource utilization of the proposed 2P-IaaS. The results show the superiority of the proposed composition algorithm 2P-IaaS in terms of CPU, memory, storage, and bandwidth utilization over the benchmarks. Obviously, approaches with low utilization values are efficient in their resources usage than others with high utilization level. Thus, although SecondNet achieved the best utilization levels in CPU, memory, and storage, it possesses a lower acceptance ratio, which impacts its overall efficiency. Moreover, the proposed 2P-IaaS achieved a better utilization (i.e., low utilization levels) than Oktopus in CPU, memory, and storage, and the best bandwidth utilization among both benchmarks. Indeed, using virtualization techniques and the ontology-based solution with reasoning capability resulted in minimal

datacenter resource utilization as well as less blocking of IaaS requests. The integration of semantic similarity, closeness centrality and BRW techniques in the proposed approach provides an efficient methodology for datacenter resource allocation. Moreover, the proposed composition algorithm 2P-IaaS can be applied to various tree-like datacenter topologies (e.g. fat-tree).

4.6 Composition Approach Limitations

Although ontology-based modeling has been applied in different research fields, it possesses limitations with regards to the growing scale in the search space. Particularly in the cloud computing environment, the ever-growing scale in the search space highlights this issue. In that context, the ontology model load time and response time are the most important performance metrics that evaluate the scalability of the ontology model and the growing search space, where:

- Load Time is defined as the time consumed to load the ontology and check its consistency including the time to do any preparation before querying, i.e., populating the resource pool.
- Response Time is defined as the time measured starting with executing the query and ending when all the query results were stored into a local variable, without iterating the results.

Regarding the ontology load time, the designed ontology-model expressivity level controlled this metric and performed well in a reasonable time. However, ever-growing search space of managed resources leads to a significantly growing response time. Furthermore, performing re-querying trials in the search space looking for candidate resources contributes to response time as well. This increase in response time increases the time complexity for the running algorithms and impacts the efficiency of the overall approach in large search spaces.

Table 4.2: 2P-IaaS Composition: Execution Time Analysis

	Measured Time (sec)			
# IaaS Requests	Load Time	Discovery Time	Composition Time	Total Sim.
500 IaaS	2.124	97.080	0.717	17460.7
600 IaaS	2.141	119.954	0.917	21964.4
700 IaaS	2.350	191.535	1.167	26541.5
800 IaaS	2.102	335.789	1.257	31820.7

In 4.2, the table shows the performance of the proposed 2P-IaaS composition algorithm with increasing number of IaaS requests regards the ontology load time, the average time for discovering time (Phase-1 and Phase-2), the average time for composition time (Phase-2) and the overall measured simulation time. It is clear that loading the ontology consumes less time in loading, that ensures the efficient ontology design considering the ontology expressivity. The low values for the average discovery and composition time ensures the efficiency of the adopted reasoning engine. However, the time consuming in the total simulation time reflects the multiple iterations achieved and the backtrack with relaxation conditions that we proposed in our algorithms. Moreover, the growing search space and the increasing number of IaaS requests contribute significantly on the total simulation time. This observation limits the applicability of the proposed approach to small-scale datacenters as well as small numbers of requests. Also, the results shown in table 4.1 confirmed this conclusion, where a noticeable decrease in the acceptance ratio of the proposed approach occurs with the increasing numbers of IaaS requests, although 2P-IaaS achieved a high acceptance ratio among the benchmarks. This highlights an existing scalability challenge in the presented composition algorithm.

Another limitation exists in the adopted semantic similarity equation 4.3. In the sense that, the adopted equation does not differentiate between the over/under- provisioning cases, i.e., if a customer asks for a VM with $CPU = 2GHz$, and the available VMs at the pool are with $1GHz$ and $3GHz$, the output of the similarity equation will be the same. However, the customer will not be satisfied with the resource with $CPU = 1GHz$, since

it is under his estimation. Customers all the time are looking for over provisioning with lower cost, at least their exact requirement but not under provisioning resources.

4.7 Summary

This chapter presents an efficient heuristic approach for cloud IaaS resource allocation, 2P-IaaS. The novelty of this approach resides in the employment of ontology-based modeling in representing a unified VRP for datacenter resources, employing reasoning capability, and exploiting semantic similarity, closeness centrality, and random walk techniques while solving the resource allocation problem.

A brief background of resource modeling was introduced. Key foundation concepts, which the presented approach relies on, were presented, including generalized VRP, semantic similarity, closeness centrality and random walk techniques. The presented approach exploits the inherent characteristics of ontology-based modeling to discover the requested resources efficiently and adopts semantic similarity measurements for evaluation. Furthermore, closeness centrality and random walk discovery were used to assemble the discovered resources forming the desired topology. The use of ontology-based modeling supports the resource-allocation-efficient solution in multiple directions. It (1) unifies diverse resource representation in the pool, (2) automates the composition algorithm that subsequently automates the resource allocation solution, and (3) exploits the reasoning capabilities of the ontology in achieving model consistency and performing classification operations.

The formation of VRP in advance reduced the complexity of the algorithm as well as utilizing the associated reasoning operation. The applicability of semantic-based concepts along with reasoning capabilities as an IaaS resource allocation technique was proven using simulation. In addition, experiments confirmed that using ontology-based modeling integrated with the key foundations in a composition approach achieved an efficient resource usage goal, and contributed significantly to increasing an IaaS provider's revenue.

Chapter 5

Efficient Cloud IaaS Resource Allocation

Many real-world engineering optimization problems are characterized by multiple and often conflicting objectives and a huge search space to explore. Resource allocation in cloud datacenters is one of these problems. CSPs are still suffering from inefficient management for their massive datacenter resources. In the previous chapter a heuristic solution for IaaS resource allocation problem in datacenters was presented; this solution makes use of an ontology-based composition algorithm as an IaaS resource allocation technique. However, the proposed heuristic solution does not guarantee an optimal solution and is only adequate for an approximate near-optimal solution in reasonable computational time.

In this chapter, an efficient management system for datacenter resources is achieved, ensuring an optimal solution for the resource allocation problem. The desired effective management approach for datacenter resources makes use of a mathematical formulation for the IaaS resource allocation problem. In this approach, the proposed mathematical models rely on managing the convergence of network and cloud resources through a unified representation and even treatment of the resources. The proposed approach consists of two mathematical models that realize the unified treatment of the datacenter's resources in order to facilitate their management and to achieve the desired efficiency.

This chapter proceeds in two parts as follows: Section 5.1 gives an overview on the motivation for an efficient IaaS resource allocation solution. Section 5.2 introduces the Linear Programming stages in solving problems and presents some of the related work. Part I: A MILP-based Approach for Efficient Cloud IaaS Resource Allocation is described in

Section 5.3. Part II: A Cost-Efficient QoS-Aware Model for Cloud IaaS Resource Allocation in Large Datacenters is presented in Section 5.4. In Section 5.5, the simulation environment and the performance evaluation of the presented models are discussed. Finally, Section 5.6 concludes the chapter.

5.1 Overview

The development of new intensive computational models like MapReduce [129] and the desire for high-performance applications has led to an increased demand for IaaS services. These applications demand a large amount of resources and utilize cloud computing infrastructures for storing and processing large sets of data [169]. Also, most of these computationally intensive applications require networking between their components, resulting in the need for managing the convergence between cloud infrastructure and networking resources [25].

Popular cloud infrastructure providers such as Amazon EC2 offer their services in terms of VMs without having any network bandwidth guarantees as performance metrics [169]. Furthermore, the literature has paid little attention to managing the convergence of networking and other infrastructure resources. As a result, the absence of such assurances affects the overall performance of the provisioned services and impacts the CSP's revenue [169]. Efficient cloud and network resource management significantly impacts the datacenter utilization level. Proper datacenter resource management can benefit both CSPs and customers in several ways: it allows CSPs to use their resources efficiently, accept more requests, and increase their revenue, while better meeting customers' QoS expectations.

Adoption of a mathematical formulation for the resource allocation problem is motivated by the fact that the quality of the solution obtained from the ontology-based heuristic algorithm lacks optimality. Furthermore, ontology-based modeling impacts the efficiency of the proposed composition algorithm, particularly in the presence of the growing on-

tology space. In addition, restricting the problem space in one or more dimensions, to enable efficient heuristics, comes at the expense of the optimality of the obtained solution. Thus, mathematical modeling was adopted when formulating the IaaS resource allocation problem in datacenters.

Linear Programming (LP) as a mathematical representation model has been extensively used in many real-world applications including industrial and governmental applications, also military applications exploit LP in accurate representations. The popularity of LP stems from its ability to model large and complex problems. Furthermore, LP models provide the users the ability to solve such problems in a reasonable amount of time by using effective algorithms and modern computers. LP models deal with minimizing or maximizing problem of a linear function in the presence of linear equality and/or inequality constraints [170]. Furthermore, the evolution of simplex methods for solving linear programs provides considerable insight into the theory of LP and yields an efficient algorithm in practice.

5.2 Linear Programming Modeling and Related Work

The modeling and analysis of a linear programming problem evolves through several stages. These stages are defined as the problem formulation phase, the model construction phase, the model solving phase, the model testing phase, and the implementation phase [170]. In the problem formulation phase, a detailed study of the system has been carried out and the specific problem that needs to be solved has been identified. This detailed study includes the system constraints, restrictions, or limitations, and the objective function(s). In the model construction phase, an abstraction of the defined problem has been constructed through a mathematical model formulation. This mathematical model takes the form of minimizing or maximizing a linear function with a set of inequality constraints. The following is an illustrative example of a simple minimization problem. Consider the following

linear problem.

$$Z = Minimize \quad 2x_1 + 5x_2 \tag{5.1}$$

$$Subject\ to. x_1 + x_2 \geq 6 \tag{5.2}$$

$$-x_1 - 2x_2 \geq -18 \tag{5.3}$$

$$x_1, x_2 \geq 0 \tag{5.4}$$

In this case, the model defines two decision variables x_1 and x_2. The objective function to be minimized is $2x_1+5x_2$. The optimization problem is thus to find values for x_1 and x_2 having the smallest possible objective value. It is worth mentioning that the results obtained from solving this model represent solutions to the model and not necessarily solutions to the actual system unless the model adequately represents the true situation. The third step is to derive a solution. A proper solving technique (e.g., the simplex method) is chosen. Such a technique comes up with one or more optimal solutions. Also, a heuristic or an approximate solution may be determined along with some assessment of its quality. In the case of multiple objective functions, one may seek efficient or Pareto-optimal solutions, where a further improvement in any objective function value is necessarily accompanied by a decline in another objective function value. In the model testing phase, assessment of the solutions obtained by the model is carried out. This assessment performs an examination of the solution from the model and its sensitivity to relevant system parameters. This analysis provides insights into the system and it can also be used to ascertain the reliability of the proposed model. At this stage, one may wish to enrich the model further by incorporating other important features and restructuring the system model, or, on the other hand, one may choose to simplify the model. The final stage is the implementation phase, where the primary purpose of a model is achieved by assisting interactively in the decision-making process.

In the following section, the aforementioned stages are used in formulating the IaaS

resource allocation problem. The objective in this formulation is to maximize the resource utilization level in the datacenter with a set of capacity violation constraints. Furthermore, this formulation aims to solve the previous heuristic algorithm challenge regards optimality while considering the composition concept. In the literature, many proposals have addressed the resource allocation problem resulting from interconnected resources constrained by availability and capacity. In this context, allocation of the interconnected resources shares many similarities with the traditional VNE problem. VNE aims to embed a virtual network request composed of virtual nodes and virtual links into the substrate network. A number of approaches have been proposed in that regard. Some of these approaches have decomposed the problem into two phases: the node mapping phase and link mapping phase; other methods have solved these two phases simultaneously or cooperatively [171] [172].

In [173], Farooq Butt *et al.* proposed a topology-aware VN embedding and re-optimization exploiting migration approach. It is worth mentioning that most of the VNE proposal focused on CPU and network resources. Other existing work reported in the literature has solved the resource allocation problem by employing different aims and assumptions. Much of the research has addressed server consolidation and live VM migration as resource allocation techniques, e.g., [174] [175] [176] [177]. In [178], the authors explored the performance overhead of these techniques on the datacenters, in addition to the migration cost. In [179], Zhani *et al.* proposed a VDC Planner as a migration-aware dynamic virtual datacenter embedding framework. This proposal aimed to achieve high revenue while minimizing the total energy cost over time. The authors considered the migration cost in their resource allocation approach. In addition, they aimed to increase the provider revenue, decrease the scheduling delay, and save energy by applying the migration concept to the VDC embedding problem.

However, our approach differs from the aforementioned research in several ways: (1) our solution works without access to the details of the physical resources, utilizing a gen-

eralized virtual resource pool and the abstraction provided by a unified VR model, (2) the presented solution aims at converging the management of networking and computing resources through generalized representation and combined management, (3) the adopted VR model simplifies the datacenter resource management and allows uniform treatment of all the resources, (4) the proposed approach avoids using the migration concept with its associated cost and processing overheads, and (5) the integration of semantic similarity with closeness centrality confirms the operation of the optimization model for resource allocation solutions.

5.3 Part I: MILP-based Approach for Efficient Cloud IaaS Resource Allocation

In this section, an IaaS resource allocation approach for datacenter resources is proposed based on mathematical modeling. This approach is formulated as a two-phase Mixed Integer Linear Program (MILP-2P-IaaS): (i) mapping of hosting resources, and (ii) connectivity composition. The presented model integrates the semantic similarity formulation with closeness centrality under the unified combined manager; the *composer*. The proposed approach has been formulated as a multi-objective optimization model that produces several non-dominated solutions. Optimality in multi-objective optimization problems is known as Pareto optimality, where none of the obtained solutions is better than the others, with respect to all objectives. Moreover, this set of "trade-off" solutions naturally reflects the multi-criteria decision making used by CSPs. Obtaining multiple non-dominated solutions allows CSPs to have various solutions that can be used in different decision-making scenarios. Furthermore, the proposed multi-objective model has been solved using the ϵ-constraint as a Pareto-set for an optimal solution.

The mathematical model presented in this section integrates the semantic similarity and closeness centrality concepts into one optimization objective function. The importance

of combined management for cloud and network resources over the generalized VRP is highlighted as a way of unification; this way of unification makes the management flexible. The unified VR ontology model that populates the pool, as shown in Figure 4.2, can be mathematically represented as follows. Let $VR = (P, S, I)$ be a VR, P represents a set of requested resources with their specific parameters, S is a service attribute that defines the provisioned service, and I is the set of interfaces that represents the connectivity of one VR with another VR. This model is applied to all the diverse resources respecting the defined QoS classes. Furthermore, semantic similarity as represented in Eq. (4.3) and closeness centrality as represented in Eq. (4.5) are integrated into the proposed multi-objective MILP model. In addition, closeness centrality in Eq. (4.5) has been adapted to represent the distance of the shortest path between nodes in terms of number of hops. Adopting shortest path calculation ensures the locality-aware concept is followed and saves bandwidth.

5.3.1 MILP-2P-IaaS Resource Allocation Approach Formulation

The proposed MILP-2P-IaaS resource allocation approach has been formally defined as working on the VRP denoted by V. The presented multi-objective model aims to maximize the similarity integrated with the centrality between the requested and the available resources; solving this MILP model achieves the optimal resource allocation in the data-center.

The two proposed phases, mapping of hosting resources and connectivity composition, execute in sequential order. As the customer submits his request, the *composer* starts to serve the request by executing Phase-1, the mapping of hosting resources phase. The *composer* employs the integrated semantic similarity and closeness centrality model while searching for candidate VRs for the received request. After completing Phase-1, Phase-2, the connectivity composition phase, is put into operation. In this phase, the *composer* exe-

cutes the shortest path calculations looking for the shortest connectivity paths between the successful candidate resources mapped in Phase-1, then allocates the appropriate network resources. The *composer* efficiently manages the integration of the semantic similarity and closeness centrality models in both phases in order to obtain an optimal resource allocation solution. An accepted IaaS request is counted when it passes the two phases successfully. Otherwise, it is rejected. Both techniques, semantic similarity and closeness centrality, participate efficiently in mapping the hosting resources and in composing the connectivity paths.

5.3.2 Phase-1: Mapping of Hosting Resources

In Phase-1, the integration between the similarity evaluation in Eq. (4.3) with the closeness evaluation in Eq. (4.5) is achieved in the following way. When considering a set of IaaS requests denoted by N, each request $n \in N$ asks for a set of resources as represented by $r \in R$. The created pool V represents the available free VRs and each resource $v \in V$ is described as having a set of attributes $a \in A$ (e.g., CPU, memory, storage, bandwidth). The total capacity of each VR from attribute a is defined by C_{va} and is limited by the physical hosting resource capacity. The customer requests q_{ra} units of attribute a from resource r. The shortest distance between two VRs v and v' on the pool is defined by $d_{v,v'}$. To decide on the mapping of a resource and the closeness between resources in the pool, the following decision variables are defined:

- $x_{rv} = 1$, if virtual resource v is mapped to the requested resource r and 0 otherwise.
- $y_{v,v'} = 1$, if a shortest path between resources v and v' is selected and 0 otherwise. Where v,$v' \in V$.

The mapping of the hosting resources phase model can be defined as follows:

Objective function:

$$Eval_{rv} = Max. \sum_{r \in R} \sum_{v \in V} \sum_{a \in A} \left(\left(1 - \left| \frac{C_{va} - q_{ra}}{Max_a - Min_a} \right| \right) * x_{rv} \right) + \left(\frac{1}{\sum_{v,v' \in S_n} d_{v,v'}} * y_{v,v'} \right), \forall n \in N, S_n \subset V \tag{5.5}$$

Where, Max_a and Min_a are the maximum and minimum values of the VR attributes (CPU, mem, and storage) bounded by physical server capacity, and $d_{v,v'}$ represents the distance between resource v and v' in a subset of previous discovered resources S_n in the pool V.

The first term represents the semantic similarity evaluation and the second term represents the closeness centrality evaluation. It is clear that the Phase-1 model has a multi-objective function that the given formulation aims to solve. Different techniques are introduced to efficiently solve the multi-objective function. One such example is the adoption of the ϵ-constraint method. This method states that one objective is selected to be optimized, and all the remaining objectives are converted into constraints by setting an upper bound to each. In the presented model, the first term representing the semantic similarity evaluation is selected as the primary objective, and the closeness centrality term is turned into a set of bounded constraints.

1. *Objective function:*

$$Eval_{rv} = Max. \sum_{r \in R} \sum_{v \in V} \sum_{a \in A} \left(\left(1 - \left| \frac{C_{va} - q_{ra}}{Max_a - Min_a} \right| \right) * x_{rv} \right) + \left(\frac{1}{\sum_{v,v' \in S_n} d_{v,v'}} * x_{rv} \right), \forall n \in N, S_n \subset V \tag{5.6}$$

2. *Constraints:*

- Assignment constraints: A resource r is assigned to only VR v.

$$\sum_{v \in V} x_{rv} \leqslant 1, \forall r \in R \tag{5.7}$$

- Assignment constraints: A virtual resource v is assigned to only resource r in request n.

$$\sum_{r \in R} x_{rv} \leqslant 1, \forall v \in V \tag{5.8}$$

- Capacity Constraints: Requested resource r capacity from attribute a doesn't exceed the capacity of the VR v from attribute a.

$$\sum_{r \in R} x_{rv} * q_{ra} \leqslant C_{va}, \forall v \in V, \forall a \in A \tag{5.9}$$

- Connectivity path and VRs Linking constraint: Ensures that each allocated path $y_{vv'}$ is represented by only two VRs x_{iv} and $x_{jv'}$ as source and destination.

$$y_{v,v'} - x_{iv}.x_{jv'} \leqslant 0, \forall v, v' \in V \times V, \forall i, j \in R \times R \tag{5.10}$$

- Path satisfaction constraint: Ensures that a unique path is selected.

$$\sum_{v \in V} \sum_{v' \in V} y_{v,v'} \leqslant 1, \forall r \in R, v \neq v' \tag{5.11}$$

- Connectivity path length constraint: Connectivity path length should not exceed the defined ϵ value.

$$y_{v,v'}.d_{v,v'} \leqslant \epsilon, \forall v, v' \in V \times V, v \neq v' \tag{5.12}$$

Where ϵ is altered to generate the entire optimal solution set.

At the end of mapping the hosting resources phase, a set of VRs, which satisfy part of the request, are reserved in a mapping list and fed to Phase-2. Up to this point, the request is considered as partially satisfied.

5.3.3 Phase-2: The Connectivity Composition

As soon as Phase-1 is finished, Phase-2 receives the mapped VRs from Phase-1 as an input, and starts allocating resources for the connectivity path between the mapped resources respecting the customers' QoS requirements.

Consider the situation where a request n is given with requested bandwidth b_n as a QoS constraint. Each link $l \in L$ in the network topology has a total capacity of B_l. This capacity is updated after each successful composition. After the completion of Phase-1, a set of requested connections K is generated among the successfully mapped resources. It is necessary to map these connections on a set of generated shortest paths $p \in P$. In order to decide on a successful path composition, the following decision variable is defined:

- z_k^p is a binary decision variable indicating whether the connection k is assigned to path p or not.

The connectivity composition phase model can be defined as follows:

1. *Objective function:*

$$f = Max. \sum_{k \in K} \sum_{p \in P} \frac{1}{c_p} . z_k^p \tag{5.13}$$

   where c_p represents the number of hops in path p.

2. *Constraints:*

- Connection satisfaction constraint: ensure a unique allocation for each connection.

$$\sum_{p \in P} z_k^p \leqslant 1, \forall k \in K \tag{5.14}$$

- Link capacity constraint: total allocated bandwidth on link l for all requested connections k should not violate the link capacity.

$$\sum_{k \in K} \sum_{p \in P} b_k . z_k^p . \delta_l^p \leqslant B_l, \forall l \in L \tag{5.15}$$

where : $\delta_l^p = 1$, if path p uses link l and 0 otherwise.

By the end of this phase, the request is completely satisfied, and the infrastructure is completely composed and delivered to the customer. The semantic similarity evaluation is adopted in the mapping of the hosting resources as a pivotal term. Closeness centrality is interpreted differently in the MILP-2P-IaaS approach phases. In Phase-1, closeness centrality ensures the closeness of the selected set of VRs as candidate resources to each other. This helps in saving bandwidth and improves datacenter resource utilization. In Phase-2, closeness centrality is completely tuned to the shortest path implementation; this implementation ensures the proximity of the allocated path end-points. As the two phases are sequentially and cooperatively executed, resource allocation in the datacenter is optimized. The mapping of the hosting resources phase plays a crucial role in the datacenter resource management. Accordingly, proper mapping for the requested resources in Phase-1 leads to an overall efficient resource usage and a high acceptance ratio.

Despite the optimal solution obtained by the MILP model, it is challenged by a scalability limitation and growing computational complexity. The growth in the scale of datacenters, along with the increasing demand in IaaS requests, impacts the datacenter's utilization level. Indeed, the MILP mathematical model with a large number of variables and constraints has a scalability limitation issue; it provides an optimal solution at the

expense of the applicability of the solution. Finding a near-optimal or optimal resource allocation solution for MILP is still computationally intractable since MILP is still known to be an NP-Hard problem. The following section presents an efficient model that addresses the scalability issue in datacenters.

5.4 Part II: A Cost-Efficient QoS-Aware Model for Cloud IaaS Resource Allocation in Large Datacenters

The increasing popularity of cloud computing is leading to the emergence of large virtualized datacenters to fulfill the increasing demands of complex and dynamic IT systems and services. In large datacenters, CSPs are still challenged by how to handle and serve a large number of IaaS requests and manage their massive repositories efficiently. CSPs aim to realize the potential benefits given by the cloud computing paradigm without sacrificing their profits.

Although deploying large datacenters consumes an enormous amount of electrical power resulting in high Operational Cost (OPEX), they are still attractive repositories for enterprises to deploy their services. Moreover, large-scale datacenters are still benefiting CSPs economically and technically in several ways: (1) supporting bulk deployments of large data mining jobs (e.g., indexing the web) [180], (2) providing economies of scale, and (3) benefiting large numbers of similar servers that yield relatively low manning requirements and ease management [181].

Indeed, inefficient resource management within large datacenters significantly escalates power consumption and increases their OPEX [182]. Therefore, adopting an efficient resource allocation solution in large datacenters improves the datacenter's utilization level, which consequently reduces its OPEX [3].

Shortcomings in earlier solutions, MILP-2P-IaaS and 2P-IaaS, make them impractical for large datacenters. Furthermore, finding an optimal solution is still computationally too

expensive. In order to circumvent these shortcomings, a cost-efficient model acquainted with QoS requirements is defined as a solution to the IaaS resource allocation problem in large datacenters. This solution makes use of large-scale optimization tools and proposes a CG formulation for the IaaS resource allocation problem (RA-IaaS-CG). The proposed solution scales well in large datacenters and realizes a cost-efficient model for CSPs, in the sense that it minimizes the cost of VI deployment with efficient datacenter resource usage while respecting the customers' QoS requirements.

In the literature, there is a host of research efforts that have addressed resource allocation in large datacenters from different perspectives. In particular, energy-aware resource allocation in large datacenters has been exhaustively studied in the literature. Since energy consumption is proportional to the service provider's OPEX, energy-aware proposals share many similarities with our work. The existing literature has focused on reducing OPEX and improving datacenter performance while adopting heuristic algorithms and mathematical modeling formulations as in [183], [184], and [174].

The work presented in this section is characterized by using a large-scale optimization tool in formulating the IaaS resource allocation problem. This tool provides a quite promising solution for the resource management issues in large datacenters. To sum up, the proposed resource allocation approach in the following section has a crucial improvement over the aforementioned related work regarding the scalability and the optimality of the obtained solution in a reasonable computation time.

