Cost- and Performance-Aware Resource Management in Cloud Infrastructures

Robayet Nasim

Faculty of Health, Science and Technology

Computer Science

DOCTORAL THESIS | Karlstad University Studies | 2017:21

Cost- and Performance-Aware Resource Management in Cloud Infrastructures

Robayet Nasim

DOCTORAL THESIS | Karlstad University Studies | 2017:21

Cost- and Performance-Aware Resource Management in Cloud Infrastructures

Robayet Nasim

DOCTORAL THESIS

Karlstad University Studies | 2017:21

urn:nbn:se:kau:diva-48482

ISSN 1403-8099

ISBN 978-91-7063-783-4 (print)

ISBN 978-91-7063-784-1 (pdf)

Distribution:
Karlstad University
Faculty of Health, Science and Technology
Department of Mathematics and Computer Science
SE-651 88 Karlstad, Sweden
+46 54 700 10 00

Print: Universitetstryckeriet, Karlstad 2017

WWW.KAU.SE

Cost- and Performance-Aware Resource Management in Cloud Infrastructures

ROBAYET NASIM
Department of Mathematics and Computer Science
Karlstad University

Abstract

High availability, cost effectiveness and ease of application deployment have accelerated the adoption rate of cloud computing. This fast proliferation of cloud computing promotes the rapid development of large-scale infrastructures. However, large cloud datacenters (DCs) require infrastructure, design, deployment, scalability and reliability and need better management techniques to achieve sustainable design benefits. Resources inside cloud infrastructures often operate at low utilization, rarely exceeding 20-30%, which increases the operational cost significantly, especially due to energy consumption. To reduce operational cost without affecting quality of service (QoS) requirements, cloud applications should be allocated just enough resources to minimize their completion time or to maximize utilization.

The focus of this thesis is to enable resource-efficient and performance-aware cloud infrastructures by addressing above mentioned cost and performance related challenges. In particular, we propose algorithms, techniques, and deployment strategies for improving the dynamic allocation of virtual machines (VMs) into physical machines (PMs).

For minimizing the operational cost, we mainly focus on optimizing energy consumption of PMs by applying dynamic VM consolidation methods. To make VM consolidation techniques more efficient, we propose to utilize multiple paths to spread traffic and deploy recent queue management schemes which can maximize network resource utilization and reduce both downtime and migration time for live migration techniques. In addition, a dynamic resource allocation scheme is presented to distribute workloads among geographically dispersed DCs considering their location based time varying costs due to e.g. carbon emission or bandwidth provision. For optimizing performance level objectives, we focus on interference among applications contending in shared resources and propose a novel VM consolidation scheme considering sensitivity of the VMs to their demanded resources. Further, to investigate the impact of uncertain parameters on cloud resource allocation and applications' QoS such as unpredictable variations in demand, we develop an optimization model based on the theory of robust optimization. Furthermore, in order to handle the scalability issues in the context of large scale infrastructures, a robust and fast Tabu Search algorithm is designed and evaluated.

Keywords: Cloud Computing, OpenStack, Robust Optimization, Latency, Tabu Search, Resource Management, Resource Contention, QoS.

Acknowledgements

PhD is a rewarding journey, which would not have been possible without the support of a number of people. First and foremost, I would like to express my sincere gratitude and special thanks to my supervisor, Professor Andreas J. Kassler, who has been a tremendous mentor for me. His inspirational ideas (sometimes too many!), insightful advice and helpful criticism support towards the completion of this thesis. I sincerely appreciate his understanding and exceptional patience, when things do not turn out as expected. I am also grateful to my co-supervisor, Professor Anna Brunström for her insightful comments on my research. On a personal note, I am also thankful to my MSc thesis supervisor, Associate Professor Sonja Buchegger for introducing me to research and providing guidance on developing research skills.

I would like to thank all colleagues and friends from the department of computer science at Karlstad university, in particular the Distributed and Systems and Communication Research (DISCO) group for maintaining a motivating and friendly environment for research. Specially, the help of Associate Professor Stefan Alfredsson regarding technical issues was invaluable to me as a fresh research student.

I have been very privileged to work and collaborate with a lot of very inspiring, competent, and nice people and I am happy to be able to use this space to thank all of my co-authors. In addition, I am grateful to Professor Erik Elmroth to accept the role of opponent in my thesis defense. I would also like to thank the members of the examination committee for their time and effort in the reading and assessment my PhD thesis.