5.4.1 Column Generation Preliminaries

When a problem is too large and/or complex to be solved all at once, it might be decomposed into smaller and easier problems. Choosing a sound decomposition method is critical to solving difficult problems successfully. Extensive studies have been made on solving numerous real-life problems using various decomposition methods. Adopting decomposition

methods in solving these problems requires considerable related knowledge and analysis, since the decomposition techniques are highly problem-dependent. In this section, we are focusing on solving the IaaS resource allocation problem in the presence of a large number of IaaS requests and managed resources using an efficient decomposition method. To better illustrate the key idea of the decomposition method, let us call the following model the *master problem* (MP) [185].

$$z^*_{MP} := Min \quad \sum_{j \in J} c_j x_j \tag{5.16}$$

$$Subject\ to. \sum_{j \in J} a_j x_j \geq b \tag{5.17}$$

$$x_j \geq 0, \forall j \in J \tag{5.18}$$

The formulation above denotes the *primal* problem. The *dual* model is found by transposing the primal problem as follows:

$$z^{MP} * = Max \quad \sum_{i \in I} b_i^t y_i \tag{5.19}$$

$$Subject\ to: \sum_{i \in I} a_i^t y_i \geq c \tag{5.20}$$

$$y_i \geq 0 \tag{5.21}$$

Where b_i^t is the transpose of b_i and a_i^t is the transpose of a_i. The dual variables are denoted by the vectors X and Y. A strong duality states that the optimal solution value for the dual problem is equal to the optimal solution value for the primal problem. This means that an optimal solution for a primal problem can be obtained by solving its dual model, and vice verse. Hereafter, we will focus on the *Primal* model, meanwhile, using the dual variables.

In linear programming optimization problems, the objective function can be improved if

there is a variable that has a positive reduced cost (for maximization problems). Instead of testing all of the possible variables that may improve the objective function, a subproblem is formulated using dual information to determine the variable that has the highest reduced cost. The new variable (i.e., column) is then passed to the problem that contains a subset of variables by adopting the simplex method [170] as one of the most successful methods in solving LP problems. In every iteration, the solver (i.e. CPLEX) looks for a non-basic variable to price out and enters the basis. Given non-negative dual vector ϕ for dual variables, we aim to find x_j with the minimum reduced cost $\bar{c}_j$, where $\bar{c_j} = c_j - \phi^t a_j$. In the simplex method, this is accomplished by computing the reduced cost $\bar{c_j}, \forall j \in J$, and then selecting the most negative one. When the number of variables (i.e.,$|J|$) is huge, this computation becomes complex and time consuming, it is even impractical to enumerate all the variables explicitly. In order to solve this issue, we work on a restricted subset of $\bar{J} \subset J$ that constitutes the Restricted Master Problem (RMP).

In that context, it is worth mentioning that the basic principle of CG [185] is similar to the simplex method. That is, in every iteration of CG, we select profitable variables to enter the basis. However, there are two main differences between the simplex method and CG. Firstly, instead of working with the complete set of variables, CG only deals with a reasonably small subset of variables denoted by RMP. Secondly, the pricing step of CG is more efficient than the simplex method for large problems in the sense that, instead of computing the reduced cost $\bar{c_j}, \forall j \in J$, the pricing operation of CG is to find a column with improved negative reduced cost by solving an optimization problem, called the pricing subproblem. The pricing subproblem is defined as follows: Let x_i and ϕ be the primal and dual optimal solution of the current RMP, respectively. Then the subproblem ($\bar{c}_j^* = min c_j - \phi^t a_j$) performs the pricing. If there exists some efficient algorithm that solves the pricing subproblem in reasonable time less than the consumed time in computing the reduced costs for all non-basic variables, CG becomes more computationally efficient than the simplex method and speeds up the early stages of CG. Moreover, because CG only

deals with a subset of variables, it needs much less computational space than the simplex method.

After solving the RMP, we get the dual vector ϕ. In order to evaluate the optimality of the current solution, check if $\bar{c}_j^* \geq 0$. If so, there is no negative $\bar{c}_j, j \in J$ and the current RMP solution is optimal, and the algorithm stops. Otherwise, the derived column from the optimal subproblem solution is added to the RMP, and repeated by re-optimizing the RMP. The key step in CG formulation is deciding how to model the pricing subproblem and solve it efficiently. CG formulation is mainly based on the Dantzig-Wolfe decomposition method, whose general methodology is to find an optimal solution by solving the problem using only a subset of the problem's variables, or associated columns [186] [185]. To make the new model tractable, columns are generated iteratively. These columns will be included in this subset only if they will improve the objective function. It is obvious that Dantzig-Wolfe decomposition is closely related with column generation.

Consider the following formulation as an example to outline the Dantzig-Wolfe method.

$$z^* = Min \quad c^t x \tag{5.22}$$

$$Subject\ to : Ax \geq b \tag{5.23}$$

$$x \in X \tag{5.24}$$

In this formulation, we are interested in performing the optimization over a discrete set X. In order to Dantzig-Wolfe decompose a problem so that the whole problem $Ax = b$ does not have to be solved at once, the constraint matrix A in the formulation should be on a certain structure. This structure is known as block diagonal structure, where the matrix is divided into blocks with non-zero coefficients. These blocks that constitute the constraint matrix are solved independently on subproblems. Assume that the matrix A

has a block diagonal structure represented as:

$$Ax = \begin{pmatrix} B_1 & & & \\ & B_2 & & \\ & & .. & \\ & & & B_K \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ . \\ x_K \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ . \\ b_K \end{pmatrix} \tag{5.25}$$

Dantzig-Wolfe decomposition yields K subproblems, each with its own constraints and associated dual variables. In Dantzig-Wolfe decomposition X is replaced by $conv(X)$, each $x \in X$ can be represented as a convex combination of extreme points $\{x_p\}_{p \in P}$ plus a non-negative combination of extreme rays $\{x_r\}_{r \in R}$ of $conv(X)$, i.e.,

$$x = \sum_{p \in P} x_p \lambda_p + \sum_{r \in R} x_r \lambda_r, \sum_{p \in P} \lambda_p = 1 \tag{5.26}$$

where the index set P and R are finite. Further, by substituting for x in

$$\bar{c^*} := min \quad \left\{ (c^t - \phi^t A)x - \phi_0 \mid x \in X \right\} \tag{5.27}$$

This subproblem is an integer linear program. When $\bar{c^*} \geq 0$, there is no negative reduced cost, and the program terminates. When $\bar{c^*} \leq 0$ and finite, an optimal solution to Eq. 5.27 is an extreme point x_p of $conv(X)$, and the corresponding column is added to the RMP. In the general CG approach, the optimal solutions to RMP are often integral or close to integral. If the solution is not integral, round heuristics can be used, or the method of branch and price can be employed. The key idea is to branch on the column variables that are preventing the integrality constraints to be satisfied, then to proceed in a manner similar to branch and bound. The main advantage of Dantzig-Wolfe decomposition over the original model, is decomposing the original complex problem into a set of well-studied easy problems and using existing algorithms and methods to solve the subproblem

efficiently.

5.4.2 IaaS Resource Allocation Using Column Generation Formulation (RA-IaaS-CG)

The proposed cost-efficient model has been formally defined using CG working on the preexisting VRP V. This pool is formed by partitioning the physical resources into a set of VRs following QoS class distribution [131]. The use of the Column Generation technique means that the IaaS resource allocation problem is decomposed into a master problem and a pricing problem. The key idea of Column Generation is to work only with a restricted number of columns, forming the RMP. The pricing problem is solved iteratively as long as it has a significantly reduced price; the corresponding column is added to the RMP. Otherwise, the current solution is optimal. A column in this formulation represents a feasible allocation of VRs to an IaaS request. Linking the master and pricing problem is achieved by transferring the optimal values of the dual variables corresponding to the master problem constraints to the pricing problem.

In the Column Generation formulation, the IaaS resource allocation problem is reformulated in terms of Independent Infrastructure Allocating Configurations (IIACs). An IIAC indexed by c is the principal component in this formulation that defines an allocating configuration of one IaaS request. Accordingly, an IIAC can be defined as a set of VRs and connections used to satisfy customers' QoS requirements (CPU, memory, storage, and bandwidth). For each infrastructure configuration c, there exist a set of virtual resources $v \in V_q$ belonging to a defined QoS class $q \in Q$. Associated with each configuration c, is a set of paths p which satisfy the requested QoS bandwidth constraint. This formulation adopts the VRs published in the pool V to create different IIACs.

We denote by C, the set of all possible IIACs. We define (λ_c), $c \in C$ as a decision variable where $\lambda_c = 1$, if the IIAC c is used in the allocation solution, 0 otherwise. Thus,

based on the new formulation, the IaaS resource allocation in the best case selects a maximum of N IIACs. This case indicates that each IaaS request is granted by a distinguished IIAC. An IIAC configuration $c \in C$ is defined by the vector $(a_n^c)_{n \in N}$, where a_n^c represents that IIAC c serves IaaS request n.

5.4.3 The Master Problem

The master problem is defined by the selection of maximum N configurations among the generated IIACs that minimize the service provider allocation cost. $COST_c$ is defined as the cost of configuration c, representing the total cost of all the VRs and the paths on that configuration for allocating an IaaS request as granted by IIAC c. The cost of configuration is calculated as follows:

$$COST_c = \sum_{v \in V_c} \zeta_v^c . c_v + \sum_{l \in L_c} b_l . c_l \tag{5.28}$$

where ζ_v^c is a parameter indicating that the VR v is used in configuration c, b_l is the bandwidth used on link l in the configuration c, c_v and c_l represents the costs of the VR and Vlink respectively in configuration c. The master problem formulation denoted by *Master-ILP* is defined as follows:

1. *Objective Function:*

$$Min. \sum_{c \in C} COST_c \lambda_c \tag{5.29}$$

2. *Constraints*:

   - The total used VRs cannot exceed the QoS classes' capacity C.

$$\sum_{c \in C} \sum_{v \in V_q} -\zeta_v^c . \lambda_c \geqslant -C_q, \forall q \in Q \qquad (\delta_q) \tag{5.30}$$

- The used BW for each link b_l cannot exceed the capacity of link l.

$$\sum_{c \in C} -\lambda_c . b_l \geqslant -B_l, \forall l \in L \qquad (\eta_l) \tag{5.31}$$

- Unique selection of one IIAC for serving IaaS request.

$$\sum_{c \in C} \lambda_c . a_n^c \geqslant 1, \forall n \in N \qquad (\alpha_n) \tag{5.32}$$

- Guarantee the maximum number of allocating configurations.

$$\sum_{c \in C} -\lambda_c \geqslant -N \qquad (\beta) \tag{5.33}$$

- Integrality constraint of master variable λ_c.

$$\lambda_c \in \{0, 1\} \tag{5.34}$$

5.4.4 Pricing Problem

The pricing problem becomes one of generating additional IIAC configurations, i.e., an additional column is added to the RMP constraints' matrix, in other words, generating a configuration that improves the current value of the master objective function. The pricing problem is defined as follows. Let $(\delta_q), (\eta_l), (\alpha_n)$, and (β) be the dual variables associated with the constraints (6.4), (6.5), (6.6), and (6.7) in the master problem. Then, the reduced cost of variable λ_c can be written as:

$$\overline{COST_c} = COST_c + \sum_{l \in L} \eta_l . b_l + \sum_{q \in Q} \delta_q + \beta - \sum_{n \in N} \alpha_n . a_n^c \tag{5.35}$$

In order to linearize the reduced cost expression Eq. (6.10) and represent it in terms of the pricing problem variables, the following decision variables are defined:

- z_n: a binary decision variable indicates if an IaaS request n is granted by configuration $c \in C$ or not.
- x_r^v: a binary decision variable indicates if a requested resource r is allocated to virtual resource v or not.
- y_k^p: a binary decision variable indicates if a connection k uses path p or not.

Then, the relation between the pricing problem decision variables and the coefficient of the master problem is derived as follows:

- For each $c \in C$ and $n \in N$:

$$a_n^c = z_n \tag{5.36}$$

- For each link $l \in L$:

$$b_l = \sum_{n \in N} \sum_{k \in K_n} \sum_{p \in P_c} \sum_{l \in L_p} \pi_l^p . b_n . y_k^p \tag{5.37}$$

  Where:

  $\pi_l^p = 1$, if link l is used in path p and 0, otherwise.

- For each $c \in C$ and resource $r \in R_n$:

$$\zeta_v^c = \sum_{n \in N} \sum_{v \in V} x_r^v \tag{5.38}$$

Accordingly, the reduced cost Eq. (6.10) can then be expressed with the following linear formulation:

1. *Objective:*

$$\overline{COST_c} = COST_c + \sum_{l \in L} \eta_l \times \sum_{n \in N} \sum_{k \in K_n} \sum_{p \in P_c} \sum_{l \in l_P} b_n \pi_l^p y_k^p + \sum_{q \in Q} \delta_q + \beta - \sum_{n \in N} \alpha_n . z_n \tag{5.39}$$

Where :

$$COST_c = \sum_{n \in N} \sum_{r \in R_n} \sum_{v \in V} x_r^v . c_v + \sum_{k \in K_n} \sum_{p \in P_c} \sum_{l \in l_P} b_n . \pi_l^p . y_k^p . c_l \tag{5.40}$$

2. *Constraints:*

- Ensure a unique allocation for each requested resource r in IaaS request n on a virtual resource v.

$$\sum_{v \in V_q} x_r^v \leqslant z_n, \forall r \in R_n, \forall n \in N \tag{5.41}$$

- Ensure a unique path allocation for each requested connection k in IaaS request n.

$$\sum_{p \in P} y_k^p \leqslant z_n, \forall k \in K_n, \forall n \in N \tag{5.42}$$

- Ensure that at least one allocated path p for each couple of virtual resources $(v\grave{v})$ is allocated (Linking the decision variables x and y).

$$x_r^v . x_{\hat{r}}^{\grave{v}} \geqslant y_k^p, \forall (v, \grave{v}) \in P \times P, \forall (r, \hat{r}) \in K_n \times K_n. \tag{5.43}$$

- Ensure that the connectivity path length does not exceed ϵ.

$$L(p) \leqslant \epsilon, \forall p \in P_n \tag{5.44}$$

- Ensure that the generated infrastructure configuration can handle the allocation of maximum one request.

$$\sum_{n \in N} z_n \leqslant 1 \tag{5.45}$$

5.4.5 Solving the RA-IaaS-CG Model

To start solving the RA-IaaS-CG model, the RMP has to be initialized by using a set of dummy columns with high cost. Lower bounds for ILP problems can be computed through straightforward relaxation. The LP relaxation of *Master-ILP*, denoted by *Master-LPR*, is obtained by relaxing the integrality constraint in Eq. (6.8) to be $\lambda_c \in [0, 1]$. The procedure for solving this model consists of a nested double loop and can be described as follows:

1. Initialize the RMP with a set of dummy columns with large cost.
2. Solve the *Master-LPR* using CPLEX solver.
3. Pass the optimal dual variables to the pricing problem.
4. Solve the pricing problem optimally to find a new column.
   (a) If a column with negative reduced cost has been found, add this column to the *Master-LPR* and re-optimize. Otherwise, *Master-LPR* is optimally solved.
   (b) To calculate the integer solution of *Master-ILP*, the integrality constraint is re-established again on λ and then the CPLEX solver proceeds with the branch-and-bound procedure.

It should be noted that the termination of the *Master-LPR* solution indicates that the optimal solution lower bound that is very tight to the *Master-ILP* has been reached. In order to obtain an integer solution to *Master-ILP*, CG needs to be embedded within a branch-and-bound framework. So, re-establishing the integrality constraint again and applying a branch-and-bound procedure to the master problem will provide an upper bound solution for the IaaS allocation problem. Adding additional columns iteratively in the pricing problem operation, is necessary to solve the *Master-LPR* at non-root nodes of the branch-and-bound search tree. Otherwise, solving the *Master-LPR* over the existing columns is unlikely to find an optimal or even feasible solution to the original problem.

5.5 Performance Evaluation

In this section, the efficiency of the proposed models MILP-2P-IaaS and RA-IaaS-CG are evaluated via simulation. The setting of the conducted simulation is given, along with a description of the experiments, the evaluated performance metrics, as well as the results obtained.

5.5.1 Simulation Settings

In the simulation environment, a datacenter is constructed with 80 physical machines, 4 top-of-rack switches, 4 aggregate switches, and 2 core switches. The intra-datacenter network topology used is the VL2 topology as described in [3], which provides a full bisection bandwidth in the datacenter network. Each physical machine has a 4 $CPUcore$, 8 GB of memory, 1 TB of storage and contains a 1 $Gbps$ network adapter. The inter-switch bandwidth in this three layer topology is 1 $Gbps$.

IaaS requests are generated randomly following a uniform distribution covering three defined QoS classes: gratis, middle, and production [131]. In a realistic scenario, IaaS requests may not arrive at regular time intervals. Also, online resource allocation approaches allow datacenter capacity fragmentation; this leads to a degradation in the datacenter utilization level [187]. Thus, in order to assess the performance of the proposed model under a bulk of IaaS requests, the MILP-2P-IaaS and RA-IaaS-CG approaches follow a periodical approach [188], where the planning time is divided into a set of periods $p \in P$. At the beginning of each period the *composer* collects the received requests and allocates them in small-batches in order to optimize the datacenter's resource utilization over time. Also, the *composer* guarantees the reserved resources for the requests running from previous periods during those requests' durations, so, the period length is variable and defined based on a random number of the received requests. Meanwhile, a random number of IaaS requests

leave the system.

The periodic small-batch allocation approach is modeled as follows: let P be the set of planning periods of time and $R(0)$ represent the initial set of IaaS requests. The set of IaaS requests $R(p)$ indexed by $p \geq 1$ is defined as:

$$R(p) = R(p-1) + R_{new}(p) - R_{drop}(p) \tag{5.46}$$

where $R(p-1)$ is the set of accepted IaaS requests at the end of period $p-1$. $R_{new}(p)$ is the set of new incoming requests, and $R_{drop}(p)$ is the set of ending IaaS requests at the outset of period p. An IaaS request is modeled in an abstract way as a set of VRs that are connected to a virtual switch as shown in Figure 4.3 [6]. The traffic pattern between the requested resources can be represented as a hose model with specific capacity on bidirectional links where the virtual switch is at the root of the output topology [189].

It is worth mentioning that, throughout this dissertation, all the presented mathematical models have been solved using IBM CPLEX solver [190] to ensure the obtained solutions optimality. The *Gap* calculation is one of the necessary calculations provided by IBM solver that proves the obtained solution's optimality; it is calculated as the difference between the optimal solution and the obtained feasible solution, i.e., the difference between the primal and the dual solutions. Thus, having a *Gap* equal to 0 achieves an optimal solution. However, obtaining a zero value *Gap* required integer relaxation and fractions occurrence, it means instead of allocating a VM to a request; the model will allocate 0.2 VM to the request, which is not realistic in the cloud context. Furthermore, our experiments in both parts achieved close to zero *Gap* values $\leq 5\%$. Hereafter, our claimed optimal solution in the obtained results can be expressed as near optimal with acceptable *Gap*.

5.5.2 MILP-2P-IaaS Performance Evaluation

To better illustrate the efficiency of the MILP-2P-IaaS model, the performance evaluation is carried out against the well known heuristics, SecondNet [5] and Oktopus [6]. These benchmarks used the same simulation environment settings and followed the periodical approach in serving the received requests. Furthermore, to quantify the performance of the MILP-2P-IaaS approach, **Acceptance ratio** and **Datacenter's resource utilization** were measured as performance metrics.

Evaluation Results

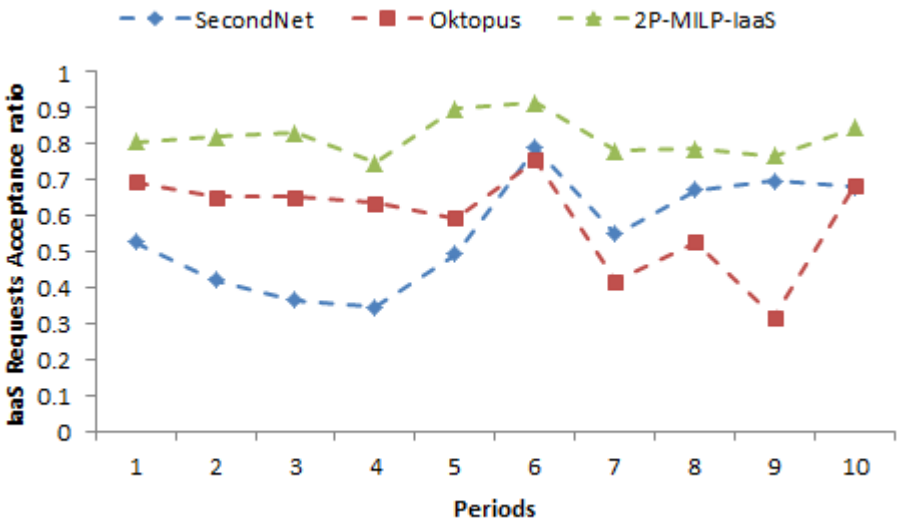
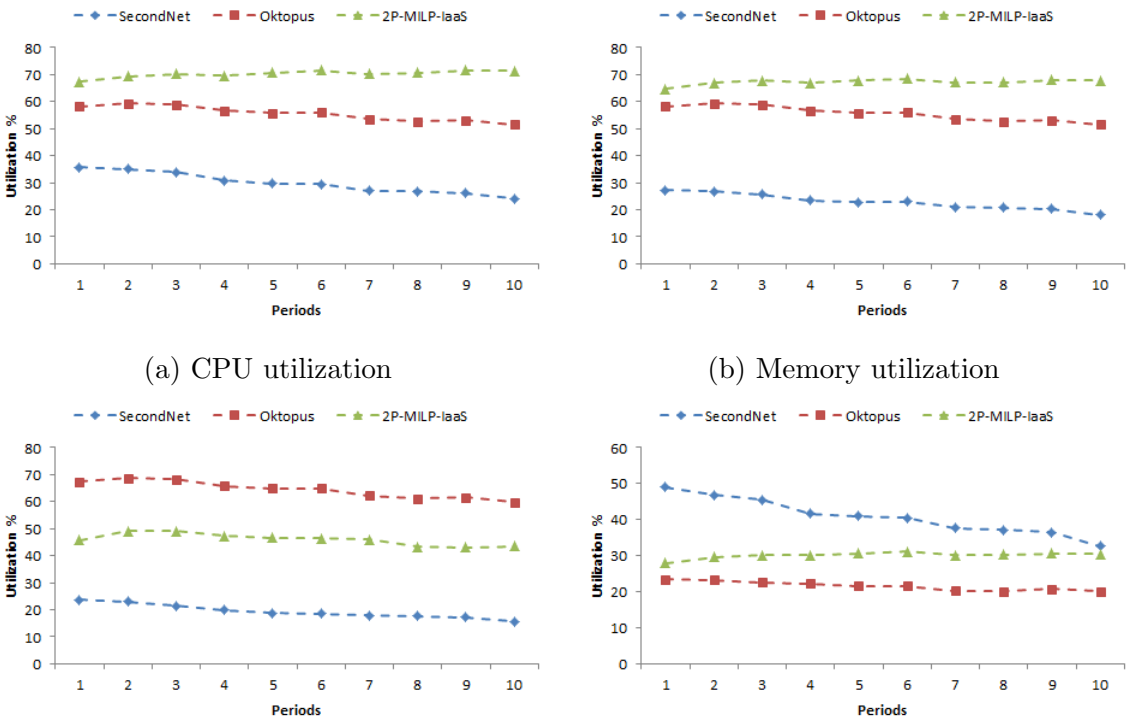


Figure 5.1: MILP-2P-IaaS Model: IaaS request acceptance

Through extensive experiments, the MILP-2P-IaaS acceptance ratio was evaluated among SecondNet and Oktopus. Figure 5.1 shows the average acceptance ratio for IaaS requests vs. the allocation time period. As expected, it was observed that MILP-2P-IaaS outperformed benchmarks Oktopus and SecondNet. The MILP-2P-IaaS approach achieved, on average, a 25% higher acceptance ratio than the benchmarks. It was already mentioned before that SecondNet adopted a resource allocation approach based on a bipar-



(a) CPU utilization

(b) Memory utilization

(c) Storage utilization

(d) Bandwidth utilization

Figure 5.2: MILP-2P-IaaS Model Resource Utilization

tite graph; the output dense connectivity topology impedes multiplexing multiple requests in the network. Thus, a high blocking rate occurred. The other benchmark, Oktopus, relied on a virtual cluster greedy allocation approach, illustrating that an IaaS request's acceptance ratio has peaks and valleys. This indicates that Oktopus is not performing well in a periodical approach serving a bulk of IaaS requests.

Moreover, Figures 5.2a, 5.2b, 5.2c, and 5.2d present the percentage of CPU, memory, storage, and bandwidth utilization vs. the allocation time periods, respectively. These figures confirm the efficient resource utilization of the MILP-2P-IaaS model on benchmarks. As a result of the high acceptance ratio, MILP-2P-IaaS showed a higher utilization level in memory and CPU closer to Oktopus as illustrated in figures 5.2a and 5.2b, however, it is acceptable in the presence of the high acceptance ratio. These findings have proved the efficiency of the proposed MILP-2P-IaaS model in allocating resources with a high acceptance ratio. In figures 5.2c and 5.2d the results show the superiority of the MILP-

2P-IaaS model in terms of storage and bandwidth utilization. Figure 5.2c indicates the efficiency of the similarity concept in allocating VRs. Likewise, figure 5.2d proves the effectiveness of the closeness centrality in saving bandwidth. Moreover, using the closeness centrality evaluation in MILP-2P-IaaS ensures the proximity of the mapped resources and reduces the blocking event occurrence. It is clear that SecondNet possesses the lowest resource utilization level that impacts its working resource allocation efficiency particularly, with the bulk of IaaS requests.

The results, as evidenced by these curves, confirm the MILP-2P-IaaS approach expectation in terms of resource usage efficiency. The results show the stability of the MILP-2P-IaaS model in resource utilization levels, particularly with a high acceptance ratio. Indeed, the integration of semantic similarity with closeness centrality in a multi-objective mathematical model and the unification of the datacenter's resource management, resulted in less blocking of IaaS requests.

5.5.3 RA-IaaS-CG Performance Evaluation

Assessment of RA-IaaS-CG is carried out using two simulation scenarios to confirm the superiority of the proposed approach regarding the obtained solution optimality and scalability. The defined scenarios are the optimality scenario and the scalability scenario. The optimality scenario evaluates the solution quality obtained from the proposed RA-IaaS-CG model against existing heuristics solutions, SecondNet [5], Oktopus [6], and our 2P-IaaS ontology-based approach, while the scalability scenario evaluates the scalability obtained from the solution against an adapted version from our MILP-2P-IaaS model as described in section 5.3. This model is denoted by MILP-2P-IaaS*, where the joint optimization model in this work has been adapted as a multi-objective function that optimizes the cost and is solved by the ϵ-constraints method. The scalability scenario has been evaluated with variable IaaS request loads.

To quantify the performance of the RA-IaaS-CG approach in the optimality scenario, the **Acceptance ratio** and **Datacenter's resource utilization** are measured as performance metrics. In the scalability scenario, the scalability of the proposed model is evaluated under increasing IaaS requests using the following performance metrics:

- Execution time denoted by T_e: measured as the time consumed to end the resource allocation process for the planning time.
- Solution cost: the cost of the obtained solution is calculated based on the unit price of the allocated resources.

RA-IaaS-CG is considered to perform well if it can achieve the lowest cost with the highest acceptance ratio.

Evaluation Results

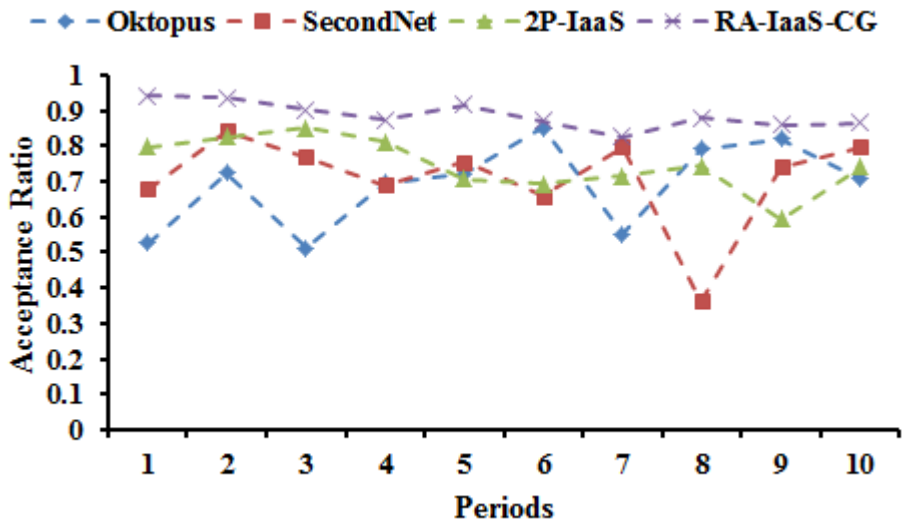
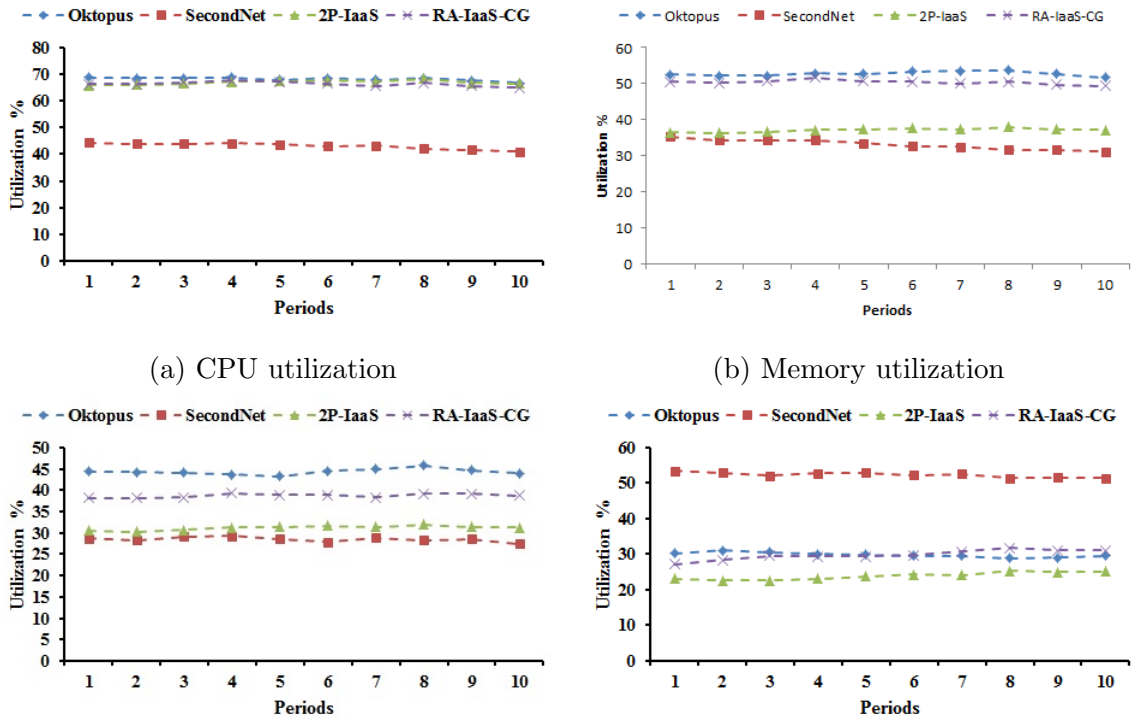


Figure 5.3: RA-IaaS-CG Model: IaaS request acceptance

Through extensive experiments, the IaaS request acceptance ratio was evaluated among SecondNet, Oktopus, 2P-IaaS. Figure 5.3 shows the average acceptance ratio for IaaS requests in terms of the allocation time periods. As expected, our RA-IaaS-CG outperformed



(a) CPU utilization

(b) Memory utilization

(c) Storage utilization

(d) Bandwidth utilization

Figure 5.4: RA-IaaS-CG Model Resource Utilization

benchmarks SecondNet, Oktopus, and 2P-IaaS. The RA-IaaS-CG approach achieved, on average, a 15% higher acceptance ratio than benchmarks. The designed heuristics, SecondNet and Oktopus, follow the same behavior in 5.5.2 while evaluating their performance under a bulk of IaaS requests at each period, in the sense that SecondNet and Oktopus achieved a higher blocking rate and acceptance ratio with peaks and valleys. The convergence results between RA-IaaS-CG and 2P-IaaS stemmed from applying the semantic similarity and the closeness centrality concepts in the composition technique and confirmed the near-optimality solution obtained by 2P-IaaS.

Moreover, figures 5.4a, 5.4b, 5.4c, and 5.4d present the percentage of CPU, memory, storage, and bandwidth utilization vs. the allocation time periods, respectively. Because of the high acceptance, RA-IaaS-CG showed a higher utilization level as shown in figures 5.4a, 5.4b, 5.4c and 5.4d, interchangeable with other benchmarks. This observation confirms the stability of the RA-IaaS-CG in resource utilization over the benchmarks. Moreover,

SecondNet and Oktopus achieved proper utilization levels in memory, storage, and BW even if they have a low acceptance ratio. This impacts the resource allocation efficiency in these benchmarks. Furthermore, these findings have proved the efficiency of our RA-IaaS-CG model in allocating resources with a high acceptance ratio.

In order to evaluate the scalability efficiently, five incremental IaaS traffic loads were applied, where each IaaS load was defined through the following parameters:

- Number of IaaS requests represented as ($\#N$).
- Total number of virtual nodes represented as ($\sum_{n\in N}|R_n|$).
- Total number of virtual links represented as ($\sum_{n\in N}|K_n|$).

Table 5.1: Scalability performance of RA-IaaS-CG vs. MILP-2P-IaaS (DC size=80)

IaaS Requests	**RA-IaaS-CG**		**MILP-2P-IaaS***	
($\#N$,$\sum_{n\in N}\|R_n\|$,$\sum_{n\in N}\|K_n\|$)	$cost$(X$)	T_e(sec)	$cost$(X$)	T_e(sec)
(40,157,117)	7906	907	18953	438
(50,198,148)	8798	1455	20986	1588
(60,243,183)	9197	1879	24127	2922
(70,281,211)	10342	2500	UFST	3600
(80,315,235)	10493	3561	UFST	3600

The results presented in Table 5.1 confirmed our expectations of RA-IaaS-CG regarding the scalability and the cost of the obtained solutions. In terms of the execution time, RA-IaaS-CG performed better than MILP-2P-IaaS* for different IaaS request batch sizes since MILP suffers from a scalability limitation. As the computation time increases dramatically with IaaS requests, an upper bound of 3600 seconds was defined for the execution time. In addition, the results shown confirm that RS-IaaS-CG has on average a lower solution cost than MILP-2P-IaaS*. It is noted that lower batch sizes of requests allow MILP to calculate a solution with feasible cost but not an optimal solution. However, allocation of higher batch sizes of requests exceeded the upper bound of execution time without even a feasible solution. Thus, an Unfeasible Solution (UFST) state is recorded as shown in Table 5.1.

5.6 Summary

This chapter presented an efficient approach for datacenter resource management aimed at improving the datacenter's poor resource utilization. The presented approach introduced unification management for datacenter resources under a combined controller. This unification method adopted the collected diverse VRs in the generalized VRP. Also, it defined a combined controller for control, management, and optimization of cloud and networking resource allocation. Two mathematical models for realizing the unified management approach and managing the convergence of network and cloud resources were defined: MILP-2P-IaaS and RA-IaaS-CG.

The presented MILP-2P-IaaS model is characterized by integrating semantic similarity with closeness centrality for efficient resource management. This integration ensured the convergence management of networking and computing resources in the proposed model. Furthermore, the achieved convergence confirmed that the provided model solved the datacenter performance degradation problem using an efficient resource allocation approach. Also, this model contributed significantly to increasing IaaS providers' revenue by accepting more IaaS requests. However, the rapid increase in IaaS demands and the growth in the scale of datacenters, prevents MILP from performing well at different load levels of IaaS requests. Indeed, the MILP-based model suffers from scalability issue and time-consuming operations.

The presented cost-efficient model RA-IaaS-CG has overcome the MILP limitations in terms of scalability; this model fits large datacenters and the growing demand in IaaS requests. The novelty of this model is the use of the large-scale optimization tool, Column Generation. The RA-IaaS-CG model is characterized by evolving scalability with solution optimality for efficient resource allocation. In addition, this model also contributes significantly to minimizing the datacenter's OPEX along with improving its utilization level. Furthermore, simulation results reveal significant gains in terms of resource utilization such

that the optimal solution has been achieved.

Chapter 6

A Distributed Economics-based Framework for Efficient IaaS provisioning in Geo-Datacenters

The increasing popularity of cloud computing has lead to the emergence of large virtualized datacenters hosting complex and dynamic IT systems and services. Moreover, the recent emergence of cloud computing as a new distributed computing paradigm offers a compelling solution for better application performance, scalability, and reliability. This evolving paradigm has brought along new profit opportunities for CSPs in hosting a wide range of prevalent large-scale distributed applications, e.g., high-performance computing, social networks, and most recently, MapReduce-based applications, yet CSPs have faced a set of challenges that arose with that paradigm. These challenges are mainly attributed to the ever-growing scale of the CSPs' managed datacenters, the unexpected growth of infrastructures demands, and the adoption of traditional provisioning approaches, in the sense that, existing provisioning systems rely mostly on a single controller in collecting the received infrastructure requests and performing the resource allocation in a centralized manner [183] [169]. Unfortunately, these approaches cannot provide timely and efficient infrastructure provisioning and do not scale well in a working environment with a large number of customers and resources. Indeed, CSP operations are becoming increasingly complex and time-consuming, and do not provide infrastructures with QoS guarantees. Consequently, scalability, computational complexity, and QoS-assurance challenges occur.

In the previous chapter, we introduced a solution for the IaaS resource allocation prob-

lem in large datacenters using an efficient decomposition approach. This decomposition approach presented a model (RA-IaaS-CG) that mainly relied on CG formulation as a large-scale optimization tool. Adopting this model afforded CSPs the scalability and cost-efficiency benefits of the resulting solutions. However, large datacenters have proven the model's economical inefficiencies in the presence of growing IaaS demands and expanding managed datacenters. Thus, there is an urgent need for a proper solution that favors affordable infrastructures for hosting novel emerging applications efficiently and improving customer satisfaction. In this chapter, we introduce a distributed framework that enables CSPs to offer efficient infrastructures for hosting large-scale distributed applications. In contrast to the existing centralized provisioning systems, the proposed framework mainly relies on a distributed architecture for CSPs that manages geographically distributed datacenters (Geo-datacenters) logically grouped in regions. Moreover, the desired architecture incorporates two decentralized resource allocation approaches, hierarchical and distributed. These working approaches employ efficient economic models in performing efficient coordination in the architecture. Furthermore, at the regional level, a mathematical model for the regional IaaS resource allocation problem (RCG-IaaS) is proposed. This model constitutes the core of the two decentralized approaches and it is formulated in CG as a large-scale optimization tool.

The remainder of this chapter is presented as follows. An overview on the distributed computing paradigm which has emerged is presented in section 6.1. Section 6.2 introduces the challenges of the existing centralized models. Section 6.3 elaborates on the contributions of the proposed distributed framework. Section 6.4 presents the proposed distributed CSP framework including the system architecture and the working economic models. Section 6.5 discusses the decentralized resource allocation approaches, hierarchical and distributed, with the proposed regional IaaS resource allocation model (RCG-IaaS) formulation. Section 6.6 illustrates the simulation environment and performance evaluation. Section 6.7 concludes the chapter.

6.1 Overview

Recently, cloud computing has emerged as a distributed dispersed computing paradigm. Notable examples include interactive applications (e.g. online market 'eBay' and social networks) [191], intensive resources applications (e.g data-intensive search engines [136] and high-performance computing applications [135]), and analytic frameworks that are based on variations of the MapReduce computational model [137]. Obviously, IaaS service is a natural evolution of ongoing work on high-performance scientific and grid computing, which mainly focuses on supporting large-scale scientific applications [192] like climate modeling and high-energy physics [193]. Many computation-intensive applications that have emerged, running across geographically distributed sites, benefit from this distribution potential capabilities, e.g., high-bandwidth sensors applications that produce streaming data [194] like weather radar and video surveillance systems, and business-critical applications such as financial risk calculations in banking [195].

Currently, CSPs are challenged by the lack of resources and the exponential growth of infrastructure demands for hosting new large-scale applications. In addition, the migration of enterprises' applications to the cloud, e.g., business-critical applications, imposes additional challenges and application-specific constraints [196]. Also, the extensive requirements of efficient IaaS in a reasonable time which come at the expense of high monetary costs increases the CSP's responsibility. All these challenges escalate when CSPs seek to extend their coverage and maximize their long-term profit. In that context, many research proposals in the literature have addressed different aspects of these challenges. Several solutions have been achieved that assist CSPs in hosting the arising large-scale applications efficiently. Obviously, resource allocation in cloud computing is a normal evolving stream for the Virtual Network Embedding (VNE) problem, in the sense that integrating the VMs as end-points in the VN constitutes the core of cloud infrastructure services. Solving the VNE problem has been extensively studied in the literature addressing various challenges.

The challenges are principally related to the scalability of the proposed solutions and the quality of the solutions in a reasonable time, i.e., the increased computational complexity.

Houidi *et al.* [197] addressed the challenge concerning the distributed mapping of VNs, i.e., virtual nodes and links to the underlying substrate network, in an efficient manner. The authors designed, implemented and evaluated a distributed mapping algorithm based on a multi-agent framework, where multi-agents ensured the distributed negotiation and synchronization between substrate nodes. However, in their proposal [198], the authors proposed a centralized approach that split the request using Max-Flow Min-Cut based on prices offered by different InPs then decided where to place the partitions. Splitting of the VN request across multiple InPs is solved using both Max-Flow Min-Cut algorithms and Mixed Integer Program (MIP). Unfortunately, these proposals have mainly relied on a centralized approach based on an MIP formulation. Scalability is practically infeasible, particularly if the working environment spans distributed locations with a massive number of managed resources. In addition, splitting VN requests may cause inefficiency and it proves unsuitable for many recent applications. Furthermore, scaling up the agent-based algorithms to work in the order of thousands of substrate nodes has imposed additional communication overheads among agents.

Chowdhury *et al.* [199] proposed a distributed embedding solution called PolyViNE. PolyViNE introduced a distributed protocol that coordinated the VN embedding process across InPs and ensured competitive prices for SPs. Each InP enforced its local policy in allocating resources in its own network before forwarding the unembedded nodes and links to a neighboring InP. A global connectivity has been established to maintain the communication among the participating InPs. The process continued recursively until the whole request was embedded. However, PolyViNE performed a dynamic partitioning for the requests among InPs. Moreover, it performed its operation through a multi-step distributed embedding that imposed more communication overhead for selecting the lowest price at each step. Furthermore, the authors mentioned essential issues in PolyViNE

regarding scalability, response time, and computation overhead.

In the cloud networking context, recent proposals have addressed resource allocation on distributed clouds with different aims and objectives. These proposals provided alternative solutions using different techniques. However, most of the work in that context adopted centralized models over distributed architectures. Zhang *et al.* [200] presented a dynamic service placement solution in a geographically distributed cloud environment that was based on control- and game- theoretic models. Their solution optimized the desired objective dynamically over time according to both demand and resource price fluctuations. However, adopting a control model to perform dynamic placement has been achieved through designing an analysis and prediction module. In that case, the quality of the prediction significantly impacts the optimality of the placements obtained and incurs a high convergence rate. Also, using the game-model-defined Nash equilibrium comes at the expense of the convergence time of the approach. As a consequence, the system response time and customers' satisfaction are negatively affected.

Amokrane *et al.* [169] introduced Virtual Datacenters (VDCs) as an adapted VN with VMs as end-points. The authors then proposed a resource management framework for embedding VDC across geographically distributed datacenters, named Greenhead. The authors achieved their objectives by presenting a two-step approach. The first step is to divide VDC requests into partitions such that it minimizes the Inter-partition bandwidth and maximizes Intra-partition bandwidth. The VDC partitioning problem was solved by adapting Louvain algorithm [201]. The second step is defined by assigning each partition to a datacenter based on electricity prices, datacenters' power usage effectiveness, the availability of renewables, and the carbon footprint per unit of power. Amokrane *et al.* achieved the embedding by designing a greedy algorithm. However, the proposed framework mainly relied on a centralized approach where all the incoming VDC requests are submitted to a central hub for allocation. This approach significantly impacted the scalability of the proposed management framework. In addition, the proposed approach achieved its objec-

tives by enforcing the partitioning of the VDC's requests among dispersed locations for the sake of reducing the energy cost. This partitioning may not be suited to many recent applications. As well as, Papagianni *et al.* [183] followed the centralized model for mapping the Networked Cloud Mapping (NCM) problem. Moreover, they formulated NCM using an MIP formulation, so their solution is practically infeasible due to the enormous growth in managed resources and incoming requests. Even with a relaxed MIP, it is still computation intractable for large numbers.

Kuan-yin *et al.* [202] proposed a cost-aware two-phase meta heuristic algorithm, Cut-and-Search, as a solution for solving the VM placement problem in a geo-distributed cloud; however, it is a centralized approach with an approximation algorithm. Cut-and-Search is an iterative search algorithm that needs a termination condition, otherwise, the number of iterations will be high. Even restricting the number of iterations impacts the accuracy of the search, hence, the efficiency of the obtained solution is inevitably affected.

Obviously, most of this related work had a common feature: a centralized approach to working on distributed architectures. Even distributed solutions are still using a multi-agent framework where there is additional communication overhead among agents, especially in the presence of a huge number of managed resources. Moreover, MIP-based models are still computationally intractable and suffer from scalability issues. However, the continued evolution of distributed infrastructures, in terms of scale and capabilities, demands new efficient solutions for the scalability and computational complexity issues that arise. In this chapter, we aim to provide a promising solution for these issues. Initially, the proposed framework follows distributed models working on distributed architectures rather than centralized models. In our framework, we considered the inherent characteristics of geographically distributed datacenters and the necessity of adapting an efficient distributed resource allocation approach. Moreover, we focused on the computational complexity which arises in the presence of large numbers of managed resources and incoming requests, which current centralized approaches have failed to solve. In that context, we

propose two scalable approaches for the IaaS resource allocation in Geo-datacenters. The proposed approaches cope with the increasing number of managed resources, the expected high volume of IaaS requests with stringent QoS requirements, while minimizing the cost. Moreover, these approaches are quite promising solutions for the scalability issue that exists in centralized solutions and they significantly reduce the computational complexity. Furthermore, the proposed approaches employ efficient economical models in performing resource allocation respecting the customers' QoS requirements. We further substantiate the performance of the scalability approaches by proposing a regional resource allocation model based on CG formulation as a large-scale optimization tool. The proposed model is an integral part in the proposed scalability approaches that guarantees the efficiency of the resulting solutions. We advocate a distributed framework for IaaS provisioning that guarantees affordable, scalable, QoS-assured infrastructure for hosting large-scale applications in Geo-datacenters.

6.2 Centralized Model Challenges

The proliferation of large-scale datacenters in accordance with the growth of recent applications, has posed an interesting challenges for CSPs. On one hand, CSPs want to minimize power consumption to reduce costs, while on the other hand, IaaS customers are expecting their applications' resource needs to be met with lower costs. In addition, existing resource allocation approaches present challenges on the CSP side. These challenges include performing optimal resource allocation for IaaS requests in a reasonable time, utilizing the managed resources efficiently, and guaranteeing the provisioned infrastructures' performance. These challenges escalate with a dramatic growth in IaaS demands associated with stringent QoS requirements.

Classically, most resource allocation objectives are achieved in a centralized manner, often relying on a single node responsible for, e.g., load balancing [203]. Centralized re-

source allocation methods are a dominant part of many of the existing IaaS provisioning systems. These methods have a number of drawbacks that attack the efficiency of any centralized solution [203]. The main drawbacks of these solutions are as follows:

- The environment remains static while the central coordinator is calculating the optimal resource allocation. This prevents the system from responding to any unpredictable changes in the environment or even accepting new requests.
- All coordination messages must route through the central point, counteracting the benefits from having resources distributed over the network. This issue reduces the scalability and creates a fundamentally brittle system [203].

The emergence of the distributed computing paradigm allows the traditional centralized approach to work in these distributed architectures [183] [169]. Although centralized solutions are possible to some extent, they present several inconveniences for large-scale infrastructures spread across different geographical domains. Also, centralized approaches are often not very appropriate to working on such large scales. Moreover, such a centralized model is not adequate and is inefficient in cases where the distributed nature is dominant.

In a mathematical modeling formulation for large-scale resource allocation problems in centralized models, many variables and constraints exist that lead to a huge number of feasible solutions and increase the computational complexity for finding the optimal solution. The computational complexity in centralized controllers impedes the adoption of IaaS, particularly with applications that require QoS guarantees and where latency is critically perceived by customers, e.g., video streaming [191]. Additionally, adopting heuristics algorithms in centralized models working in large-scale environments restricts the problem space in one or more dimensions to enable efficient heuristics, at the expense of the optimality of the solution.

Indeed, centralized approaches for resource allocation in highly-distributed and dynamic environments are impractical and inefficient and this is still an existing challenge. These

drawbacks lead to the need for a truly decentralized resource allocation approach that does not rely on a central controller and that utilizes the distributed architecture efficiently [203]. There is a scalability issue with the computational complexity challenge that exists in the centralized approaches, which arises with large numbers of constraints and variables. This well-known NP-Hard problem motivates this research to propose an efficient decentralized solution that can dynamically adapt to ever-changing conditions of a really large number of requests/resources at a fine-grained level and in a reasonable time.

6.3 Contributions

Two requirements are of particular importance in addressing the aforementioned challenges and guarantee an efficient infrastructure for an enormous number of applications in Geo-datacenters. First, there is a need for a scalable approach that copes with serving a large number of IaaS requests and the ever-increasing managed resources. Secondly, there is a need for an efficient resource allocation technique that guarantees the potential benefits for infrastructure stakeholders in terms of QoS assurance and low response time. Moreover, it should ensure the CSP's benefits of low operational cost and efficient utilization for the managed resources. The desired resource allocation approach should deal with an enormous number of managed resources and IaaS requests efficiently such that it mitigates the computationally intractable limitation. In that context, we advocate a distributed framework that enables CSPs to offer efficient infrastructures for hosting large-scale distributed applications. In contrast to the existing centralized provisioning systems, the proposed framework mainly relies on a distributed architecture for CSPs that manages geographically distributed datacenters (Geo-datacenters) logically grouped into regions. Moreover, the desired architecture incorporates two decentralized resource allocation approaches, hierarchical and distributed. These working approaches employ an efficient economic model in performing their operations. Furthermore, working at the regional level, a mathematical model for the regional IaaS resource allocation problem (RCG-IaaS) is proposed. This

model constitutes the core of the two decentralized approaches and is formulated in CG as a large-scale optimization tool.