I would like to thank the European Commission for their financial support through the Interreg IVB NorthSea region project ITRACT. In addition, some work presented in this thesis is financially supported by KKStiftelsen through the project READY and HITS, which I really appreciate.

A special thanks to my parents. Words cannot express how grateful I am to my father, Nasimul Ghani, and my mother, Ferdoshi Begum for all of the sacrifices that you have made on my behalf. Your prayer for me was what sustained me thus far. I also thank my brother, Ridnan Nasim (Dihan), my sister, Refata Nasim (Rimi) for their encouragement. Further, I would like to thank all of my friends and relatives who supported me to strive towards my goal. Last but not least, I would like to express appreciation to my beloved wife, Afrina Zabeen and my son, Arham Nasim for their unconditional love and immeasurable support throughout this journey. My family has always been the source of my inspiration and strength. I dedicate this thesis to them.

Robayet Nasim

Karlstad, June 2017 Robayet Nasim

List of Appended Papers

1. **Robayet Nasim**, Andreas J. Kassler. Deploying OpenStack: Virtual Infrastructure or Dedicated Hardware. In *IEEE 38th Annual International Computers, Software and Applications Conference Workshops (COMPSACW 2014)*, pages 84-89, Västerås, Sweden, July 21-25, 2014.

2. Cristian Hernandez Benet, **Robayet Nasim**, Kyoomars Alizadeh Noghani, Andreas J. Kassler. OpenStackEmu - A Cloud Testbed Combining Network Emulation with OpenStack and SDN. In *14th IEEE Annual Consumer Communications and Networking Conference (CCNC 2017)*, pages 566-568, Las Vegas, USA, January 8-11, 2017.

3. **Robayet Nasim**, Andreas J. Kassler, Ivana Podnar Ẑarko and Aleksandar Antonic̀. Mobile Publish/Subscribe System for Intelligent Transport Systems over a Cloud Environment. In *2014 IEEE International Conference on Cloud and Autonomic Computing (CAC 2014)*, pages 187-195, London, UK, September 8-12, 2014.

4. **Robayet Nasim**, Andreas J. Kassler. Network Centric Performance Improvement for Live VM Migration. In *8th IEEE International Conference on Cloud Computing (CLOUD 2015)*, pages 106-113, New York, USA, June 27-July 2, 2015.

5. **Robayet Nasim**, Mirza Mohd Shahriar Maswood, Andreas J. Kassler, Deep Medhi. Cost-Efficient Resource Scheduling under QoS Constraints for Geo-Distributed Data Centers. Under submission

6. **Robayet Nasim**, Javid Taheri, Andreas J. Kassler. Optimizing Virtual Machine Consolidation in Virtualized Datacenters Using Resource Sensitivity. In *The 8th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2016)*, pages 168-175, Luxembourg, December 12-15, 2016.

7. **Robayet Nasim**, Enrica Zola, Andreas J. Kassler. Robust Optimization for Energy-Efficient Virtual Machine Consolidation in Modern Datacenters. Under submission

8. **Robayet Nasim**, Andreas J. Kassler. A Robust Tabu Search Heuristic for VM Consolidation under Demand Uncertainity in Virtualized Datacenters. In *17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2017)*, Madrid, Spain, May 14-17, 2017.

Comments on my Participation

Paper I I am responsible for setting up the testbed and carrying out all the experimental evaluations presented in the paper. Andreas helps me developing the ideas for experimental design and gives feedback on results and evaluation. I author all the written parts with constructive comments from Andreas.

Paper II I am mainly responsible for setting up the OpenStack-based private cloud infrastructure. In addition to fruitful discussions on all aspects of this work, all co-authors help with defining use case scenarios and developing OpenStackEmu. I author the following sections in the paper: introduction and motivation, OpenStackEmu architecture and implementation (partially) and all the example use case scenarios. Further, I help in reviewing the other parts.

Paper III I am responsible for the design, implementation, and the experimental evaluations of the paper. The ideas for the experiments come out from the discussion between the co-authors and myself. I am also responsible for the implementation of the ITS application *Real-time Public Transit Tracking* and the underlying *Publish-Subscribe System MoPS* is implemented by the co-authors, Ivana and Aleksandar. I am the principal author of all parts of the paper. Andreas and Ivana have contributed during the review process.