The contributions of this work can be summarized as follows:

- We propose a distributed architecture for Geo-datacenters that span the globe arranged in a set of regions, each region managed and controlled by an independent, elected *Regional_Coordinator*. The overall framework is managed by a *Main_CSP* controller that is concerned with defining the geographical boundaries of each region, the number of datacenters, and the average price for each regional resource. Deploying Geo-datacenters in dispersed regions confirms the reliability of the provisioned infrastructure and ensures resource availability in distributed locations. Further, this distributed deployment positively impacts the accepted number of IaaS requests and the efficiency of the provisioned infrastructures.

- We propose two decentralized resource allocation approaches that address the inadequacy and inefficiency of centralized approaches regard scalability and computational complexity issues. Particularly, the large-scale resource allocation problem entails computational complexity that impacts the running applications' performance. The proposed approaches employ economic models as potential solutions for efficient resource allocation. We aimed to reduce the computational time and the backbone network traffic with the *Main_CSP*, as well. The proposed approaches are:

  - A hierarchical resource allocation approach. This approach exploits a **Double-side** auction mechanism that splits the incoming IaaS demands among the *Regional_Coordinators* driven by the CSP's benefit in each region. As a result, the computational complexity is reduced. The **Double-side** auction provides incentives to customers based on such attractive criteria as price and quality to express their willingness to pay for their requested infrastructures.

  - A distributed resource allocation approach that makes use of a **Single-Round Sealed-Bid** auction mechanism as a coordination strategy between *Regional_Coordinators*. This approach stimulates CSP's partners (i.e., *Regional_Coordinators*) to compete for a good QoS service, with a global objective of maximizing the CSP's profit.

- We extend our mathematical model in [3] for a regional IaaS resource allocation approach **(RCG-IaaS)**. The extended model, working through the proposed two decentralization approaches, aims to maximize the CSP's net profit while being acquainted with the available inter/intra-datacenter bandwidth. Additionally, the proposed model respects customer QoS requirements and guarantees the provisioned infrastructure-hosted application's performance is balanced against the infrastructure allocation costs.

6.4 The Proposed Distributed Framework

Several inconveniences have been brought about by adopting centralized resource allocation approaches over distributed architectures. These inconveniences make these approaches inefficient, impractical, and unable to scale well. In this section, we propose an efficient distributed architecture that is based on regions. This architecture allows efficient resource utilization to be performed within the distributed architecture.

6.4.1 System Architecture

In this architecture, we are interested in providing an efficient infrastructure that hosts large-scale applications. These applications serve distributed end-users with stringent QoS requirements. Thus, the main infrastructure stakeholder, the named IaaS customer, owns large-scale distributed applications and is keen to have a reliable infrastructure for hosting these applications. The desired applications are distributed applications that require

VMs on distributed datacenters to host their components. These applications mainly rely on cloud-distributed architecture to achieve their objectives. Hosted applications are categorized into three QoS classes following the Google datasets categories, named gratis, middle, and production [131]. Consequently, the provisioned infrastructures follow the same categorization.

On the other side, a CSP who is responsible for efficiently satisfying customer requests owns the distributed framework that hosts the desired applications. The proposed distributed framework is to be built by deploying multiple datacenters in dispersed geographical locations. These distributed datacenters are arranged in groups forming regions. Defining each region's boundaries is an administrative task carried out by the CSP considering many criteria, e.g., the proximity of these datacenters to each other. Another criteria for grouping and region formation is sharing the same regional hours of daylight and possessing a tiny fluctuation in utility prices. It is worth noting that utility prices differ between regions and they even vary over time in some regions. Moreover, the backbone network link cost is directly proportional to its length, i.e., the geographic distance between the connected datacenter. Thus, it is better to perform partitioning on the backbone network between regions such that reduce the allocation cost. This has been achieved in our proposed design that resulted in the created regions. Furthermore, the distance inside the region during the resource allocation is measured by the number of hops between the region's datacenters. So, minimizing the number of hops reduces the allocation cost as well.

An important challenge in hosting large-scale distributed applications in a Geo-distributed cloud system is to minimize the overall operating cost. This cost is mainly attributed to electricity cost and the wide area network (WAN) communication cost (i.e. backbone network). In this architecture, we grouped the datacenters based on their proximity to each other such that it minimizes resource allocation in the backbone network. We develop a model to capture the intrinsic trade-off between Inter/Intra-datacenter bandwidth al-

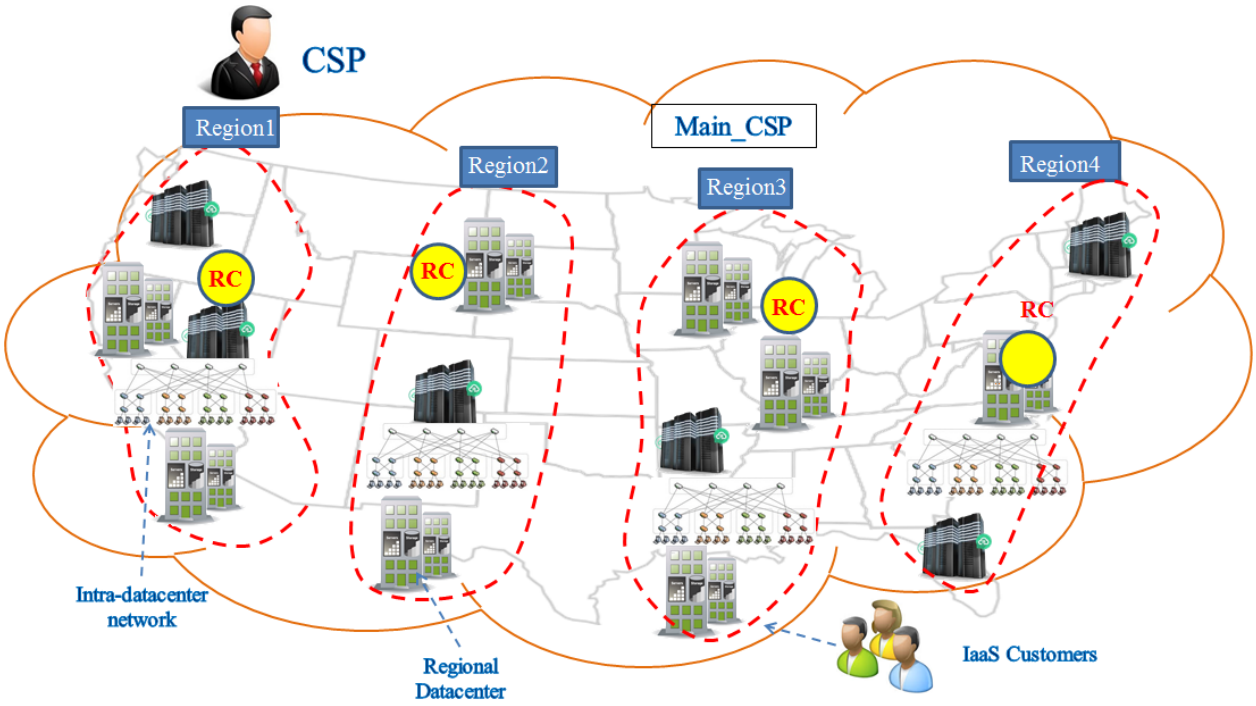


Figure 6.1: Proposed distributed architecture

location costs, and we formulate the regional resource allocation problem considering the impact of this ratio in the design.

The connectivity among distributed datacenters is achieved through a backbone network. The entire infrastructure (including the backbone network) is owned and managed by the same IaaS provider. Therefore, in order to be close to the application's end-users, the hosted applications are served from the same region datacenters. The closeness of the hosted application components within the same region benefits applications like scientific applications, astronomical prediction, and weather forecasting, as well as entertainment applications, where end-users are sensitive to latency, e.g., video streaming (Netflix), social networking, and Google web search. Further, proximity of datacenters to end-users reduces the allocation on the backbone network which is significantly more expensive than Intra-datacenter [204]. Providing infrastructures for hosting applications in a number of regions is beyond the scope of this dissertation.

Figure 6.1 illustrates the architecture of the proposed distributed framework. Hereafter,

we will refer to the infrastructure stakeholder as an IaaS customer who aims to have a cost-efficient and reliable infrastructure for hosting and running his applications. As shown in Figure 6.1, the proposed architecture is made up of a set of datacenters spanning a large area and grouped into regions (e.g., East, West, Central). Two main entities are defined, the central *Main_CSP* and a set of *Regional_Coordinators* (RCs) corresponding to the regions of the entire architecture. Periodically in each region, an election algorithm promotes the highest resource utilization datacenter to be the *Regional_Coordinator* for the region. Obviously, the infrastructure provider will make use of this distributed infrastructure with the objective of maximizing its revenue, improving customer satisfaction, and minimizing the provisioning cost and service response time. Intuitively, distributed customer requests are forwarded to the nearest *Regional_Coordinator* in their region. It is required to allocate the appropriate resources for hosting the customers' applications, respecting their QoS requirements. Based on the solution used, the recipient *Regional_Coordinator* has the choice of whether to allocate the requested resources within its region or forward the received requests to the *Main_CSP* for a decision. The *Main_CSP* is capable of solving any issues that arise from additional constraints such as resource outages or customer location.

In our architecture, the elected *Regional_Coordinator* is defined as an independent entity that manages the entire region datacenters and represents them in upper-level decision-making at the *Main_CSP*. Each datacenter in the region contributes with all its available resources in creating a general resource repository. The elected *Regional_Coordinator* is responsible for managing the created resource pool and performing an efficient resource allocation for the IaaS requests received. The backbone network is divided among the regions, allowing bandwidth to be better managed. This backbone partitioning reduces the allocation cost that is more expensive than for the Intra-datacenter network [204]. One of our main design objectives is to prohibit request splitting among regions (out of scope) in order to reduce allocation costs.

Table 6.1 presents all the notations that will be used in the next proposed approaches.

Table 6.1: Proposed Approaches Notations

Notation	**Meaning**
R_g	Region g in the distributed architecture
D_g	set of datacenters in region g
L_g	set of backbone network links in region g
Q_q	Class q in a set of QoS classes
C_q^g	Capacity of VRs from QoS q in region g
B_a^g	Aggregated Intra-bandwidth in region g
I_n	IaaS request n in a set of IaaS requests
V_g	The generalized virtual resource pool in region g

A region can be modeled as a weighted undirected graph $R_g(D_g, L_g)$ with a set of datacenters and backbone links. Each datacenter D_g in the entire architecture is composed of a set of physical servers interconnected in a tree-like topology (e.g., CLOS, fat-tree) as an internal network topology. Physical servers have adopted virtualization techniques (e.g., partitioning) to form a set of VRs (i.e., VMs) following the predefined QoS classes [131]. The capacity of VRs from QoS class q denoted by C_q is defined as the total number of VRs available at the regional resource pool. Also, the aggregated value for the Intra-datacenter bandwidth denoted by B_a is calculated based on the oversubscription values [205]. Typical datacenter network designs are oversubscribed by a factor of 2.5:1 (400 Mbps) to 8:1 (125 Mbps), i.e., each server in the datacenter is connected in the datacenter network by a 400 Mbps link. Collecting various granularities of VRs from all the regional datacenters constitutes the regional virtual resource pool denoted by V_g. This pool is the general repository for all resources in this region including the Intra-datacenter network and the Inter-datacenter network.

The coordination between the *Main_CSP* and *Regional_Coordinators* in performing IaaS resource allocation takes two forms: vertical and horizontal coordination. In vertical coordination, the communication is carried out between the *Main_CSP* as a central hub and the *Regional_Coordinators*. In addition, a *Regional_Coordinator* is not allowed to communicate with another coordinator. In horizontal coordination, however, the com-

munication is carried out among the *Regional_Coordinators* only, without involving the *Main_CSP*. The role of the *Main_CSP* in the latter form is limited to collecting the reports from *Regional_Coordinators*. Both coordination schemes are adopted in the proposed framework with two applied approaches. These schemes rely on an efficient and economical model.

6.4.2 Economic models in the cloud computing market

The paradigm of cloud computing has spontaneously prompted a wide interest in market-based resource allocation mechanisms by which a CSP aims at efficiently allocating cloud resources among potential users. The task of spreading finite resources across specific customers is also exist in human negotiation and constitutes the basis of modern economics. Adopting economic models is therefore equally well-suited to performing resource allocation in cloud management systems. Economic models are characterized by their scalability and adaptability to the rapid changes in the market conditions. They support an effective decentralized decision making process by providing a well-understood class of protocols. In addition, they also provide incentives for participation in commercial environments. The adoption of economic models in cloud computing is mainly attributed to the widespread adoption of cloud services on a utility basis. This characteristic has created a competitive open-market environment that needs to be balanced, particularly in the case where individual users and providers are acting in a self-interested manner, where the resulting interactions need to be regulated [206]. In addition, observed variances in customers' requirements and regional prices urgently required a regulated trading environment that considers the benefits for both IaaS customers and CSPs [207]. Furthermore, resource bundling that has complicated the traditional resource allocation models has raised an interest in such economic models and, in particular, auction algorithms. In the context of cloud computing, an auction can be explained as the allocation of resources to the highest bidder.

Our proposed framework aims to solve the distributed resource allocation challenges by proposing self-interested entities (i.e. *Regional_Coordinators*) that iteratively trade bundles of VRs in a cloud market controlled by the *Main_CSP*. These entities have an explicit and implicit operation. We adopted efficient economic models, i.e., auction-based mechanisms in performing vertical and horizontal coordination in different applied approaches that organized the information splitting/sharing between the explicit and the implicit operations.

6.4.3 Auction-based Coordination Schemes

Auctions proved their economic efficiency in setting the price of commodities based on supply and demand in real-world markets. Existing auction mechanisms support different negotiations models between sellers and buyers, e.g., one-to-many (for instance, single-sided auction) or many-to-many (e.g., double auction) with the objective to reduce these negotiations to a single value (i.e., price). Moreover, in an auction, participants may be allowed to bid for one commodity or a bundle items at one time. Therefore, the design of the auction mechanism is the hot spot in micro-economics. Combinatorial Auctions (CAs) that allow bids for bundles of items provide a great way of allocating multiple distinguishable items among bidders. CAs can make bidders flexibly reveal their preferences on items, which can decline the bidding risk, increase revenue and thus remarkably improve the economic efficiency of the auction [208]. Auction-based mechanisms have been proposed in various fields such as network bandwidth, wireless spectrum, energy industries and advertising. These mechanisms investigate how participants behave in a competition for resources. Cloud computing appeared to be an effective market-oriented computing paradigm, so currently researchers are interested in investigating the economic aspects of cloud computing from different points of view [208]. As a consequence, because of their ability to establish market prices in a market-oriented computing paradigm, auctions became an efficient means of economic resource allocation in the cloud context [209].

There exist four main types of auction protocol: the Sealed-Bid, Sealed-Bid Second Price (Vickrey), English, and Dutch. Traditionally, The English and Dutch auctions are the common open outcry with ascending price/multiple bid protocol and descending price/single bid protocol, respectively. The Sealed-Bid auction is a sealed single bid following the first price protocol. In the Sealed-Bid auction all bids remain sealed until they are opened simultaneously. Finally, the Vickrey auction is also a sealed-bid protocol, except that the winning participant pays the amount of the second bid as the best price. All the mentioned four auction protocols have the same return in private value auctions; thus selection of an auction protocol depends on specific scenarios.

Double auctions are many-to-many negotiations, that enable multiple participants (i.e. buyers and sellers) to bid simultaneously in one auction. In order to perform successful auctioning, the desired property of an auction mechanism should exist, which is truthfulness. An auction is truthful if the participants benefit most when they reveal their true valuations to the auctioning mechanism [210].

It is worth mentioning that the auction is not entirely new to the cloud computing community. In Amazon, after allocating computing resources for the long-term and on-demand users, Amazon EC2 sells its remaining VMs instances through an auction called Spot Instances [210]. Further, auctions will clearly be one of the most desirable allocation schemes in this regard. The adoption of auctioning is supported by its successful application in various fields ranging from selling wireless spectrum to transportation procurement for large industries [210].

In our architecture, we adopted the auctioning mechanism in the two decentralized approaches as follows:

- A Double-side auction mechanism: this mechanism is applied in the hierarchical approach as a vertical coordination.
- A Single-bid sealed auction mechanism: this mechanism is applied as a coordination

strategy between *Regional_Coordinators* in the distributed approach.

6.5 Decentralized Approaches

The previously mentioned drawbacks of centralized approaches, in particular, scalability and computational complexity, call for a truly decentralized approach for resource allocation. The desired approach does not rely on a central controller, and also it efficiently utilizes the distributed architecture [203]. In this section, we propose two efficient decentralized approaches that can dynamically and efficiently adapt to ever-changing conditions that result from an enormous number of requests/resources at a fine-grained level and in a reasonable time. The proposed approaches are known as, the hierarchical and distributed resource allocation approaches. The hierarchical approach mainly relies on a vertical coordination with dispersed regional resource allocation arranged in two levels, while the distributed approach achieves the horizontal coordination among different controllers at the same level, thus, performing efficient resource allocation independently. We also introduce a *Regional Resource Allocation Model* (RCG-IaaS) that mainly relies on an efficient decomposition technique, Column Generation, as a large-scale optimization tool.

6.5.1 The hierarchical resource allocation approach

In this approach, controllers are arranged hierarchically into two levels as shown in Figure (6.2). The first level accommodates the *Main_CSP* controller, while the second level is made up of all the registered *Regional_Coordinators*. We achieved a local view optimization (within each region) with global view objectives (to maximize the CSP net profit) [211]. The local view optimization is performed by an efficient resource allocation model (RCG-IaaS) that adopts the large VR pool created in each region. The global view objective is achieved by sharing the regional utilization levels and the pricing information with the upper level in the hierarchy using an efficient economic model, a double-side auction.

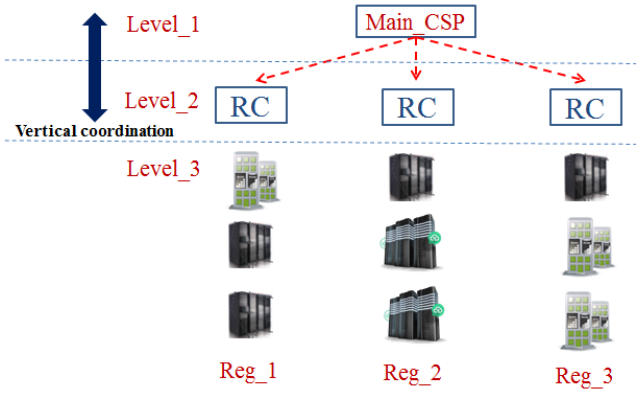


Figure 6.2: Proposed distributed architecture

In order to mitigate the computational complexity challenge, we split the main controller operation that is concerned with resource allocation for incoming IaaS requests into two tasks. The first task distributes the incoming IaaS requests among the regions according to the CSPs' profit calculated at each *Regional_Coordinator*. This task is considered to be a lightweight operation for decision-making with regard to the received request distribution. The second task is the regional resource allocation task that requires heavyweight operations in order to optimize the resource allocation in each region independently. Splitting the computations reduces the resource allocation constraints and variables and allows the centralized controller to respond efficiently and promptly to any further incoming requests. The regional resource allocation task is performed on each *Regional_Coordinator* while the IaaS requests distribution task is accomplished centrally on the *Main_CSP* controller. Coordination between these two tasks is critical in improving the efficiency of resource allocation, particularly in a large-scale environment [212], so, a vertical coordination has been achieved using a double-side auction mechanism between the two hierarchical levels to carry out the overall approach efficiently.

The adoption of a double-side auction is motivated by the fact that participants in this type of auction are negotiating the price in the many-to-many game transparently. Additionally, a double-side auction is similar to a combinatorial auction where both sides

submit bids for multiple items. Thus, it is considerably more efficient than a one-sided combined auction. Furthermore, double-side auctions prevent providers from having monopolies, making them suitable for the cloud computing market [207]. Double-side auction entities are the customers' IaaS requests on one side and the *Regional_Coordinators* on the other side. The *Main_CSP* acts as an auctioneer that handles organizing the double-side auction and maintaining the collected prices and bids from *Regional_Coordinators* and IaaS customers respectively. The *Main_CSP* starts its operation by collecting the bids from both sides and determining the winning IaaS requests and regions. The *Main_CSP* then calculates the best match between the IaaS requests and regions that will achieve the maximum profit for the CSPs.

Our motivation for this approach stems from the fact that, to maximize the profit, we have to reduce the allocation cost on the backbone. Also, reducing the computational complexity at the central controller contributes to our motivation. Since the resource allocation problem incorporates various regional prices and multiple IaaS customers bidding, the double-side auction is well suited for this market-based environment. In this regard, we formulate the problem of accepting and distributing the incoming IaaS requests among *Regional_Coordinators* as a double-side auction problem. The objective of this problem is to assign the accepted IaaS requests to several regions profitably. We assume that the adopted double-side auction mechanism is truthful, that is, the participants reveal their true valuation without exaggerating their bids or misrepresenting their true valuation. This assumption ensures the benefits for both the IaaS customers and the CSP's different *Regional_Coordinators* in balancing the cloud market.

In this approach, each *Regional_Coordinator* receives all the customers' submitted requests in its region and forwards them to the *Main_CSP* for assessment and re-distribution. This forwarding constitutes the only traffic on the backbone network between all the controllers and the *Main_CSP*. Then, the *Main_CSP* starts to execute the double-side auction distribution algorithm such that the aggregated profit is maximized. Algorithm 3 elabo-

rates on the hierarchical resource allocation approach. The operational steps of the proposed hierarchical approach start by sorting the incoming IaaS requests based on the submitted bids in descending order (line 5). Then, the prices offered by *Regional_Coordinators* are sorted in ascending order (line 6). The algorithm proceeds in matching the sorted IaaS requests with the corresponding sorted *Regional_Coordinators*, thereby achieving the maximum profit. For each IaaS request, the algorithm checks the availability of the resources in the corresponding region (line 10), if the region has enough resources, the matching is completed and the request is assigned to this region (lines 11-17). Up to this step, the request is only assigned and tentative cost and profit are calculated, but no resources are yet allocated.

These steps are repeated for all the incoming IaaS requests at the *Main_CSP*. Requests associated with failed allocations are counted as rejected (lines 20-23). The algorithm ends up with an assignment list ($Assign_{n,g}$), indicating the accepted requests and the regions that are candidates for allocation. The *Main_CSP* as an auctioneer starts to announce the matches of winning requests and winning regions, and then to distribute the requests to those regions. Distributing the requests among the regions is motivated by the fact that the CSP aims to maximize its profit by performing a competition among *Regional_Coordinators* using a double-side auction mechanism. Each *Regional_Coordinator* receives its assigned requests and starts to allocate the appropriate resources such that it maximizes its revenue.

It is well known that the hierarchical approach is an intermediate stage between the centralized and the fully distributed approaches, however it is likely centralized with an enhancement in computational complexity. The efficiency of the proposed approach has been increased over the centralized by adopting the double-side auction profit-aware IaaS distribution among registered regions. After finishing the lightweight computational task represented in the double-side auction distribution, the hierarchical approach proceeds to the heavyweight computational task, i.e., resource allocation in each region individually using the RCG-IaaS model. Each *Regional_Coordinator* receives its assigned requests and

Algorithm 3 Hierarchical Resource Allocation Algorithm Pseudo-code

Input: Set of N IaaSRequests, $I_n = (\text{NR}es_n, \text{NC}on_n, \text{B}id_n)$
Input: Set of G *Regional_Coordinators*, $R_g = (C_v^g, B_a^g, P_v^g, P_l^g)$
Output: Assignment of accepted IaaS requests to *Regional_Coordinators*, $\text{A}ssign_{n,g}$
Steps:
Sort N Bids $\text{B}id_1 \geq \text{B}id_2 \geq \text{B}id_3 \geq .... \geq \text{B}id_N$
Sort G Prices $C_v^1 \leq C_v^2 \leq C_v^3 \leq .... \leq C_v^G$
$n \leftarrow 0$
while $(n \leq N)$ **do**
  for each R_g in G **do**
    if $C_v^g - \text{NR}es_n \geq 0$ and $B_a^g - \text{NC}on_n \geq 0$ **then**
      $Accepted++$
      $C_v^g \leftarrow C_v^g - \text{NR}es_n$
      $B_a^g \leftarrow B_a^g - \text{NC}on_n$
      $Assign_{n,g} \leftarrow (n, g)$
      $Allocated \leftarrow true$
      $n \leftarrow n + 1$
      $break$
    end if
  end for
  if $(!Allocated)$ **then**
    $Allocated \leftarrow false$
    $Rejected++$
    $n \leftarrow n + 1$
  end if
end while

starts to allocate the appropriate resources to maximize the CSP's revenue. This hierarchical split reduces the computational complexity and makes the framework more responsive to any sudden fluctuations or further incoming requests, since the busy-time problem has vanished.

6.5.2 The distributed resource allocation approach

The proposal of a distributed resource allocation approach that is aligned with the underlying infrastructure is motivated by the fact that *Regional_Coordinators* need to act independently without the *Main_CSP*. In addition, benefits and advantages brought by the distributed architecture can be strongly affected if the proposed architecture previously employed centralized models. Thus, in this section, we propose a fully distributed resource allocation approach that makes use of an auction mechanism as a coordination strategy. In this approach, a horizontal coordination is achieved among independent *Regional_Coordinators*, with no involvement from the *Main_CSP*. *Regional_Coordinators* make the decisions for their allocations, perform coordination, and finally reallocate again without involving the *Main_CSP*. Using auctions reduces the communications overhead among participants since bidders bid their true value in a single-bid sealed price auction [209].

In the proposed approach, each *Regional_Coordinator* optimizes its allocations as a self-interest entity, yet cooperates to some extent with other *Regional_Coordinators* in order to achieve the shared objective of maximizing the *Main_CSP's* total profit. Figure 6.3 illustrates the distributed approach controller arrangement where all the *Regional_Coordinators* perform horizontal coordination through the backbone network. We argue that the proposed approach increases the acceptance ratio, eliminates the convergence time, prevents the duplication of allocations, and ensures the consistency of the resource allocation process. The proposed distributed resource allocation approach alleviates the drawbacks of

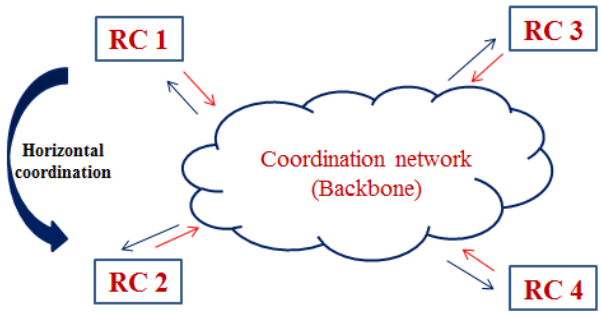


Figure 6.3: Proposed distributed architecture

the centralized approach following a well-known general strategy in mitigating the computational complexity and reducing the traffic on the backbone network. This methodology is partitioning the computation space among a set of predefined entities that are capable of performing the required computations independently.

Popular distributed algorithms are challenged by two main issues, convergence time and the consistency of the shared data. Our proposed approach overcomes these limitations with an efficient auction mechanism. The proposed approach works by using a single-bid sealed price auction mechanism as a transition between two iterations of resource allocation.

The operational steps of the proposed approach are summarized as follows:

1. In each region, perform the first iteration resource allocation by adopting the RCG-IaaS model.
2. Calculate the CSP main profit in each region individually.
3. Execute the single-bid sealed price auction mechanism (for rejected requests) as follows:
   (a) ***If*** (Rejected Requests exist) ***Then***
   Perform Task 1
   (b) ***If*** (Found announcements on the shared area) ***Then***
   Perform Task 2

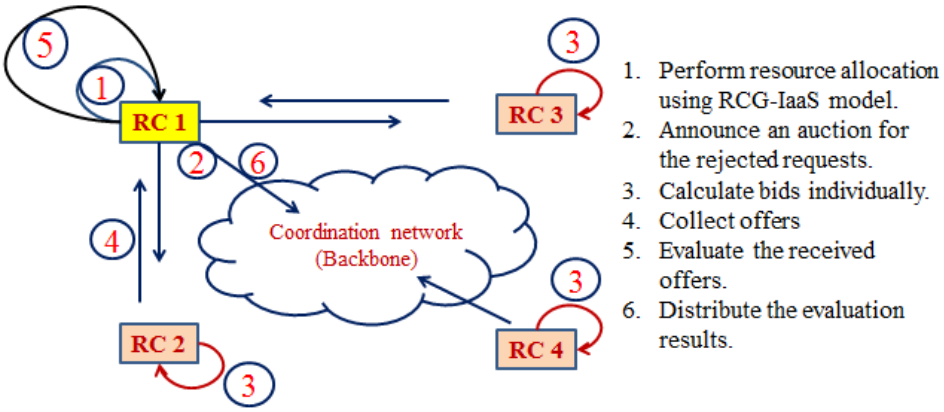
(c) Distribute the task results among the auction participants.

4. Perform a second iteration of the regional resource allocation by adopting the RCG-IaaS model on the forwarded requests from other regions as an auction result.

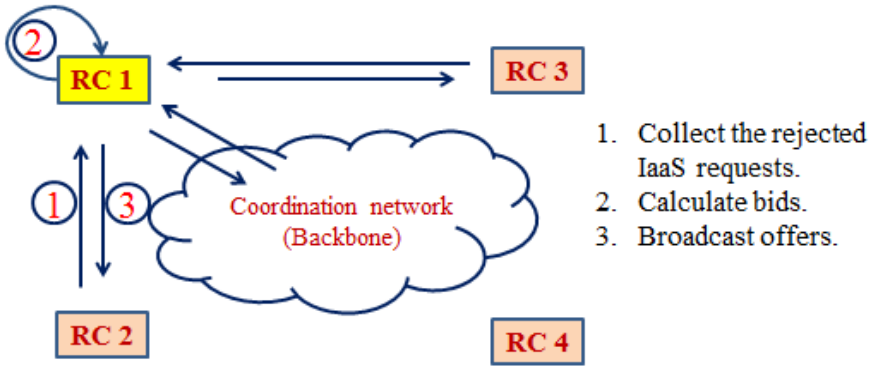
5. Calculate CSP complementary profit at each region.

Following these operational steps, the auction mechanism achieves the transition from the first to the second iteration by adopting a single-bid sealed price auction mechanism among *Regional_Coordinators*. This auction mechanism achieves the desired horizontal coordination. In addition, it is worth mentioning that CSP profit is achieved by accumulating the calculated profit in each iteration. The communication among different *Regional_Coordinators* is carried out through defining a shared area accessed by a unique identity for each *Regional_Coordinator*.

The proposed approach proceeds by performing the first iteration, where each *Regional_Coordinator* collects the IaaS requests received in its region (step 1). Then, each region independently handles allocating the resources for the collected requests employing the RCG-IaaS model and calculating its profit (step 2). After performing the first iteration of resource allocation, each *Regional_Coordinator* checks for rejected requests and announces an auction if a rejection exists by performing task (1) (step 3a). Meanwhile, each *Regional_Coordinator* checks for other participants' announcements (i.e., *Regional_Coordinators*) in the shared area and responds to them by placing its bids by performing task (2) (step 3b). Then, The *Regional_Coordinator* starts to distribute the task results of either task (1) or task (2) to other participants. These task results are represented as the evaluation results for received offers as an output of task(1) and/or placing bids for existing announcements as an output of task(2) (step 3c). The approach then proceeds to the second iteration of resource allocation and final profit calculations (steps 4 and 5).



(a) Sender perspective



(b) Receiver perspective

Figure 6.4: Steps of *RC 1* operations

More illustrations on the operational-step execution order from different perspectives are presented in Figures 6.4a and 6.4b. These figures show a *Regional_Coordinator* operating as an auctioneer or bidder (i.e., sender/receiver). Also, figure 6.5 shows the sequence of the operational-steps execution. Algorithms 4 and 5 illustrate the working tasks within the proposed distributed algorithm, where task (1) is *Announce Auction*, and task (2) is *Joined Announced Auctions*. These tasks work in parallel within two running threads at each *Regional_Coordinator*. The first announces an auction for the rejected requests while the second listens to other participants' auction announcements.

The first task, presented in algorithm 4, is executed immediately after the first iteration of resource allocation when there are rejected requests. In this task, each *Re-*

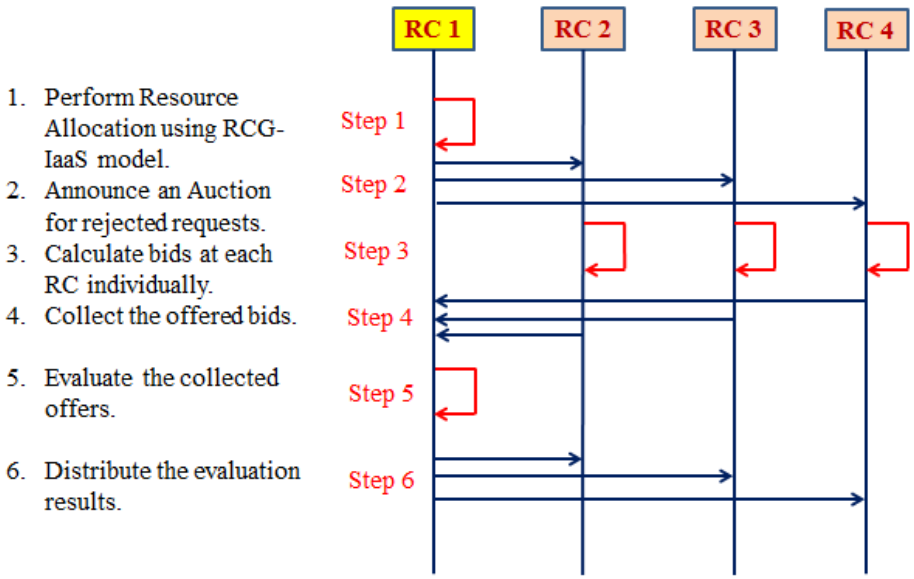


Figure 6.5: Operational-steps execution sequence

gional_Coordinator announces an auction on a shared accessible area and waits for offers from other participants (line 3). Announced *Regional_Coordinators* start to collect other participants' offers (line 4), then merge the collected bids into one list $AllBid_{n,g}$ (line 5). Each *Regional_Coordinator* starts to sort the merged bids for each request in descending order (line 6), and evaluate them. For each request, it selects the most profitable offer such that it maximizes the overall CSP's profit. Meanwhile, bids with zero values are rejected (lines 7-12). A request is counted as rejected if its corresponding bid value in $Assign_{n,g} < 0$ or does not already exist, otherwise, the algorithm proceeds to the next request. Finally, each *Regional_Coordinator* starts to forward its requests to the winning offer regions (line 13).

The second task illustrated in algorithm 5 is a contentious running task that listens for any announced auctions in the shared area (line 3). In each period, the *Regional_Coordinator* collects all the announced rejected requests that exist in the shared area in a REJ_List_g (line 4). Then, it sorts the rejected list in descending order by its original submitted bids (line 5). The *Regional_Coordinator* further starts to check its resource availability for each request, and calculates its new offer regarding that request if it has

Algorithm 4 Task 1: Announce Auction

Input: Set of N Rejected IaaSRequest $I_n = (\text{NR}es_n, \text{NC}on_n, \text{B}id_n)$
Output: Assignment of Rejected I_n to G *Regional_Coordinators*, $\text{A}ssign_{n,g}$
Announce an auction by placing the rejected requests in a shared accessible area.
Collect the incoming bids from each region in $Bid_{n,g}$.
Merge the incoming bids into one list $AllBid_{n,g} \leftarrow Bid_{n,g}$.
Sort $AllBid_{n,g}$ in descending order.
for each (I_n) in (N) **do**
  if $I_n = AllBid_{n,g}$ and $AllBid_{n,g} > 0$ **then**
    Set $Assign_{n,g} \leftarrow 1$
    continue
  end if
end for
Broadcast $Assign_{n,g}$ to all the winning regions.

Algorithm 5 Task 2: Join Announced Auctions

Input: Set of N Rejected IaaSRequest $I_n = (\text{NR}es_n, \text{NC}on_n, \text{B}id_n)$
Output: Offer List $\text{Offer\_L}ist_g$
while *true* **do**
  Collect the announced rejected requests on a $\text{Rej\_L}ist_g$.
  Sort $\text{Rej\_L}ist_g$ in descending order.
  if $size(\text{Rej\_L}ist_g) \geq 0$ **then**
    for each (I_n) in $\text{Rej\_L}ist_g$ **do**
      if $(C_v^g - \text{NR}es_n \geq 0)$ and $(B_{agg}^g - \text{NC}on_n \geq 0)$ **then**
        $C_v^g \leftarrow C_v^g - \text{NR}es_n$
        $B_a^g \leftarrow B_a^g - \text{NC}on_n$
        Calculate Bid_n
        $\text{Offer\_L}ist_g \leftarrow Bid_n$
      else
        $\text{Offer\_L}ist_g \leftarrow 0$
      end if
    end for
  end if
  Broadcast $\text{Offer\_L}ist_g$ to other participants
end while

enough resources, otherwise, it offers 0 (lines 7-16). This evaluation result is saved in the new OFFER_List_g. The *Regional_Coordinator* starts to divide the OFFER_List_g back into regions. Finally, it broadcasts to each region its corresponding portion in its offer list (line 18). After executing the two tasks successfully, each *Regional_Coordinator* receives its new requests. Then, the second iteration of resource allocation starts immediately, where each *Regional_Coordinator* performs the resource allocation model (RCG-IaaS) independently on the forwarded requests.

In this approach, the auction mechanism achieves the desired horizontal coordination. The benefits gained from that algorithm can be summarized by the highest acceptance ratio with the lowest allocation time. Although this approach achieved a gain in profit, it was at the expense of the allocation cost. Also, it eliminates the convergence time of similar distributed algorithms, since the approach has only two iterations. The consistency of allocation is maintained by the auction mechanism that prevents replication and declares only one winner.

6.5.3 Regional Resource Allocation Model (RCG-IaaS)

A fundamental problem in building large-scale resource allocation systems is the need for efficient and scalable techniques that guarantee the expected QoS for customers' applications. The successful operation of the cost-efficient model in [3] motivates us to extend and adapt decomposition methodology and the basic CG model to work on a regional scale with multiple datacenters connected through the backbone. Furthermore, the extended model copes with additional constraints regards the Intra/Inter-datacenter bandwidth and various regions' costs. The goal of the adapted cost-efficient model was changed with the scale. In the private cloud model, the focus is on reducing the IT costs associated with CAPEX and OPEX, while in the context of the public cloud model, the focus is to maximize the provider's profit [213]. The growing scale of the proposed model with different utility prices

in the region stimulates the CSP to define his objective as maximization of his net profit. Populating the cost-efficient model to work on multiple distributed datacenters contributes significantly to the proposed distributed architecture efficiency and the provider's profit.

The inherent characteristic of the CG technique is its decomposition strategy in addressing the resource allocation problem. In that sense, the IaaS resource allocation problem is decomposed into a master problem and a pricing problem. Column Generation also works only with a restricted number of columns, forming the RMP. RMP ensured that Column Generation could manipulate a large number of variables and constraints in a reduced formulation. The pricing problem is solved iteratively as long as the price is significantly reduced; the corresponding column is added to the RMP. In other respects, the current solution is optimal. A column in this formulation represents a feasible allocation of VRs to an IaaS request. The master problem and the pricing problem are linked by transferring the optimal values of the master problem variables to the pricing problem.

Our distributed architecture consists of G regions, each region has D datacenters that constitute the regional pool denoted by V_g. In the following, the presented formulation is for one region, where C_q^d and A_d represents the capacity of VRs from QoS q, $q \in Q$ and the aggregated Intra-bandwidth in datacenter d, $d \in D$, respectively. For the backbone network, B_l represents the residual bandwidth on backbone network link l, $l \in L$, where L represents a set of backbone links that connects all of the region's datacenters.

The proposed Column Generation formulation denoted by (RCG-IaaS) is defined as follows. We reformulate the IaaS resource allocation problem in terms of Independent Infrastructure Allocating Configurations (IIACs), an IIAC indexed by c defines an allocating configuration of one IaaS request. An IIAC can be represented by the couple $IIAC_c = (V_c, P_c)$, where, V_c represents a set of VRs and P_c represents a set of paths used to satisfy customer requirements. Each path P_c is composed of a set of links $l \in L_c$, where $V_c, L_c \subset V_g$. Customers IaaS requests can be represented by $I_n = (V_n, K_n, P_n)$, where V_n represents a set of requested VRs, K_n represents a set of connections between the VRS

with bandwidth b_n, and P_n represents the bid that the customer is willing to pay. We define the following decision variable λ_c, which is equal to 1 if the IIAC c is used in the allocation solution, and 0 otherwise. We aim to maximize the CSP revenue by accepting more IaaS requests and to minimize the cost of the assigned configurations.

Master problem formulation

The resource allocation problem can be formulated with respect to the variable λ_c, $c \in C$. The allocation problem then chooses a maximum of N configurations. In the best case, the model can serve all the received requests, and each request is granted by its IIAC. The choice of the optimal set of IIAC configurations corresponds to the so-called master problem. We define $COST_c$ as the cost of an IIAC c representing the total cost of all the VRs and the paths on that configuration. We define REV_c as the profit gained by serving an IIAC c, to a set of N IaaS requests.

$$REV_c = \sum_{n \in N} a_n^c P_n - COST_c \tag{6.1}$$

Where:

a_n^c represents that IIAC c serves IaaS request n.

P_n represents the offered bid by request n.

The cost of configuration is calculated as follows:

$$COST_c = \sum_{v \in V_c} \zeta_v^c . c_v + \sum_{l \in L_c} b_l . c_l^b + \sum_{l \in L_c} \gamma_b^c . c_l^i \tag{6.2}$$

- ζ_v^c parameter indicates that the VR v is used in configuration c.
- γ_b^c the Intra-bandwidth b used in configuration c.
- b_l is the used Inter-datacenter bandwidth on backbone network link l in configuration c.

- c_v, c_l^b, and c_l^i are the costs of the VR, Inter-, and Intra-datacenter links in configuration c.

The master problem model formulation, denoted by Master-ILP is defined as follows:

- *Objective Function:*

$$Max. \sum_{c \in C} REV_c \lambda_c \tag{6.3}$$

- *Constraints*:

1. The VRs used cannot exceed the total number of available QoS class VRs in datacenter d.

$$\sum_{c \in C} \sum_{v \in V_q} \zeta_v^c . \lambda_c \leqslant C_q^d, \forall q \in Q, \forall d \in D \qquad \left(\delta_q^d\right) \tag{6.4}$$

2. The Intra-datacenter BW used cannot exceed the datacenter aggregated bandwidth.

$$\sum_{c \in C} \gamma_b^c . \lambda_c \leqslant A_d, \forall d \in D \qquad (\phi_d) \tag{6.5}$$

3. The Inter-datacenter BW used for each link b_l cannot exceed the capacity of link l in the backbone network.

$$\sum_{c \in C} \lambda_c . b_l \leqslant B_l, \forall l \in L \qquad (\eta_l) \tag{6.6}$$

4. Unique selection of one IIAC for serving an IaaS request

$$\sum_{c \in C} \lambda_c . a_n^c \leqslant 1, \forall n \in N \qquad (\alpha_n) \tag{6.7}$$

5. Guarantee the maximum number of allocating configurations

$$\sum_{c \in C} \lambda_c \leqslant N \qquad (\beta) \tag{6.8}$$

6. Integrality constraint of master variable λ_c.

$$\lambda_c \in \{0, 1\} \tag{6.9}$$

Pricing problem

The pricing problem is the problem of generating additional IIAC configurations. As previously mentioned, an additional column is added to the RMP at a significantly reduced price that results from solving the pricing problem iteratively. It is defined as follows. Let $\left(\delta_q^d\right), (\phi_d), (\eta_l), (\alpha_n), and\, (\beta)$ be the dual variables associated with constraints (6.4), (6.5), (6.6), (6.7) and (6.8) in the master problem. Then, the reduced cost of variable λ_c can be written as:

$$\overline{REV_c} = REV_c - \sum_{n \in N} \alpha_n . a_n^c - \sum_{l \in L} \eta_l . b_l - \sum_{d \in D} \sum_{q \in Q} \delta_q^d - \sum_{d \in D} \phi_d . \gamma_b^c - \beta \tag{6.10}$$

In order to represent the reduced cost equation in pricing problem variables we define the following decision variables:

z_n: a binary decision variable that indicates if an infrastructure n is granted by configuration $c \in C$ or not.

x_r^v: a binary decision variable that indicates if a requested resource r is allocated to virtual resource v or not.

y_k^p: a binary decision variable that indicates if a connection k uses path p or not.

We derive the following relations between the variables of the pricing problem and the coefficient of the master problem:

For each $c \in C$ and $n \in N$, we have:

$$a_n^c = z_n \tag{6.11}$$

For each link $l \in L$, we have:

$$b_l = \sum_{n \in N} \sum_{k \in K_n} \sum_{p \in P_c} \sum_{l \in L_p} \pi_l^p . b_n . y_k^p \tag{6.12}$$

Where: π_l^p is a parameter that indicates that link l is used in path p. For each $c \in C$ and resource $v \in V_c$, we have:

$$\zeta_v^c = \sum_{n \in N} \sum_{v \in V_c} x_r^v \tag{6.13}$$

For each Intra-datacenter bandwidth b, we have:

$$\gamma_b^c = \sum_{n \in N} \sum_{k \in K_n} \sum_{p \in P_c} \sum_{l \in L_p} \pi_l^p . b_n . y_k^p \tag{6.14}$$

However, Intra-datacenter paths are assumed to be single-hop paths, i.e. $\sum_{l \in L_p} \pi_l^p = 1$ as the oversubscription factor used in designing the datacenters communication networks' implicitly covered allocated bandwidth for multiple links in each path. Equation 6.14 will be simplified to:

$$\gamma_b^c = \sum_{n \in N} \sum_{k \in K_n} \sum_{p \in P_c} b_n . y_k^p \tag{6.15}$$

Accordingly, the pricing problem objective function (Eq.6.10) and constraints can be expressed as follows.

- *Objective:*

$$\begin{aligned} \overline{REV_c} = \sum_{n \in N} z_n . P_n - COST_c - \sum_{n \in N} \alpha_n . z_n - \sum_{l \in L} \eta_l \times \sum_{n \in N} \sum_{k \in K_n} \sum_{p \in P_c} \sum_{l \in l_P} b_n \pi_l^p y_k^p \\ - \sum_{d \in D} \sum_{q \in Q} \delta_q^d - \sum_{d \in D} \phi_d \times \sum_{n \in N} \sum_{k \in K_n} \sum_{p \in P_c} b_n y_k^p - \beta \end{aligned} \tag{6.16}$$

Where :

$$COST_c = \sum_{n \in N} \left[\sum_{r \in R_n} \sum_{v \in V} x_r^v . c_v + \sum_{k \in K_n} \sum_{p \in P_c} \sum_{l \in l_P} b_n . \pi_l^p . y_k^p . c_l \right] \tag{6.17}$$

- *Constraints:*

1. For an accepted request n there is a unique allocation for each requested resource.

$$\sum_{v \in V_q} x_r^v \leqslant z_n, \forall r \in R_n, \forall n \in N, \forall q \in Q \tag{6.18}$$

2. For an accepted request n there is a unique path allocation for each requested connection.

$$\sum_{p \in P} y_k^p \leqslant z_n, \forall k \in K_n, \forall n \in N \tag{6.19}$$

3. Linking the decision variables x and y, connection constraints; at least one allocated path p for each couple of virtual resources $(v\grave{v})$

$$x_r^v . x_{\grave{r}}^{\grave{v}} \geqslant y_k^p, \forall (v, \grave{v}) \in P \times P, \forall (r, \grave{r}) \in K_n \times K_n. \tag{6.20}$$

4. Path length does not exceed ϵ

$$L(p) \leqslant \epsilon, \forall p \in P_n \tag{6.21}$$

5. Ensure that the generated configuration (infrastructure) can handle the allocation of maximum one request.

$$\sum_{n \in N} z_n \leqslant 1 \tag{6.22}$$

6. Linking the decision variables z and y.

$$z_n \geqslant \sum_{p \in P} y_k^p, \forall n \in N, \forall k \in K_n \tag{6.23}$$

Solving the RCG-IaaS Model

To start solving the RCG-IaaS model, we followed the same steps that were used in solving the RA-IaaS-CG model in 5.4.5. However, since the objective function is changed, the RMP has to be initialized by using a set of dummy columns with zero revenue. The upper bounds for ILP problems can be computed through straightforward relaxation. The LP relaxation of *Master-ILP*, denoted by *Master-LPR*, is obtained by relaxing the integrality constraint in Eq. (6.9) to be $\lambda_c \in [0, 1]$. The procedure for solving this model proceeds as follows:

1. Initialize the RMP with a set of dummy columns with zero revenue.
2. Solve the *Master-LPR* using the CPLEX solver.
3. Pass the optimal dual variables to the pricing problem.
4. Optimally solve the pricing problem to find a new column.
   (a) If a column with positive reduced cost has been found, add this column to the *Master-LPR* and re-optimize. Otherwise, *Master-LPR* is optimally solved.
   (b) To calculate the integer solution of *Master-ILP*, we re-establish the integrality constraints again on λ and proceed with a branch-and-bound procedure using the CPLEX solver.

We note that the termination of the *Master-LPR* solution indicates that the optimal solution upper bound that is very tight to the *Master-ILP* has been reached. To derive an integer solution from the fractionary solution of *Master-LPR*, we therefore embed CG

within a branch-and-bound framework. So, re-establishing the integrality constraint again and applying a branch-and-bound procedure to the master problem will provide an upper bound solution for the IaaS allocation problem. Adding additional columns iteratively using the pricing problem operation is necessary to solve the *Master-LPR* at non-root nodes of the branch-and-bound search tree. Otherwise, solving the *Master-LPR* over the existing columns is unlikely to find an optimal or even feasible solution to the original problem.

6.6 Performance Evaluation

This section elaborates on the conducted experiments and evaluations that assess the distributed architecture along with the two decentralized approaches and the *Regional Resource Allocation Model* (RCG-IaaS) in two parts. In the first part, extensive experiments have been carried out to prove the effectiveness of the proposed distributed architecture. Furthermore, the granularity of the regions is defined and the impact of Inter/Intra-datacenter network bandwidth variations have been evaluated. In the second part, the efficiency of the proposed *Regional Resource Allocation Model* (RCG-IaaS) has been proved via simulation against existing heuristic resource allocation solutions on distributed datacenters. Moreover, we defined evaluation scenarios which represent the performance, scalability, and economic benefits obtained from the proposed approaches.