Paper IV I am responsible for designing the testbed, and conducting experiments for evaluations. I have written most of the paper, except the abstract is written by Andreas. Andreas and I discuss regularly to decide about the experimental setup, and future steps. Andreas has also contributed proof-reading, constructive inputs and revising the paper during reviewing.

Paper V This paper is partly based on previous work by two of the co-authors, Shahriar and Deep. I am responsible for developing the model and designing the experiments. My main contributions are in joint discussion with Shahriar on the implementation and evaluation of the mathematical model. In addition, I author bulk part of the paper, except the abstract is written by Andreas and the conclusion is written by Shahriar. Further, Andreas and Deep have contributed with proof-reading and revising the paper during reviewing.

Paper VI I am the first author and responsible for writing the whole paper. The main idea for the paper is evolved from joint discussion with Javid and Andreas. I am responsible for the optimization model, implementation, and numerical evaluations of the paper. Further, Javid and Andreas have contributed with helpful feedback during the review process.

Paper VII My contribution to the work has been mathematical model design, implementation, and evaluation. Enrica has provided guidance in developing the optimization model. I have written most of the paper with valuable feedback from Andreas and Enrica. The only exception is the abstract, which is authored by Andreas.

Paper VIII I am the principal contributor of the paper. My contribution to the work has been concept development, algorithm design, and implementation. All parts of the written material is done by me. Andreas has offered his help on proof-reading, commenting and revising the paper.

Other contributions to papers, posters and deliverables

- **Robayet Nasim**, Andreas J. Kassler, "Distributed Architectures for Intelligent Transport Systems: A Survey ", Second Symposium on Network Cloud Computing and Applications (NCCA 2012), London, United Kingdom, December 3-4, 2012.
- **Robayet Nasim**, Andreas J. Kassler, "The Impact of Underlying Infrastructure on Performance of Private Cloud Deployments", In Proceedings of the 10th Swedish National Computer Networking Workshop (SNCNW 2014), Västerås, Sweden, June 2-3, 2014.
- **Robayet Nasim**, Sonja Buchegger, " XACML-Based Access Control for Decentralized Online Social Networks", In Proceedings of the 7th IEEE/ACM International Conference On Utility and Cloud Computing (UCC-2014), London, UK, December 8-11, 2014.
- **Robayet Nasim**, "Architectural Evolution of Intelligent Transport Systems (ITS) using Cloud Computing", Licentiate dissertation, Karlstad University Studies, Karlstad, Sweden, May 2015.
- **Robayet Nasim**, Cristian Hernandez Benet, Kyoomars Noghani Alizadeh, Andreas J. Kassler, " Improving VM Live Migration Experiences through SDN-based Approaches ", Poster session presented at the IAB Workshop, Karlstad, Sweden, September 26-27, 2016.
- **Robayet Nasim**, High Level Architecture Iteration 2, ITRACT Project Deliverable WP4-KAU-TD-I-020-V02-D-WP4122, Karlstad, Sweden, October 2012.
- **Robayet Nasim**, Product Selection, ITRACT Project Deliverable WP4-KAU-TD-I-020-V01-D-WP4122, December 2012.
- **Robayet Nasim**, Detailed ICT Architecture Iteration 3 (Version 1), ITRACT Project Deliverable WP4-KAU-TD-I-020-V01-D-WP4124, May 2013.
- **Robayet Nasim**, Detailed ICT Architecture Iteration 3 (Version 2), ITRACT Project Deliverable WP4-KAU-TD-I-020-V02-D-WP4124, October 2013.
- Maria Kaldenhoven, Johan Blok, Andreas J. Kassler, Andreas Arvidsson, **Robayet Nasim**, Jacob Mulder, ITRACT Work Package 4 - Best Practice Guide, ITRACT Project Deliverable WP4-HANZE-TD-I-034-v2.0-D-WP433_BestPracticeGuide.docx, January 2015.