6.6.1 Part 1: Distributed Architecture Design

To better illustrate the effectiveness of the proposed distributed architecture design along with the working approaches, first, we conducted a study on Inter/Intra-datacenter bandwidth impacts on our design and the granularity of the regions in order to define them accurately. This study also helps in defining profitable values for the backbone and aggregated bandwidth. Our objective is to investigate efficient distributed architecture design, where the CSP achieves a maximum profit while respecting customers' IaaS requests. Also,

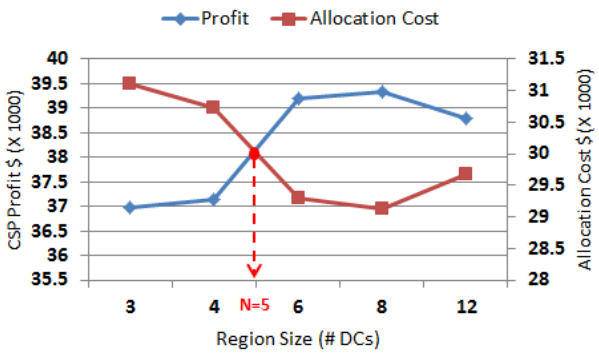


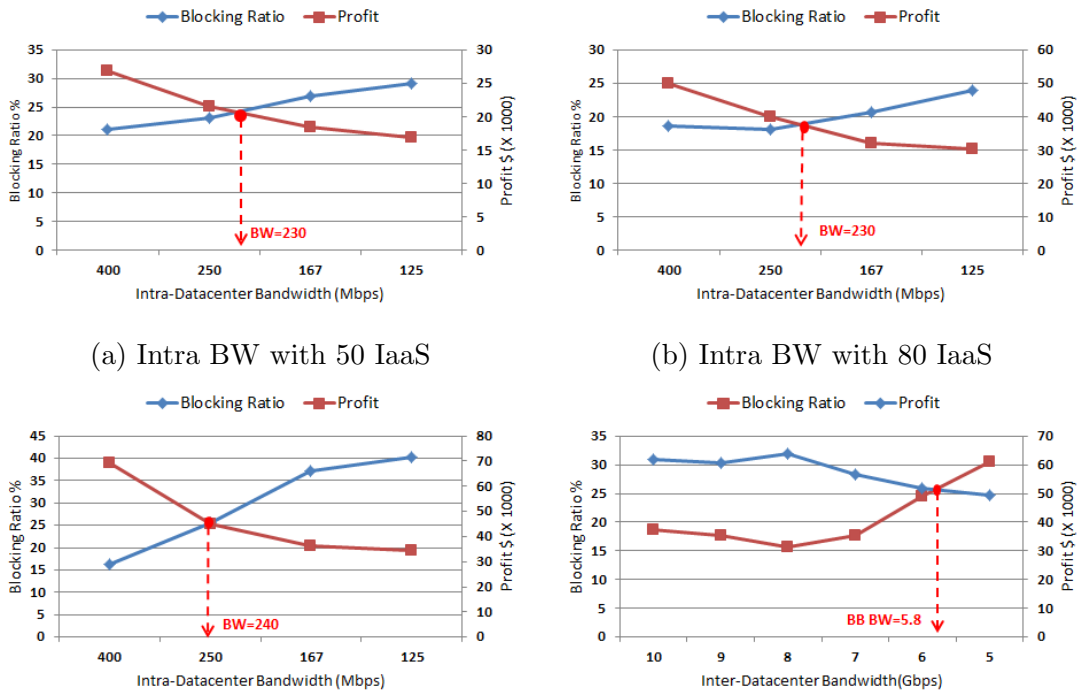
Figure 6.6: Profit-Cost trade-off with variation in region size

we aimed to minimize the network capital and operational expenses through an optimal dimensioning of the Inter/Intra-datacenter bandwidth, thereby preventing extra charges to the CSP for over-allocation on backbone networks and achieving efficient bandwidth utilization.

Figure 6.6 highlights the trade-off between the CSP's profit and the allocation cost. It is observed that, beyond a region size of $5DCs$, the proposed framework achieves maximization in profit and reduction in cost. However, below that size, the tendency is reversed.

We then studied the impact of the Intra-datacenter bandwidth variation on the blocking ratio and CSP profit. Figures 6.7a, 6.7b, and 6.7c present the Profit-Blocking Ratio trade-off with Intra-datacenter variations in the presence of different request loads. We observe that, to keep the CSP profitable, the aggregated bandwidth (Intra-datacenter BW) in the design phase should not go beyond $250Mbps$ (i.e., 4:1 oversubscription ratio).

While investigating the impact of the Inter-datacenter bandwidth variation on the blocking ratio and CSP profit for the design purpose, we discovered that the proposed model performed well serving a large number of IaaS requests with Inter-bandwidth greater than $5Gbps$ as shown in figure 6.7d. Beyond this value, a substantial increase in the cost will occur. Also, we observed that with a small IaaS request load, the model tends to allocate inside the datacenters first. This implies that the Inter-datacenter bandwidth



(a) Intra BW with 50 IaaS

(b) Intra BW with 80 IaaS

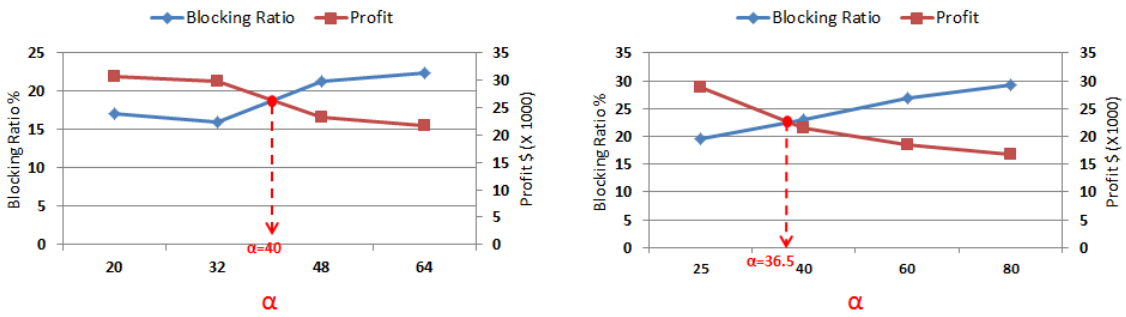
(c) Intra BW with 100 IaaS

(d) Inter BW with 100 IaaS

Figure 6.7: Profit-Blocking trade-off with Inter/Intra-datacenter variations

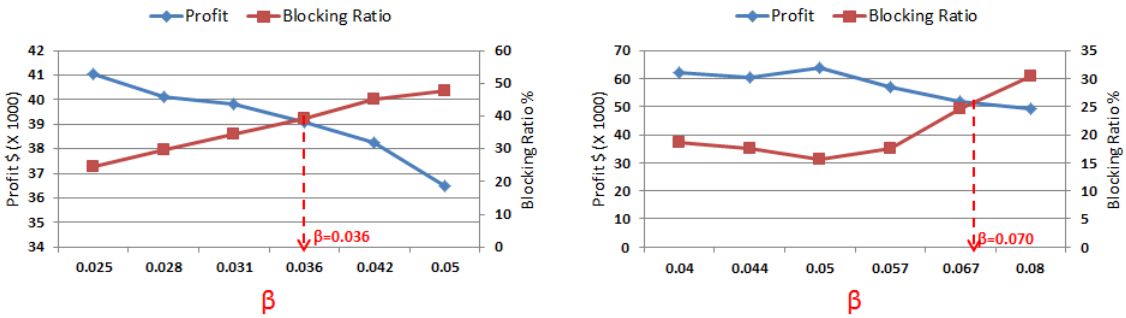
variation slightly impacts our design.

We further define parameters α and β as the Inter/Intra , Intra/Inter respectively, figures 6.8a, 6.8b, 6.9a, and 6.9b representing the impact of changing these ratios on the blocking ratio and the CSP's desired profit. Our objective in defining α and β is to fine-tune the previous bandwidth parameters, i.e., Intra-datacenter and Inter-datacenter bandwidth values. By comparing the α values in figures (6.8a and 6.8b), we observed the intersection point in the curve declaring the profit breakpoint ranges from $\alpha = 40$ to $\alpha = 36.5$. These values imply that the Intra-datacenter ranged between 200 to $273 Mbps$, which confirms our previous observations. For the Inter-datacenter bandwidth, the profit breakpoint ranges from $\beta = 0.036$ to $\beta = 0.070$ which makes the Inter-datacenter bandwidth value range between 5.7 to $6.9 Gbps$. This is also confirmed our previous design value for the Inter-datacenter bandwidth ($> 5 Gbps$ in figure (6.7d)).



(a) α ratio 1 (b) α ratio 2

Figure 6.8: Inter/Intra-datacenter BW variations impact



(a) β ratio 1 (b) β ratio 1

Figure 6.9: Intra/Inter-datacenter BW variations impact

6.6.2 Part 2: Evaluation Scenarios

In the following, the setting of the conducted simulation scenarios is given, along with a description of benchmarks used, the performance metrics evaluated, as well as the results obtained. The experimental assessments were carried out using the IBM CPLEX solver [190]. Furthermore, we exploited the *Gap* calculation to evaluate the optimality of the obtained solutions. Since the obtained solutions have $Gap \neq 0$, the obtained results can be expressed as near optimal with acceptable *Gap*.

Simulation Scenario Settings

In order to assess the efficiency of our hierarchical/distributed resource allocation approaches along with the proposed *Regional Resource Allocation Model* RCG-IaaS, three evaluation scenarios were defined. The defined scenarios concerned the performance evaluation, scalability evaluation, and economic evaluation. The performance scenario evaluates the quality of the obtained solution from the proposed RCG-IaaS model against existing heuristics solutions, while the scalability scenario evaluates the proposed decentralized approach's scalability against the RCG-IaaS model as a large-scale centralized approach with a variable IaaS requests load. The economic evaluation scenario evaluates the CSPs' net profits in the various solutions.

Adopting the design parameters obtained from our investigation in 6.6.1, we consider our simulation environment to be 12 datacenters in four regions located in the U.S.A., in the East, West, and Center. Each region has 3 datacenters, each made up of a random number of VMs with different granularities according to the predefined QoS classes: gratis, middle, and production [131]. We adopted NSFNet network topology [214], with a capacity of $10Gbps$, as a backbone network connecting the Geo-datacenters. Furthermore, aggregated values for the Intra-datacenter network bandwidth are applied according to different oversubscription values as defined in [205]. Meanwhile, the IaaS requests are generated randomly following a uniform distribution covering the three defined QoS classes. The number of VMs per request is also uniformly distributed.

In order to assess the performance of the proposed model under a bulk number of IaaS requests, the periodical approach in [188] was followed in serving IaaS requests. Simulation planning time is divided into a set of periods $p \in P$. The period length is variable and defined based on a random number of the received requests collected in small-batches. Meanwhile, a random number of IaaS requests leave the system.

Benchmarks

To better illustrate the efficiency of the RCG-IaaS model regarding the quality of the provided solution, the performance evaluation is carried out against the following benchmarks:

- Network Aware Resource Allocation in Distributed Clouds (NA-RA) [215]: Proposed an efficient resource allocation algorithm for use in distributed clouds based on formulating the allocation problem as a subgraph selection problem. The defined formulation has been approximated to an efficient 2-ϵ approximation algorithm for the optimal selection of datacenters in the distributed cloud.

- Stable Resource Allocation in Geographically Distributed Clouds (SEA) [216]: Proposed a heuristic algorithm based on an adaption to the Stability Marriage problem. To better understand the structure of the SEA problem, the authors first consider $1-D$ SEA in which only the CPU resource requirement is taken into account. Then, they considered $2-D$ SEA by involving the network requirement and incorporating virtual network embedding techniques into their solution.

In order to evaluate the scalability performance of the proposed model, we define our baseline approach as follows:

- Baseline approach: we adopted the *Regional Resource Allocation Model* as a centralized baseline approach for evaluating the efficiencies of the hierarchical and distributed approaches. The adopted baseline defined a centralized resource allocation model that allocates the incoming IaaS requests on 12 managed datacenters including the backbone network bandwidth allocation.

Performance Evaluation Metrics

In this paragraph, we conduct experiments to evaluate the performance of the proposed RCG-IaaS model, and we compare them with the performance of solutions from related

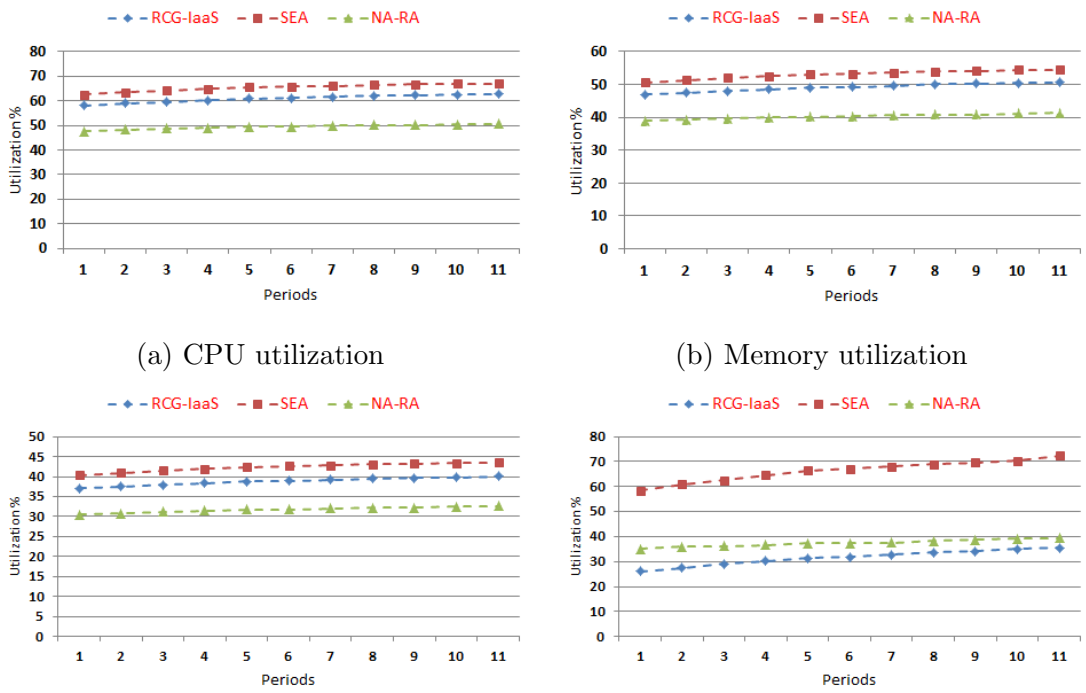
work. To quantify the performance of the RCG-IaaS model, the following performance metrics are measured:

- Acceptance ratio: calculated as the ratio of accepted IaaS requests to their total submitted requests at each period.
- Datacenter's resource utilization: measured as the ratio of the used resources (CPU, memory, storage, and bandwidth) to their total capacity in the datacenter.
- Provider's net profit: measured based on the accumulated profit of each *Regional_Coordinator* in each region. Each region's profit is calculated based on the bids collected from the successfully allocated IaaS requests, less the cost of the network resources and the VMs calculated by the model. It is worth mentioning that the cost of allocating resources in the backbone network is more expensive than allocating network resources on the Intra-datacenter bandwidth.

We evaluate the scalability of the proposed architecture under the scalability scenario in two dimensions. The first dimension is increasing IaaS requests, and the second dimension is increasing the region size (number of datacenters in each region). The following performance metrics were used:

- Resource allocation execution time denoted by ($T_e(sec)$): measured as the consumed time to end the resource allocation process for the planning period.
- Resource Allocation cost ($Cost(\$)$): the cost of the obtained solution is calculated based on the unit price of the allocated resources.

The results are obtained over many simulation experiments for each scenario. Then we calculate the average value of the performance metrics. We follow the periodical approach [188] in (generating/dropping) IaaS requests at the boundary of each period. Figures 6.10a,6.10b, 6.10c, 6.11a, and 6.11b as expected show that the proposed RCG-IaaS model



(a) CPU utilization

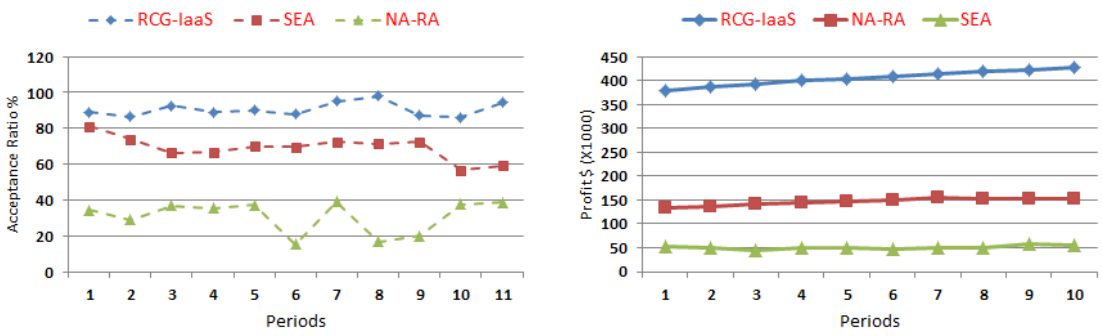
(b) Memory utilization

(c) Storage utilization

(d) Backbone BW utilization

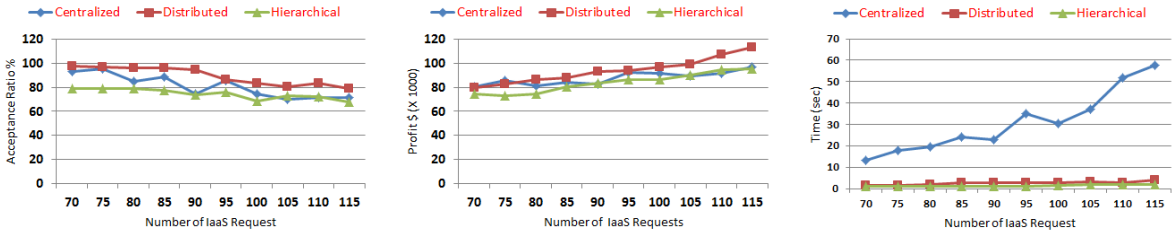
Figure 6.10: RCG-IaaS model resource utilization

outperforms related work solutions (NA-RA, SEA) in resources utilization. The RCG-IaaS model is characterized by adopting Column Generation as an efficient large-scale optimization tool. In addition, the proposed model formulation considered the Inter-datacenter allocation cost versus Intra-datacenter cost with the objective of maximizing the CSP's revenue. Formulating the model using Column Generation iteratively captures the intrinsic trade-off between Inter and Intra-datacenter bandwidth cost. Efficient results obtained in backbone network utilization as shown in figure 6.10d, showed that RCG-IaaS used Intra-bandwidth in serving the incoming requests first. In case of a resource outage (i.e., Intra-bandwidth), the model extends to perform the allocation using the Inter-datacenter bandwidth. We can observe that, NA-RA performance values are close to the values obtained by our model. NA-RA shares the same objective, to minimize the allocation in the backbone network using graphs. However, SEA performs worse than our approach, mainly because, while the Stable Marriage Problem is stable in resource allocation, it



(a) Acceptance ratio (b) RC's profit

Figure 6.11: RCG-IaaS model performance metrics



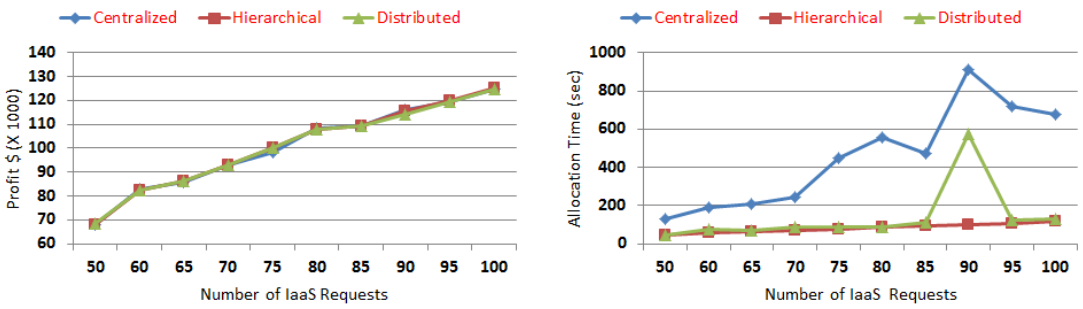
(a) Acceptance ratio (b) CSP's Profit (c) Allocation time

Figure 6.12: Blocking performance with increasing number of IaaS requests

forces partitioning of the request and leads to lower profit for the CSP.

In order to evaluate the economic benefits, we conducted experiments that measure the CSP's net profit, acceptance ratio, and the time consumed for allocating IaaS requests. Figures 6.11a and 6.11b show the behavior of the three resource allocation solutions we are considering (RCG-IaaS, SEA, and NA-RA) in terms of allocation time and net profit for CSPs as a function of increasing the IaaS requests.

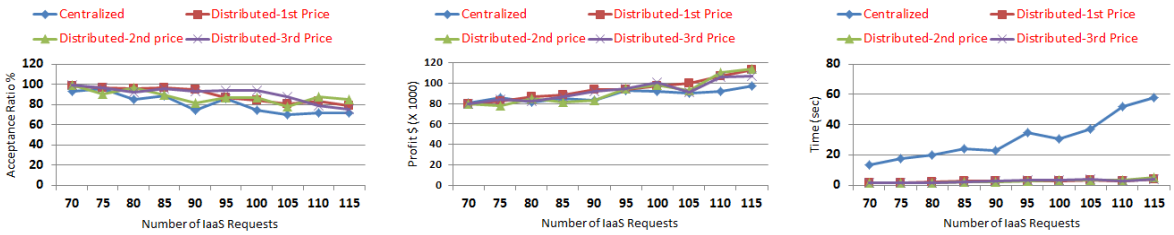
We performed two types of experiments to evaluate the efficiency of the proposed decentralized approaches. The first set of experiments evaluates the performance of the designed algorithms in terms of CSP's net profit and allocation time in empty datacenters (i.e., with non-blocking conditions). The result shown in figure 6.13a presents the efficiency of the designed algorithms in performing efficient results close to the optimal centralized approach with an average deviation beyond 1% in CSP profit. The second set of experiments is con-



(a) CSP's profit (b) Allocation time

Figure 6.13: Non-Blocking performance with increasing number of IaaS requests

ducted in busy datacenters (i.e., with blocking conditions). These experiments evaluate the applicability of auction models in the decentralized approaches. The results shown in figures (6.12a and 6.12b) proved their applicability with deviation in profit of less than 3% from the centralized approach. Indeed, both sets of experiments confirmed the superiority of the decentralized approaches over the centralized approach in allocation time, as shown in figures (6.12c and 6.13b). It is worth mentioning that the distributed approach outperformed other approaches since it has another chance for allocation. This additional opportunity increases the acceptance ratio and the profit, yet with slightly higher cost. Also, despite the global view of allocation in the centralized approach, the hierarchical approach performs very closely to the centralized approach in significantly less time. This observation implies the efficiency of using auction mechanisms.



(a) Acceptance ratio (b) CSP's Profit (c) Allocation time

Figure 6.14: Different distribution policies performance evaluations

Extensive experiments have been carried out on the distributed approach in order to assess the impacts of changing the distribution policy. The predefined distribution policy

was selecting the highest offer in the transition between first and second iterations in algorithm 4. However, in these experiments we select the 2nd and 3rd offers instead of the 1st offer. It is observed that a slight enhancement occurred in the acceptance ratio and the CSP's profit when we did so, as shown in figures (6.14a and 6.14b).But all three selected offers significantly reduced the allocation time as shown in figure (6.14c).

6.7 Conclusion

This chapter tackled the problem of IaaS provisioning scalability and the increased computational complexity in Geo-Datacenters. In that context, we first presented an efficient distributed framework for IaaS provisioning in Geo-datacenters, where various domains' applications can be hosted. The presented framework mainly relies on a distributed architecture that groups Geo-datacenters into regions based on their proximities. Working through the presented architecture, there are two decentralized approaches for resource allocation. We further formulate the regional IaaS resource allocation optimization problem using Column Generation formulation as a large-scale optimization technique (RCG-IaaS) with the objective of maximizing the CSP's net profit while respecting the customer's QoS requirements. We also considered additional regional constraints concerning the distributed architecture. In order to validate the proposed framework with its components, we conducted studies to measure the Inter/Intra-datacenter bandwidth impacts on the desired architectural design and its scalability readiness. Through our simulations, we demonstrated how the computational complexity can be reduced by employing efficient economic models. We also showed the significant contribution of auctioning mechanisms in CSPs' revenue. Experimental results have shown the achieved reduction in allocation cost and computational time of the presented architecture. Consequently, a significant increase in CSP's profit has been observed. To sum up, employing efficient economic models, i.e., auction mechanisms, coupled with a large-scale optimization tool presented an effective decentralized solution for current IaaS provisioning issues. Also, this combination

contributed to achieving efficient utilization for the managed datacenters.

Chapter 7

Conclusion and Future Work

This chapter outlines the contributions of this research work and discusses plans and directions for future work. The chapter is organized as follows: Section 7.1 summarizes the research contributions in the area of Cloud IaaS resource management. Section 7.2 illustrates the limitations of the current research and sheds light on future research directions. Finally, Section 7.3 provides some concluding remarks.

7.1 Conducted Research Work

The focus of the conducted research, thus far, has been the development of a cloud IaaS resource management system that considers both the SPs' and customers' objectives. The first step towards achieving this goal included a study in the literature of the challenges and problems associated with similar existing systems. Based on the identified limitations and drawbacks of similar existing systems, a framework for cloud IaaS resource management was designed. The framework has been presented as layered architecture that overcomes the drawbacks and limitations of similar existing systems. The framework presented here introduced a complete breakdown of CSP framework components that incorporate the advantages of using cloud computing key enabling technologies in a layered and modular architecture. The architecture achieved the design requirements of Flexibility, Efficient and Unified Management, and Scalability. The main characteristic of the proposed architecture is the existence of the VRP that is the result of a unified representation for all the diverse resources collected in a generalized repository. The proposed architecture is also characterized by the unified management of network and cloud resources. The associated

management cross-layer is incorporated into the cloud service life cycle, accepting customer requests, managing and orchestrating different components in the architecture, monitoring resources, and maintaining the QoS of the provisioned services. A set of resource allocation approaches has been evaluated, working through the designed framework. These approaches proved the effectiveness of the proposed framework and the efficiency of their applied methodologies. In the following section, a summary of the main contributions of the current research work is given:

- **Two-Phase Ontology-based Resource Allocation Approach for IaaS Cloud Service (2P-IaaS)**

  A resource allocation framework that used composition as a technique was presented. The presented approach achieved flexibility in service provisioning through several steps: (1) customizing user infrastructure requests without exposing greater detail, (2) handling incomplete topological information for unskilled users, (3) unifying resource representation in an ontology-based model, (4) exploiting the benefits associated with the ontology modeling, using reasoning capabilities in discovering the appropriate resources, and (5) orchestrating a set of techniques that manipulates the designed ontology model in achieving efficient resource allocation. The composition approach presented here relies on a unified ontology model for a VR; this model is populated to form the VRP as a general repository that hosts diverse and heterogeneous resources. The composition approach worked into two-phases (2P-IaaS) as follows: Phase-1: the mapping of hosting resources, and Phase-2: the connectivity composition phase. Furthermore, the presented approach employs semantic similarity, closeness centrality and biased random walk techniques for efficient resource allocation in a reasonable time. The applicability of using the composition concept in resource allocation has been proven by adopting the ontology model with its associated resource manipulation techniques.

- **MILP-based Approach for Efficient Cloud IaaS Resource Allocation (MILP-2P-IaaS)**

  The presented heuristic approach (2P-IaaS) is only adequate for an approximate near-optimal solution in reasonable computational time. In this contribution, a mathematical formulation for the IaaS resource allocation problem was introduced in Mixed Integer Linear Programming (MILP) for improving a datacenter's resource utilization. The presented MILP model considered the heterogeneity of resources while keeping the simplicity of the composition methodology. The presented approach adopted the unified cloud resource representation model and built a generalized resources repository. Also, in this approach, a combined controller is defined to manipulate infrastructure resources collected in the repository. A joint optimization model that performs the resource allocation as the main controller operation was also presented. This model represents the integration of the semantic similarity and closeness centrality concepts and is formulated on a two-phase Mixed Integer Linear Programming (MILP-2P-IaaS): (i) mapping of hosting resources, and (ii) connectivity composition. The presented model is mainly characterized by integrating semantic similarity and closeness centrality into a multi-objective MILP model. This model was solved by the ϵ-constraints methodology. The presented MILP-2P-IaaS model guarantees optimal solutions for the resource allocation problem.

- **A Cost-Efficient QoS-Aware Model for Cloud IaaS Resource Allocation in Large Datacenters (RA-IaaS-CG)**

  The growth in the scale of datacenters and an increasing number of IaaS requests has clearly shown a scalability issue in MILP models. Solving MILP-2P-IaaS is known to be NP-Hard, and thus, large-scale instances and models are often computationally intractable. To overcome the scalability issue and provide an efficient solution in reasonable time, a cost-efficient model acquainted with QoS requirements is presented. This model makes use of large-scale optimization tools and introduces a Column Gen-

eration formulation for IaaS resource allocation in large datacenters (RA-IaaS-CG). The presented cost-efficient model scales well in large datacenters performing an efficient CSP cost reduction while respecting the customers' QoS requirements. Also, it provides an optimal or near-optimal solution for the datacenter resource allocation problem as well as handling the scalability issue.

- **Decentralized framework for Cloud IaaS Resource allocation in Geo-datacenters**

  Despite the benefits that large datacenters bring to CSPs, they are still economically inefficient. As a result, geographically distributed datacenters (Geo-datacenters) have appeared as an economical alternative to large datacenters. In addition, the recent emergence of this distributed computing paradigm offers a compelling solution for better application performance, scalability, and reliability. Moreover, this evolving paradigm stimulates the rapid development of a wide range of prevalent large-scale distributed applications. However, there exist some challenges that impact the satisfaction of distributed end users. These challenges are mainly attributed to the inefficiencies of the hosting infrastructures provisioned by CSPs, in particular, the lack of a QoS guarantee. Unfortunately, CSPs adopt traditional approaches to coping with ever-increasing infrastructure demands and expanded datacenters. Such approaches have caused CSP's operations to become increasingly complex and time-consuming when their managed scale grows. Consequently, performance degrades in the provisioned infrastructures and in the hosted applications with high computational complexity.

  In this contribution, scalability and computational complexity challenges were addressed by introducing a distributed framework that allows CSPs to offer efficient infrastructures for hosting large-scale distributed applications in geo-datacenters. Furthermore, the presented framework incorporates two decentralized resource allocation approaches: hierarchical and distributed, where efficient economic models are used to perform efficient IaaS provisioning. Integrated into these approaches is an adapted

regional IaaS resource allocation model (RCG-IaaS) based on a Column Generation formulation. This model constitutes the core component in the two-decentralized approaches. The presented distributed framework has been verified by conducting a study on the Inter/Intra-datacenter bandwidth impacts on the architecture design. The framework design, along with proposed decentralized-based approaches, significantly contributes to the allocation time and the CSP's profit. The adapted model RCG-IaaS was confirmed to be efficient for performing resource allocation and also readily scalable in distributed datacenters. Furthermore, RCG-IaaS performed an efficient CSPs cost reduction while respecting the customers' QoS requirements.

7.2 Limitations and Future Research Work

7.2.1 Limitations

As we have illustrated, the focus of the conducted research has been the development of an IaaS management framework that assists CSPs in managing their datacenters efficiently. Throughout this research, a set of resource allocation solutions have been presented. However, there are limitations to the current research work, particularly the following:

- ***Adopting Ontology-based Modeling***

  Adopting ontology-based modeling in the composition algorithm impacts the quality of the proposed solution because of the growing scale in the search space. Indeed, in a cloud computing environment, the ever-growing scale in the search space highlights the inefficiencies of the ontology-based modeling, leading to a noticeable decrease in the performance and the acceptance ratio of the proposed approach with an increasing number of IaaS requests, even though 2P-IaaS achieved a high acceptance ratio among benchmarks. This observation shows the limited applicability of the proposed approach to only small-scale datacenters with small number of requests and indi-

cates a scalability challenge in the presented composition algorithm. Nevertheless, we plan to continue our research work to enhance the performance of our presented resource discovery and composition mechanism. Our goal is to improve the efficiency of the presented approach while maintaining the benefits gained from applying the composition technique. This can be achieved by adopting techniques that manage the growing resource scale efficiently instead of using ontology-based modeling. In addition, we plan to overcome the performance degradation issues which occurred due to placing multiple VMs on a single physical machine, i.e., the VM consolidation problem.

- ***Static Resource Allocation***

  The proposed solutions and models in this dissertation use static resource allocation model where the generated pool is a preparatory step prior performing the allocation. Also, the designed algorithms do not handle any dynamic issues while performing resource allocation. This is impractical in the sense that the public cloud environment is dynamic and frequently changeable. The growing dynamicity in the working environment, particularly the emergence of mobile cloud computing, indicates a necessity for dynamic resource allocation systems that cope with the increasing number of customers, the frequent mobility of customers, and the rapid fluctuations in IaaS requests. The dynamic IaaS resource allocation system involves dynamicity in generating the virtual resource pool and allocating the resources, as well. As a requirement for dynamic resource allocation, a prediction module needs to be designed that is concerned with the historical workload modeling and prediction of incoming requests for resources.

- ***Adopting a Static Pricing Approach***

  The presented cost-efficient model (RA-IaaS-CG) and regional IaaS resource allocation model (RCG-IaaS) adopt a static pricing approach that is an unrealistic issue.

This static approach is economically impractical in a cloud market with multiple CSPs and a growing number of customers, in the sense that customers like to negotiate on prices, aiming to have low prices. Furthermore, there is no guarantee preventing CSPs from having monopolies. In addition, there is no guarantee that participants (e.g., customers and CSPs) will reveal their true valuation without misrepresenting it or exaggerating their bids. Thus, we plan to adopt a dynamic pricing approach instead of a static approach. The Vickrey-Clarke-Groves (VCG) mechanism can be used in this case to prevent CSPs from having monopolies and to force them to reveal their values truthfully. On the other hand, VCG can be used to penalize excessive customers if a drop occurs in the running infrastructure performance, thereby maximizing the utility function for all participants.

7.2.2 Future Research

In addition to overcoming the aforementioned limitations, there are also some interesting directions for our future work, the main one being to investigate the applicability of the proposed framework for hosting the most dominant applications, e.g., multimedia applications. These types of applications dictate more QoS parameters that should be taken in consideration. We will further explore how our solution can be extended to host mobile cloud applications and perform efficiently in such dynamic environment. This investigation can be carried out by implementing the proposed framework and the working solutions on a real established test-bed.

Another future direction is performing QoS/QoE-assurance for customers, a monitoring module integrated with an adaptive re-configuration module needs to be designed such that it continuously monitors the physical/virtual resources status and detects any performance degradation that may arise from the implemented resource allocation models and algorithms. In addition, this module is responsible for maintaining the provisioned infras-

tructure QoS and responding immediately to the unpredictable changes that may occurs. Integration of a proactive fault-tolerant module that adopts efficient learning techniques that works in dynamic environment, can guarantee the provisioned QoS and assure better customer QoE. It is important to pay attention to the customers' QoE, particularly for the growing multimedia and gaming applications that cause customers to experience delay.

7.3 Concluding Remarks

Resource management in cloud computing is a complex problem comprising a large number of challenging issues. Resource allocation is a crucial part of datacenter management problems. In addition, the evolution of recent applications and recipient devices at a rapid pace adds more challenges to this field. Allocating limited resources with constraints to customers is an NP-Hard problem which escalates with the presence of an increasing number of requests and managed resources. Furthermore, external constraints regards CAPEX and OPEX make the problem intractable. Existence of diverse and heterogeneous resources, with multiple CSPs in competition with each other, makes the environment hard to control and manage. In addition, customer assurance regarding the quality of the provisioned services and the cost paid makes the problem complicated. This dissertation presented comprehensive and effective improvements on different facets of resource allocation, by presenting a set of working solutions with different objectives at different scales. In addition, this research presented a foundation for many applied resource allocation approaches and will facilitate the evolution of resource management and allocation for the next generation of environments and provide enhanced satisfaction for both CSPs and customers.

References

[1] K. A. Scarfone, M. P. Souppaya, and P. Hoffman, "SP 800-125. Guide to Security for Full Virtualization Technologies," Gaithersburg, MD, United States, Tech. Rep., 2011.

[2] A. Belbekkouche, M. M. Hasan, and A. Karmouch, "Resource Discovery and Allocation in Network Virtualization," *Communications Surveys Tutorials, IEEE*, vol. 14, no. 4, pp. 1114–1128, Fourth 2012.

[3] 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 *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, ser. SIGCOMM '09. New York, NY, USA: ACM, 2009, pp. 51–62.

[4] M. F. Bari, R. Boutaba, P. R. R. Esteves, L. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. Zhani, "Data Center Network Virtualization: A Survey." *IEEE Communications Surveys and Tutorials*, vol. 15, no. 2, pp. 909–928, 2013.

[5] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "SecondNet: a data center network virtualization architecture with bandwidth guarantees." in *CoNEXT*, J. C. de Oliveira, M. Ott, T. G. Griffin, and M. Médard, Eds. ACM, 2010, p. 15.

[6] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards Predictable Datacenter Networks," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM '11. New York, NY, USA: ACM, 2011, pp. 242–253.

[7] "Apache CloudStack," https://cloudstack.apache.org/, Last visited: July 2015.

[8] X. Wen, G. Gu, Q. Li, Y. Gao, and X. Zhang, "Comparison of open-source cloud management platforms: OpenStack and OpenNebula," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, May 2012, pp. 2457–2461.

[9] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-Source Cloud-Computing System," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, ser. CCGRID '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 124–131.

[10] "OpenNebula Home Page," http://www.opennebula.org/, Last visited: July 2015.

[11] M. Ghijsen, J. Van der Ham, P. Grosso, and C. De Laat, "Towards an Infrastructure Description Language for Modeling Computing Infrastructures," in *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, July 2012, pp. 207–214.

[12] I. Fajjari, M. Ayari, and G. Pujolle, "VN-SLA: A Virtual Network Specification Schema for Virtual Network Provisioning," in *Networks (ICN), 2010 Ninth International Conference on*, April 2010, pp. 337–342.

[13] V. Issarny, N. Georgantas, S. Hachem, A. Zarras, P. Vassiliadis, M. Autili, M. A. Gerosa, and A. B. Hamida, "Service-Oriented Middleware for the Future Internet: State of the Art and Research Directions," *Journal of Internet Services and Applications (JISA)*, pp. 1–23, Jun. 2011.

[14] V. Tsiatsis, A. Gluhak, T. Bauge, F. Montagut, J. Bernat, M. Bauer, C. Villalonga, P. Barnaghi, and S. Krco, *The SENSEI Real World Internet Architecture, "Towards the Future Internet - Emerging Trends from European Research"*. IOS Press, 2010, pp. 247–256.

[15] L. jie Zhang and Q. Zhou, "CCOA: Cloud Computing Open Architecture," in *International Conference on Web Services*, 2009, pp. 607–616.

[16] Z. Lu, J. Wu, and W. Fu, "A Novel Cloud-Oriented WS-Management-Based Resource Management Model," in *Web Services (ICWS), 2010 IEEE International Conference on*, July 2010, pp. 676–677.

[17] R. Moreno-Vozmediano, R. Montero, and I. Llorente, "Key Challenges in Cloud Computing: Enabling the Future Internet of Services," *Internet Computing, IEEE*, vol. 17, no. 4, pp. 18–25, July 2013.

[18] B. Jennings and R. Stadler, "Resource Management in Clouds: Survey and Research Challenges," *Journal of Network and Systems Management*, pp. 1–53, 2014.

[19] X. Wen, G. Gu, Q. Li, Y. Gao, and X. Zhang, "Comparison of open-source cloud management platforms: OpenStack and OpenNebula," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, May 2012, pp. 2457–2461.

[20] F. Gomez-Folgar, A. Garcia-Loureiro, T. F. Pena, and R. Valin, "Performance of the CloudStack KVM Pod Primary Storage Under NFS Version 3," in *Proceedings of the 2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications*, ser. ISPA '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 845–846.

[21] T. Benson, A. Akella, A. Shaikh, and S. Sahu, "CloudNaaS: A Cloud Networking Platform for Enterprise Applications," in *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, ser. SOCC '11. New York, NY, USA: ACM, 2011, pp. 8:1–8:13.

[22] J. F. Joan A. García *et al.*, "Logical Infrastructure Composition Layer, the GEYSERS Holistic Approach for Infrastructure Virtualisation."

[23] J. Araujo Wickboldt, L. Zambenedetti Granville, F. Schneider, D. Dudkowski, and M. Brunner, "Rethinking cloud platforms: Network-aware flexible resource allocation in IaaS clouds," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, May 2013, pp. 450–456.

[24] A. Rai, R. Bhagwan, and S. Guha, "Generalized Resource Allocation for the Cloud," in *Proceedings of the Third ACM Symposium on Cloud Computing*, ser. SoCC '12. New York, NY, USA: ACM, 2012, pp. 15:1–15:12.

[25] Q. Duan, Y. Yan, and A. Vasilakos, "A Survey on Service-Oriented Network Virtualization Toward Convergence of Networking and Cloud Computing," *Network and Service Management, IEEE Transactions on*, vol. 9, no. 4, pp. 373–392, December 2012.

[26] I. Houidi, W. Louati, D. Zeghlache, and S. Baucke, "Virtual Resource Description and Clustering for Virtual Network Discovery," in *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, June 2009, pp. 1–6.

[27] K. M. Metwally, A. Jarray, and A. Karmouch, "Two-Phase Ontology-based Resource Allocation Approach for IaaS Cloud Service," in *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC) (CCNC 2015)*, Las Vegas, USA, Jan. 2015, pp. 790–795.

[28] K. Metwally, A. Jarray, and A. Karmouch, "MILP-based Approach for Efficient Cloud IaaS Resource Allocation," in *2015 IEEE 8th International Conference on Cloud Computing (CLOUD)*, June 2015.

[29] K. M. Metwally, A. Jarray, and A. Karmouch, "A Cost-Efficient QoS-Aware Model for Cloud IaaS Resource Allocation in Large Datacenters," in *4th IEEE International Conference on Cloud Networking (CLOUDNET'15)*, Niagara Falls, Canada, Oct. 2015.

[30] J. Kang and K. Sim, "Cloudle: An Ontology-Enhanced Cloud Service Search Engine," in *Web Information Systems Engineering – WISE 2010 Workshops*, ser. Lecture Notes in Computer Science, D. Chiu, L. Bellatreche, H. Sasaki, H.-f. Leung, S.-C. Cheung, H. Hu, and J. Shao, Eds. Springer Berlin Heidelberg, 2011, vol. 6724, pp. 416–427.

[31] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[32] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf, *NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology (Special Publication 500-292).* USA: CreateSpace Independent Publishing Platform, 2012.

[33] Q. Li, Q. Hao, L. Xiao, and Z. Li, "Adaptive Management of Virtualized Resources in Cloud Computing Using Feedback Control," in *Information Science and Engineering (ICISE), 2009 1st International Conference on*, Dec 2009, pp. 99–102.

[34] D. Chisnall, *The Definitive Guide to the Xen Hypervisor*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2007.

[35] H. Kim and N. Feamster, "Improving network management with software defined networking," *Communications Magazine, IEEE*, vol. 51, no. 2, pp. 114–119, February 2013.

[36] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet Impasse Through Virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, Apr. 2005.

[37] J. Turner and D. Taylor, "Diversifying the Internet," in *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, vol. 2, Dec 2005, pp. 6 pp.–760.

[38] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, no. 5, pp. 862–876, Apr. 2010.

[39] J. Turner and D. Taylor, "Diversifying the Internet," in *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, vol. 2, Dec 2005, pp. 6 pp.–760.

[40] J. Carapinha and J. Jiménez, "Network Virtualization: A View from the Bottom," in *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures*, ser. VISA '09. New York, NY, USA: ACM, 2009, pp. 73–80.

[41] R. Bless and C. Werle, "Control Plane Issues in the 4WARD Network Virtualization Architecture," *Electronic Communications of the EASST*, vol. 17, 2009.

[42] N. Feamster, L. Gao, and J. Rexford, "How to Lease the Internet in Your Spare Time," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 61–64, Jan. 2007.

[43] M. Chowdhury, M. Rahman, and R. Boutaba, "ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 1, pp. 206 –219, feb. 2012.

[44] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "VNR Algorithm: A Greedy Approach For Virtual Networks Reconfigurations," in *VNR Algorithm : A Greedy Approach For Virtual Networks Reconfigurations.* Houston, États-Unis: IEEE, 2011, pp. 6–11.

[45] F. Hao, T. V. Lakshman, S. Mukherjee, and H. Song, "Enhancing dynamic cloud-based services using network virtualization," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 67–74, Jan. 2010.

[46] J. He, R. Zhang-shen, Y. Li, C. yen Lee, J. Rexford, and M. Chiang, "DaVinci: Dynamically Adaptive Virtual Networks for a Customized Internet," in *in Proc. CoNEXT*, 2008.

[47] J. Carapinha, P. Feil, P. Weissmann, S. E. Thorsteinsson, c. Etemoğlu, O. Ingpórsson, S. Çiftçi, and M. Melo, "Network Virtualization - Opportunities and Challenges for Operators," in *Proceedings of the Third Future Internet Conference on Future Internet*, ser. FIS'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 138–147.

[48] N. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *Communications Magazine, IEEE*, vol. 47, no. 7, pp. 20–26, July 2009.

[49] S. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *Communications Magazine, IEEE*, vol. 51, no. 2, pp. 136–141, February 2013.

[50] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[51] C. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *Computers, IEEE Transactions on*, vol. C-34, no. 10, pp. 892–901, Oct 1985.

[52] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Aug. 2008.

[53] 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," in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, ser. SIGCOMM '09. New York, NY, USA: ACM, 2009, pp. 63–74.

[54] J.-Y. Shin, B. Wong, and E. G. Sirer, "Small-world datacenters," in *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, ser. SOCC '11. New York, NY, USA: ACM, 2011, pp. 2:1–2:13.

[55] L. Popa, S. Ratnasamy, G. Iannaccone, A. Krishnamurthy, and I. Stoica, "A Cost Comparison of Datacenter Network Architectures," in *Proceedings of the 6th International COnference*, ser. Co-NEXT '10. New York, NY, USA: ACM, 2010, pp. 16:1–16:12.

[56] D. Abts and B. Felderman, "A Guided Tour of Data-center Networking," *Commun. ACM*, vol. 55, no. 6, pp. 44–51, Jun. 2012.

[57] D. S. Marcon, R. R. Oliveira, L. P. Gaspary, and M. P. Barcellos, "Datacenter networks and relevant standards," in *Cloud Services, Networking, and Management*. John Wiley & Sons, Inc, 2015, pp. 73–104.

[58] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. Guedes, "Gatekeeper: Supporting Bandwidth Guarantees for Multi-tenant Datacenter Networks," in *Proceedings of the 3rd Conference on I/O Virtualization*, ser. WIOV'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 6–6.

[59] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, and A. Rindos, "Sddc: A software defined datacenter experimental framework," in *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, Aug 2015, pp. 189–194.

[60] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson, "Approximation Algorithms for NP-hard Problems," D. S. Hochbaum, Ed. Boston, MA, USA: PWS Publishing Co., 1997, ch. Approximation Algorithms for Bin Packing: A Survey, pp. 46–93.

[61] G. Jung, K. Joshi, M. Hiltunen, R. Schlichting, and C. Pu, "Generating Adaptation Policies for Multi-tier Applications in Consolidated Server Environments," in *Autonomic Computing, 2008. ICAC '08. International Conference on*, June 2008, pp. 23–32.

[62] R. Gupta, S. Bose, S. Sundarrajan, M. Chebiyam, and A. Chakrabarti, "A Two Stage Heuristic Algorithm for Solving the Server Consolidation Problem with Item-Item and Bin-Item Incompatibility Constraints," in *Services Computing, 2008. SCC '08. IEEE International Conference on*, vol. 2, July 2008, pp. 39–46.

[63] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong, "EnaCloud: An Energy-Saving Application Live Placement Approach for Cloud Computing Environments," in *Cloud Computing, 2009. CLOUD '09. IEEE International Conference on*, Sept 2009, pp. 17–24.

[64] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient Resource Provisioning in Compute Clouds via VM Multiplexing," in *Proceedings of the 7th International Conference on Autonomic Computing*, ser. ICAC '10. New York, NY, USA: ACM, 2010, pp. 11–20.

[65] L. Shi, B. Butler, D. Botvich, and B. Jennings, "Provisioning of requests for virtual machine sets with placement constraints in IaaS clouds," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, May 2013, pp. 499–505.

[66] D. Jayasinghe, C. Pu, T. Eilam, M. Steinder, I. Whally, and E. Snible, "Improving Performance and Availability of Services Hosted on IaaS Clouds with Structural Constraint-Aware Virtual Machine Placement," in *Services Computing (SCC), 2011 IEEE International Conference on*, July 2011, pp. 72–79.

[67] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live Migration of Virtual Machines," in *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, ser. NSDI'05. Berkeley, CA, USA: USENIX Association, 2005, pp. 273–286.

[68] S. Akoush, R. Sohan, A. Rice, A. Moore, and A. Hopper, "Predicting the Performance of Virtual Machine Migration," in *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on*, Aug 2010, pp. 37–46.

[69] S. Kikuchi and Y. Matsumoto, "What will happen if cloud management operations burst out?" in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, May 2011, pp. 97–104.

[70] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast Transparent Migration for Virtual Machines," in *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ser. ATEC '05. Berkeley, CA, USA: USENIX Association, 2005, pp. 25–25.

[71] S. Bazarbayev, M. Hiltunen, K. Joshi, W. Sanders, and R. Schlichting, "Content-Based Scheduling of Virtual Machines (VMs) in the Cloud," in *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*, July 2013, pp. 93–101.

[72] M. R. Hines and K. Gopalan, "Post-copy Based Live Virtual Machine Migration Using Adaptive Pre-paging and Dynamic Self-ballooning," in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, ser. VEE '09. New York, NY, USA: ACM, 2009, pp. 51–60.

[73] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, "Live Migration of Virtual Machine Based on Full System Trace and Replay," in *Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '09. New York, NY, USA: ACM, 2009, pp. 101–110.

[74] V. Mann, A. Vishnoi, K. Kannan, and S. Kalyanaraman, "CrossRoads: Seamless VM mobility across data centers through software defined networking," in *Network*

Operations and Management Symposium (NOMS), 2012 IEEE, April 2012, pp. 88–96.

[75] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and Gray-box Strategies for Virtual Machine Migration," in *Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation*, ser. NSDI'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 17–17.

[76] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "Kingfisher: Cost-aware elasticity in the cloud," in *INFOCOM, 2011 Proceedings IEEE*, April 2011, pp. 206–210.

[77] X. Zhang, Z.-Y. Shae, S. Zheng, and H. Jamjoom, "Virtual machine migration in an over-committed cloud," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*, April 2012, pp. 196–203.

[78] H. Xu and B. Li, "Egalitarian stable matching for VM migration in cloud computing," in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, April 2011, pp. 631–636.

[79] D. Gale and L. S. Shapley, "College Admissions and the Stability of Marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. pp. 9–15, 1962.

[80] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic Placement of Virtual Machines for Managing SLA Violations," in *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, May 2007, pp. 119–128.

[81] M. Hadji and D. Zeghlache, "Minimum Cost Maximum Flow Algorithm for Dynamic Resource Allocation in Clouds," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, June 2012, pp. 876–882.

[82] X. Meng, V. Pappas, and L. Zhang, "Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement," in *INFOCOM, 2010 Proceedings IEEE*, March 2010, pp. 1–9.

[83] L. Hu, K. D. Ryu, M. Silva, and K. Schwan, "v-Bundle: Flexible Group Resource Offerings in Clouds," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, June 2012, pp. 406–415.

[84] V. Shrivastava, P. Zerfos, K.-W. Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers," in *INFOCOM, 2011 Proceedings IEEE*, April 2011, pp. 66–70.

[85] X. Wen, K. Chen, Y. Chen, Y. Liu, Y. Xia, and C. Hu, "VirtualKnotter: Online Virtual Machine Shuffling for Congestion Resolving in Virtualized Datacenter," in *Proceedings of the 2012 IEEE 32Nd International Conference on Distributed Computing Systems*, ser. ICDCS '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 12–21.

[86] Q. Zhang, Q. Zhu, M. Zhani, and R. Boutaba, "Dynamic Service Placement in Geographically Distributed Clouds," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, June 2012, pp. 526–535.

[87] M. Alicherry and T. Lakshman, "Network aware resource allocation in distributed clouds," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 963–971.

[88] 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 *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, May 2012, pp. 498–506.

[89] V. Emeakaroha, I. Brandic, M. Maurer, and I. Breskovic, "SLA-Aware Application Deployment and Resource Allocation in Clouds," in *Computer Software and Appli-*

cations Conference Workshops (COMPSACW), 2011 IEEE 35th Annual, July 2011, pp. 298–303.

[90] S. Ghasemi-Falavarjani, M. Nematbakhsh, and B. S. Ghahfarokhi, "Context-aware multi-objective resource allocation in mobile cloud," *Computers & Electrical Engineering*, vol. 44, pp. 218 – 240, 2015.

[91] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing," *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, May 2012.

[92] L. Wu, S. Garg, and R. Buyya, "SLA-Based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments," in *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, May 2011, pp. 195–204.

[93] T. Genez, L. Bittencourt, and E. Madeira, "Workflow scheduling for SaaS / PaaS cloud providers considering two SLA levels," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*, April 2012, pp. 906–912.

[94] Z. Yusoh and M. Tang, "Composite SaaS placement and resource optimization in cloud computing using evolutionary algorithms," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, June 2012, pp. 590–597.

[95] ——, "A penalty-based grouping genetic algorithm for multiple composite SaaS components clustering in cloud," in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, Oct 2012, pp. 1396–1401.

[96] P. Bodík, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, and I. Stoica, "Surviving Failures in Bandwidth-constrained Datacenters," in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Pro-*

tocols for Computer Communication, ser. SIGCOMM '12. New York, NY, USA: ACM, 2012, pp. 431–442.

[97] G. Feng, S. Garg, R. Buyya, and W. Li, "Revenue Maximization Using Adaptive Resource Provisioning in Cloud Computing Environments," in *Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing*, ser. GRID '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 192–200.

[98] H. Xu and B. Li, "Dynamic Cloud Pricing for Revenue Maximization," *Cloud Computing, IEEE Transactions on*, vol. 1, no. 2, pp. 158–171, July 2013.

[99] S. Khan and I. Ahmad, "A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 20, no. 3, pp. 346–360, March 2009.

[100] M. Guzek, J. Pecero, B. Dorronsoro, P. Bouvry, and S. Khan, "A Cellular Genetic Algorithm for scheduling applications and energy-aware communication optimization," in *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, June 2010, pp. 241–248.

[101] F. Pinel, J. Pecero, P. Bouvry, and S. Khan, "Memory-Aware Green Scheduling on Multi-core Processors," in *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, Sept 2010, pp. 485–488.

[102] D. Kliazovich, P. Bouvry, and S. 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.

[103] Y. C. Lee and A. Zomaya, "Minimizing Energy Consumption for Precedence-Constrained Applications Using Dynamic Voltage Scaling," in *Cluster Computing*

and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on, May 2009, pp. 92–99.

[104] J. Kolodziej, S. Khan, and F. Xhafa, "Genetic Algorithms for Energy-Aware Scheduling in Computational Grids," in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2011 International Conference on*, Oct 2011, pp. 17–24.

[105] F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres, and H. Tenhunen, "Using Ant Colony System to Consolidate VMs for Green Cloud Computing," *Services Computing, IEEE Transactions on*, vol. 8, no. 2, pp. 187–198, March 2015.

[106] W. Vogels, "Beyond Server Consolidation," *Queue*, vol. 6, no. 1, pp. 20–26, Jan. 2008.

[107] E. Feller, C. Morin, and A. Esnault, "A case for fully decentralized dynamic VM consolidation in clouds," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, Dec 2012, pp. 26–33.

[108] A. Murtazaev and S. Oh, "Sercon: Server consolidation algorithm using live migration of virtual machines for green computing," *IETE Technical Review*, vol. 28, no. 3, pp. 212–231, 2011.

[109] M. Marzolla, O. Babaoglu, and F. Panzieri, "Server consolidation in Clouds through gossiping," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, June 2011, pp. 1–6.

[110] 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.

[111] H. Goudarzi, M. Ghasemazar, and M. Pedram, "SLA-based Optimization of Power and Migration Cost in Cloud Computing," in *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, May 2012, pp. 172–179.

[112] C. Mastroianni, M. Meo, and G. Papuzzo, "Probabilistic Consolidation of Virtual Machines in Self-Organizing Cloud Data Centers," *Cloud Computing, IEEE Transactions on*, vol. 1, no. 2, pp. 215–228, July 2013.

[113] Z. Xiao, W. Song, and Q. Chen, "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 6, pp. 1107–1117, June 2013.

[114] M. A. Adnan, R. Sugihara, and R. K. Gupta, "Energy Efficient Geographical Load Balancing via Dynamic Deferral of Workload," in *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing*, ser. CLOUD '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 188–195.

[115] M. Ilyas, S. Raza, C.-C. Chen, Z. Uzmi, and C.-N. Chuah, "RED-BL: Energy solution for loading data centers," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 2866–2870.

[116] H. Xu, C. Feng, and B. Li, "Temperature Aware Workload Management in Geo-distributed Datacenters," *SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 1, pp. 373–374, Jun. 2013.

[117] D. Xu and X. Liu, "Geographic trough filling for internet datacenters," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 2881–2885.

[118] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M. Neely, "Data centers power reduction: A two time scale approach for delay tolerant workloads," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 1431–1439.

[119] H. Qian and M. Rabinovich, "Application Placement and Demand Distribution in a Global Elastic Cloud: A Unified Approach," in *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*. San Jose, CA: USENIX, 2013, pp. 1–12.

[120] "Apache Software Foundation," http://www.apache.org/, Last visited: July 2015.

[121] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-Source Cloud-Computing System," in *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, May 2009, pp. 124–131.

[122] "Amazon Web Service," http://aws.amazon.com/, Last visited: July 2015.

[123] "Google App Engine," https://cloud.google.com/appengine/, Last visited: July 2015.

[124] "Azure Services Platform," http://azure.microsoft.com/, Last visited: July 2015.

[125] D. Milojicic, I. M. Llorente, and R. S. Montero, "OpenNebula: A Cloud Management Tool," *IEEE Internet Computing*, vol. 15, no. 2, pp. 11–14, 2011.

[126] K. Keahey, T. Freeman, J. Lauret, and D. Olson, "Virtual workspaces for scientific applications," *J. Phys.: Conf. Ser.*, vol. 78, no. 1, pp. 012 038+, 2007.

[127] E. Feller, L. Rilling, and C. Morin, "Snooze: A Scalable and Autonomic Virtual Machine Management Framework for Private Clouds," in *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, May 2012, pp. 482–489.

[128] A.-F. Antonescu, P. Robinson, L. M. Contreras-Murillo, J. Aznar, S. Soudan, F. Anhalt, and J. A. Garcia-Espin, "Towards Cross Stratum SLA Management with the

GEYSERS Architecture," in *Proceedings of the 2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications*, ser. ISPA '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 527–533.

[129] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging {IT} platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599 – 616, 2009.

[130] E. Escalona, R. Nejabati, J. Jimenez, and A. Tzanakaki, "GEYSERS: A Novel Architecture for Virtualization and Co-Provisioning of Dynamic Optical Networks and IT Services," in *Future Network and Mobile Summit 2011*, 2010.

[131] J. W. Charles Reiss, "Google cluster-usage traces: format+schema," Tech. Rep., 2011.

[132] "Mantychore Project," http://www.mantychore.eu/, Last visited: July 2015.

[133] L. Badger, B. Bachorik, and D. Brackney, "SAJACC Working Group Recommendations to NIST," Gaithersburg, MD, United States, Tech. Rep., 2013.

[134] (2009, Aug.) Cloud Computing Use Cases Whitepaper.

[135] C. Vecchiola, S. Pandey, and R. Buyya, "High-Performance Cloud Computing: A View of Scientific Applications," in *Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on*, Dec 2009, pp. 4–16.

[136] J. Shamsi, M. A. Khojaye, and M. A. Qasmi, "Data-Intensive Cloud Computing: Requirements, Expectations, Challenges, and Solutions," *J. Grid Comput.*, vol. 11, no. 2, pp. 281–310, Jun. 2013.

[137] D. Talia, "Clouds for Scalable Big Data Analytics," *Computer*, vol. 46, no. 5, pp. 98–101, May 2013.

[138] K. M. Sim, "Agent-Based Cloud Computing," *IEEE Transactions on Services Computing*, vol. 5, no. 4, pp. 564–577, 2012.

[139] Z. Wang, Y. Han, T. Lin, Y. Xu, S. Ci, and H. Tang, "Topology-aware virtual network embedding based on closeness centrality." *Frontiers of Computer Science*, vol. 7, no. 3, pp. 446–457, 2013.

[140] J. Van der Ham, P. Grosso, R. van der Pol, A. Toonk, and C. de Laat, "Using the Network Description Language in Optical Networks," in *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, May 2007, pp. 199–205.

[141] Y. X. (renci, I. B. (renci, K. A. (ncsu, Y. X. (renci, I. B. (renci, and K. A. (ncsu, "Using Semantic Web Description Techniques for Managing Resources in a Multi-Domain Infrastructure-as-a-Service Environment," 2013.

[142] M. Ghijsen, J. Van Der Ham, P. Grosso, C. Dumitru, H. Zhu, Z. Zhao, and C. De Laat, "A Semantic-web Approach for Modeling Computing Infrastructures," *Comput. Electr. Eng.*, vol. 39, no. 8, pp. 2553–2565, Nov. 2013.

[143] W3C, "OWL 2 Web Ontology Language Document Overview (Second Edition)," 2012, [Online; accessed 10-March-2014].

[144] G. Antoniou and F. Harmelen, "Web Ontology Language: OWL," in *Handbook on Ontologies*, ser. International Handbooks on Information Systems, S. Staab and R. Studer, Eds. Springer Berlin Heidelberg, 2009, pp. 91–110.

[145] J. Bock, P. Haase, Q. Ji, and R. Volz, "Benchmarking OWL Reasoners," in *Proceedings of the AArea2008 Workshop*, vol. 350. http://ceur-ws.org: CEUR Workshop Proceedings, June 2008.

[146] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF – W3C Recommendation," W3C, Tech. Rep., 2008.

[147] M. S. Hossain, M. M. Hassan, M. Al-Qurishi, and A. S. Alghamdi, "Resource Allocation for Service Composition in Cloud-based Video Surveillance Platform." in *ICME Workshops*, J. Z. 0002, D. Schonfeld, and D. D. Feng, Eds. IEEE, 2012, pp. 408–412.

[148] M. Zhang, R. Ranjan, S. Nepal, M. Menzel, and A. Haller, "A Declarative Recommender System for Cloud Infrastructure Services Selection," in *Proceedings of the 9th International Conference on Economics of Grids, Clouds, Systems, and Services*, ser. GECON'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 102–113.

[149] J. M. García, D. Ruiz, and A. R. Cortés, "Improving semantic web services discovery using SPARQL-based repository filtering." *J. Web Sem.*, vol. 17, pp. 12–24, 2012.

[150] C. Mao, J. Chen, and X. Yu, "An Empirical Study on Meta-Heuristic Search-Based Web Service Composition," in *Proceedings of the 2012 IEEE Ninth International Conference on e-Business Engineering*, ser. ICEBE '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 117–122.

[151] Y. Syu, Y.-Y. FanJiang, J.-Y. Kuo, and S.-P. Ma, "A Genetic Algorithm with Prioritized Objective Functions for Service Composition," in *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*, March 2012, pp. 932–937.

[152] J. Gutierrez-Garcia and K.-M. Sim, "Self-Organizing Agents for Service Composition in Cloud Computing," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, Nov 2010, pp. 59–66.

[153] T. Han and K. M. Sim, "An ontology-enhanced cloud service discovery system," in *Proc. of the International MultiConference of Engineers and Computer Scientists, Hong Kong*, 2010.

[154] L. Wang, J. Shen, and J. Yong, "A survey on bio-inspired algorithms for web service composition," in *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on*, May 2012, pp. 569–574.

[155] Q. Duan, "Modeling and Performance Analysis on Network Virtualization for Composite Network-Cloud Service Provisioning," in *Services (SERVICES), 2011 IEEE World Congress on*, July 2011, pp. 548–555.

[156] A. Klein, F. Ishikawa, and S. Honiden, "Towards Network-aware Service Composition in the Cloud," in *Proceedings of the 21st International Conference on World Wide Web*, ser. WWW '12. New York, NY, USA: ACM, 2012, pp. 959–968.

[157] M. Wang, X. Li, and X. Qiao, "Semantic web service discovery based on user preference cluster," in *Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*, Oct 2010, pp. 939–944.

[158] T. Kawamura, J.-A. De Blasio, T. Hasegawa, M. Paolucci, and K. Sycara, "Public Deployment of Semantic Service Matchmaker with UDDI Business Registry," in *The Semantic Web – ISWC 2004*, ser. Lecture Notes in Computer Science, S. McIlraith, D. Plexousakis, and F. van Harmelen, Eds. Springer Berlin Heidelberg, 2004, vol. 3298, pp. 752–766.

[159] S. B. Mokhtar, D. Preuveneers, N. Georgantas, V. Issarny, and Y. Berbers, "EASY: Efficient semAntic Service discoverY in Pervasive Computing Environments with QoS and Context Support," *J. Syst. Softw.*, vol. 81, no. 5, pp. 785–808, May 2008.

[160] A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler, "WSMX - a semantic service-oriented architecture," in *In Proceedings of the International Conference on Web Service (ICWS 2005*, 2005, pp. 321–328.

[161] G. Ting, W. Haiyang, Z. Naihui, and L. Fei, "An Improved Way to Facilitate Composition-Oriented Semantic Service Discovery," in *Computer Engineering and Technology, 2009. ICCET '09. International Conference on*, vol. 2, Jan 2009, pp. 156–160.

[162] K. Iqbal, M. Sbodio, V. Peristeras, and G. Giuliani, "Semantic Service Discovery using SAWSDL and SPARQL," in *Semantics, Knowledge and Grid, 2008. SKG '08. Fourth International Conference on*, Dec 2008, pp. 205–212.

[163] E. Sirin, B. Parsia, and J. Hendler, "Filtering and selecting semantic web services with interactive composition techniques," *Intelligent Systems, IEEE*, vol. 19, no. 4, pp. 42–49, Jul 2004.

[164] J. Vaidya, B. Shafiq, A. V. Paliwal, H. Xiong, and N. Adam, "Semantics-Based Automated Service Discovery," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 260–275, 2012.

[165] F. Lécué, E. Silva, and L. Pires, "A Framework for Dynamic Web Services Composition," in *Emerging Web Services Technology, Volume II*, ser. Whitestein Series in Software Agent Technologies and Autonomic Computing, T. Gschwind and C. Pautasso, Eds. Birkhäuser Basel, 2008, pp. 59–75.

[166] A. N. Mian, R. Beraldi, and R. Baldoni, "Identifying Open Problems in Random Walk based Service Discovery in Mobile Adhoc Networks." ser. LNI, vol. 165. GI, 2010, pp. 91–102.

[167] S. university, "Protege," Stanford Center for Biomedical Informatics Research," [Online; accessed 1-December-2013].

[168] G. S. Kumar and B. Rajkumar, "NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations." IEEE Computer Society, 2011, pp. 105–113.

[169] A. Ahmed, Z. M. Faten, L. Rami, B. Raouf, and P. Guy, "Greenhead: Virtual Data Center Embedding Across Distributed Infrastructures," *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, January-June 2013.

[170] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*. Wiley-Interscience, 2004.

[171] A. Jarray and A. Karmouch, "Column generation approach for one-shot virtual network embedding," in *Globecom Workshops (GC Wkshps), 2012 IEEE*, Dec 2012, pp. 863–868.

[172] M. Chowdhury, M. Rahman, and R. Boutaba, "ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 1, pp. 206–219, Feb 2012.

[173] N. Farooq Butt, M. Chowdhury, and R. Boutaba, "Topology-awareness and Reoptimization Mechanism for Virtual Network Embedding," in *Proceedings of the 9th IFIP TC 6 International Conference on Networking*, ser. NETWORKING'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 27–39.

[174] J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Yousif, "Autonomic resource management in virtualized data centers using fuzzy logic-based approaches," *Cluster Computing*, vol. 11, no. 3, pp. 213–227, 2008.

[175] W. Zhang, H. Qian, C. E. Wills, and M. Rabinovich, "Agile Resource Management in a Virtualized Data Center," in *Proceedings of the First Joint WOSP/SIPEW International Conference on Performance Engineering*, ser. WOSP/SIPEW '10. New York, NY, USA: ACM, 2010, pp. 129–140.

[176] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755 – 768, 2012, special Section: Energy efficiency in large-scale distributed systems.

[177] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein, "Dynamic Energy-aware Capacity Provisioning for Cloud Computing Environments," in *Proceedings of the 9th International Conference on Autonomic Computing*, ser. ICAC '12. New York, NY, USA: ACM, 2012, pp. 145–154.

[178] K. Ye, D. Huang, X. Jiang, H. Chen, and S. Wu, "Virtual Machine Based Energy-Efficient Data Center Architecture for Cloud Computing: A Performance Perspective," 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. 171–178.

[179] M. Zhani, Q. Zhang, G. Simon, and R. Boutaba, "VDC Planner: Dynamic migration-aware Virtual Data Center embedding for clouds," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, May 2013, pp. 18–25.

[180] srikanth kandula, jitu padhye, and victor bahl, "Flyways to De-Congest Data Center Networks," in *8th ACM Workshop on Hot Topics in Networks*. Association for Computing Machinery, Inc., October 2009.

[181] V. Valancius, N. Laoutaris, C. Diot, P. Rodriguez, and L. Massoulié, "Greening the Internet with Nano Data Centers," in *Proceedings of the 5 th International Conference on Emerging Networking Experiments and Technologies*, 2009, pp. 37–48.

[182] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing — the business perspective," *Decision Support Systems*, vol. 51, no. 1, pp. 176 – 189, 2011.

[183] C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervello-Pastor, and A. Monje, "On the optimal allocation of virtual resources in cloud computing networks," *Computers, IEEE Transactions on*, vol. 62, no. 6, pp. 1060–1071, June 2013.

[184] A. Beloglazov and R. Buyya, "Energy Efficient Resource Management in Virtualized Cloud Data Centers," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, May 2010, pp. 826–831.

[185] J. Desrosiers and M. Lübbecke, "A Primer in Column Generation," in *Column Generation*, G. Desaulniers, J. Desrosiers, and M. Solomon, Eds. Springer US, 2005, pp. 1–32.

[186] G. B. Dantzig and P. Wolfe, "Decomposition Principle for Linear Programs," *Operations Research*, vol. 8, no. 1, pp. 101–111, 1960.

[187] M. Alicherry and T. Lakshman, "Network aware resource allocation in distributed clouds," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 963–971.

[188] A. Jarray and A. Karmouch, "Periodical auctioning for QoS aware virtual network embedding," in *Quality of Service (IWQoS), 2012 IEEE 20th International Workshop on*, June 2012, pp. 1–4.

[189] D. Stefani Marcon, R. Ruas Oliveira, M. Cardoso Neves, L. Salete Buriol, L. Gaspary, and M. Pilla Barcellos, "Trust-based grouping for cloud datacenters: Improving security in shared infrastructures," in *IFIP Networking Conference, 2013*, May 2013, pp. 1–9.

[190] "IBM ILOG CPLEX Optimizer," http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, Last visited: July 2015.

[191] F. Wuhib, R. Stadler, and H. Lindgren, "Dynamic Resource Allocation with Management Objectives: Implementation for an OpenStack Cloud," in *Proceedings of the 8th International Conference on Network and Service Management*, ser. CNSM '12. Laxenburg, Austria, Austria: International Federation for Information Processing, 2013, pp. 309–315.

[192] R. Mian and P. Martin, "Executing Data-Intensive Workloads in a Cloud," in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012)*, ser. CCGRID '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 758–763.

[193] R. Tudoran, A. Costan, R. Wang, L. Bouge, and G. Antoniu, "Bridging Data in the Clouds: An Environment-Aware System for Geographically Distributed Data Transfers," in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, May 2014, pp. 92–101.

[194] D. Irwin, P. Shenoy, E. Cecchet, and M. Zink, "Resource management in data-intensive clouds: Opportunities and challenges," in *Local and Metropolitan Area Networks (LANMAN), 2010 17th IEEE Workshop on*, May 2010, pp. 1–6.

[195] S. Shen, V. van Beek, and A. Iosup, "Statistical Characterization of Business-Critical Workloads Hosted in Cloud Datacenters," *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2015.

[196] K. Sripanidkulchai, S. Sahu, Y. Ruan, A. Shaikh, and C. Dorai, "Are Clouds Ready for Large Distributed Applications?" *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, pp. 18–23, Apr. 2010.

[197] I. Houidi, W. Louati, and D. Zeghlache, "A Distributed Virtual Network Mapping Algorithm," in *Communications, 2008. ICC '08. IEEE International Conference on*, May 2008, pp. 5634–5640.

[198] I. Houidi, W. Louati, W. B. Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Computer Networks*, vol. 55, no. 4, pp. 1011 – 1023, 2011, special Issue on Architectures and Protocols for the Future Internet.

[199] F. Samuel, M. Chowdhury, and R. Boutaba, "PolyViNE: policy-based virtual network embedding across multiple domains," *Journal of Internet Services and Applications*, vol. 4, no. 1, 2013.

[200] Q. Zhang, Q. Zhu, M. Zhani, and R. Boutaba, "Dynamic Service Placement in Geographically Distributed Clouds," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, June 2012, pp. 526–535.

[201] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.

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

[203] P. R. Lewis, F. Faniyi, R. Bahsoon, and X. Yao, "Markets and Clouds: Adaptive and Resilient Computational Resource Allocation inspired by Economics," in *Adaptive, Dynamic, and Resilient Systems*, N. Suri and G. Cabri, Eds. Boca Raton, FL, USA: CRC Press, 2014.

[204] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The Cost of a Cloud: Research Problems in Data Center Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Dec. 2008.

[205] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, ser. SIGCOMM '08. ACM, 2008, pp. 63–74.

[206] S. H. Clearwater, Ed., *Market-based Control: A Paradigm for Distributed Resource Allocation*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 1996.

[207] P. Samimi, Y. Teimouri, and M. Mukhtar, "A combinatorial double auction resource allocation model in cloud computing," *Information Sciences*, pp. –, 2014.

[208] K. Xu, Y. Zhang, X. Shi, H. Wang, Y. Wang, and M. Shen, "Online combinatorial double auction for mobile cloud computing markets," in *IEEE 33rd International Performance Computing and Communications Conference, IPCCC 2014, Austin, TX, USA, December 5-7, 2014*, 2014, pp. 1–8.

[209] K. Chard, K. Bubendorfer, and P. Komisarczuk, "High Occupancy Resource Allocation for Grid and Cloud Systems, a Study with DRIVE," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10. New York, NY, USA: ACM, 2010, pp. 73–84.

[210] S. Zaman and D. Grosu, "Combinatorial Auction-Based Allocation of Virtual Machine Instances in Clouds," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, Nov 2010, pp. 127–134.

[211] W. Grover, "How a Network Can "Think Globally and Act Locally" and Avoid the Hazards of Incoherence in Distributed State Information," in *Transparent Optical Networks, 2007. ICTON '07. 9th International Conference on*, vol. 3, July 2007, pp. 178–182.

[212] C. Lee, P. Wang, and D. Niyato, "A Real-Time Group Auction System for Efficient Allocation of Cloud Internet Applications," *Services Computing, IEEE Transactions on*, vol. 8, no. 2, pp. 251–268, March 2015.

[213] D. Breitgand and A. Glikson, "Global enterprise cloud transformation: Centralize, distribute or federate?" in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, May 2013, pp. 892–895.

[214] "National Science Foundation Network," http://www.nsf.gov/, Last visited: July 2015.

[215] M. Alicherry and T. Lakshman, "Network aware resource allocation in distributed clouds," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 963–971.

[216] S. Zhang, Z. Qian, J. Wu, and S. Lu, "SEA: Stable resource allocation in geographically distributed clouds," in *Communications (ICC), 2014 IEEE International Conference on*, June 2014, pp. 2932–2937.

List of Publications

[1] K. M. Metwally, A. Jarray, and A. Karmouch, "Two-phase ontology-based resource allocation approach for iaas cloud service," in *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, Jan 2015, pp. 790–795.

[2] K. Metwally, A. Jarray, and A. Karmouch, "MILP-based Approach for Efficient Cloud IaaS Resource Allocation," in *2015 IEEE 8th International Conference on Cloud Computing (CLOUD)*, June 2015.

[3] K. M. Metwally, A. Jarray, and A. Karmouch, "A Cost-Efficient QoS-Aware Model for Cloud IaaS Resource Allocation in Large Datacenters," in *4th IEEE International Conference on Cloud Networking (CLOUDNET'15)*, Niagara Falls, Canada, Oct. 2015.

[4] K. Metwally, A. Jarray, and A. Karmouch, "A Distributed Economics-based Framework for Efficient IaaS provisioning in Geo-Datacenters," *IEEE Transactions on Cloud Computing*, Under Revision, 2016.