Contents

Introductory Summary

INTRODUCTION

Computing resources have been increasingly powerful, cheaper, more flexible and available on-demand due to the drastic advancement of utility oriented service model. This technological shift has enabled the pay-per-use model and on-demand resource allocation abilities, which is known as "Cloud Computing". A cloud offers a large pool of resources such as hardware, development platforms, and services which can be dynamically configured with regard to variable demand providing features such as elasticity, load balancing, and ease of accessibility. According to the National Institute of Standards and Technology (NIST), [71] cloud computing can be defined as "*a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*"

In general, a cloud environment involves three major stakeholders. *Cloud Service Providers* own or administrate the cloud infrastructure. Examples are Amazon EC^2 [4] or Azure [13]. The second stakeholder are the *Cloud Clients* who own and operate the cloud services that are hosted on the cloud infrastructures. Examples of cloud clients are Dropbox [9], Netflix [3]. Thirdly, the *End Users* use the services offered by the cloud clients. From the high level perspective while considering both cloud clients and end users, cloud computing provides three features: *flexibility* to use and release computational resources based on their demands; *performance reliability* (in terms of service level agreements (SLA)) for the hosted applications; and *cost efficiency* as the underlying infrastructure is shared among multiple end users. Given these benefits offered by cloud computing, more and more individuals and enterprises are moving their applications from traditional infrastructures to cloud infrastructures. This in turn makes cloud providers increase the underlying capacity of their infrastructures to accommodate the increasing demands. For example, Amazon owns now at least 30 cloud infrastructures where each one has between 5000 and 8000 physical machines (PMs) [72].

Despite the prevalence of cloud infrastructures, cloud providers today face significant challenges regarding resource management. Depending on the focus of the cloud providers, the resource management follows different objectives. For example, the energy efficiency objective aims to minimize the power consumption for the cloud provider. On the other hand the balanced resource utilization objective aims that all the active PMs in a cloud infrastructure should experience a similar amount of load. Figure 1 illustrates the cloud resource management problem while considering two sample objectives for one dimension of resources e.g. CPU. On the left side, three applications' CPU demands are presented in the form of virtual machines (VMs). It is assumed that VM1, VM2 and VM3 need to be hosted in a cloud infrastructure, which has several PMs that are equipped with a certain resource capacity. On the right side, the VMs

are allocated to PMs depending on the management objective. For instance, when the goal is to minimize the total energy consumption, then VMs are packed tightly into a PM and all other PMs are powered down, but when the goal is to balance the load, then workloads are distributed equally over the two powered on PMs.

Over time, cloud Datacenters (DCs) have gone through several generations. For example, in Generation 1 enterprise consumers restore their server (hardware) images into virtualized DCs to reduce their expenditure. Generation 1 DCs provided custom scripts to control, orchestrate, and manage the virtualized platform, but did not support autoscaling, multitenancy, load balancing or other cloud services. In Generation 2, public cloud providers and open source cloud platforms such as OpenStack [83] offer advanced cloud management solutions but still lack of efficient integration between system software on the PMs and the cloud management software, particularly their operating utilization level is still prohibitively low [20]. Even on production level infrastructures during peak levels the utilization is rarely over 20%-30% [20, 35, 2] and during other times most of them are running even at lower utilization. This inefficient use of resources can lead to significant operational cost for the cloud providers. Recently, in Generation 2.5, cloud providers push containers into the mainstream and introduced cloud services around that technology. However, still additional research is required to make development and deployment easier and infrastructure management better. In particular, infrastructure resource management schemes need to assign diverse applications(e.g. e-commerce, Intelligent Transport Systems (ITS) applications, social networking) to appropriate execution units (e.g. VMs, containers or microservice architecture) and schedule it effectively for a multi-located DC environment to embrace the heterogeneity of workloads and operating costs.

The focus of this thesis is on two fundamental problems associated with the resource management in virtualized cloud infrastructures. They are centered around 1) energy-efficient dynamic resource allocation, 2) resource management decisions using application-awareness. Specifically, we address several aspects of how to select PMs in a cloud infrastructure for hosting VMs such that the allocation satisfies, at the same time, the management objective of the provider and the resource demand of all the applications. This problem is referred to as multi-objective VM placement/consolidation problem.

The first problem addressed in this thesis is of the major concern for today's cloud datacenters i.e. how to reduce their energy consumption. In spite of continuous work for green equipments, energy consumption of the cloud infrastructures all over the world is growing non-linearly (an increase of 110% from 2010 to 2015 and a predicted increase of 82% from 2015 to 2020) and a similar trend is expected for the upcoming years [73]. Further, cloud infrastructures have significant impact on carbon dioxide (CO_2) emissions which are estimated to be 2% of global emissions [27]. One of the most important reasons of energy waste in cloud infrastructures is the poor utilization of the PMs [27, 2]. Under-utilized PMs contribute significantly to energy waste due to their narrow dynamic power range. For example, a powered on but com-

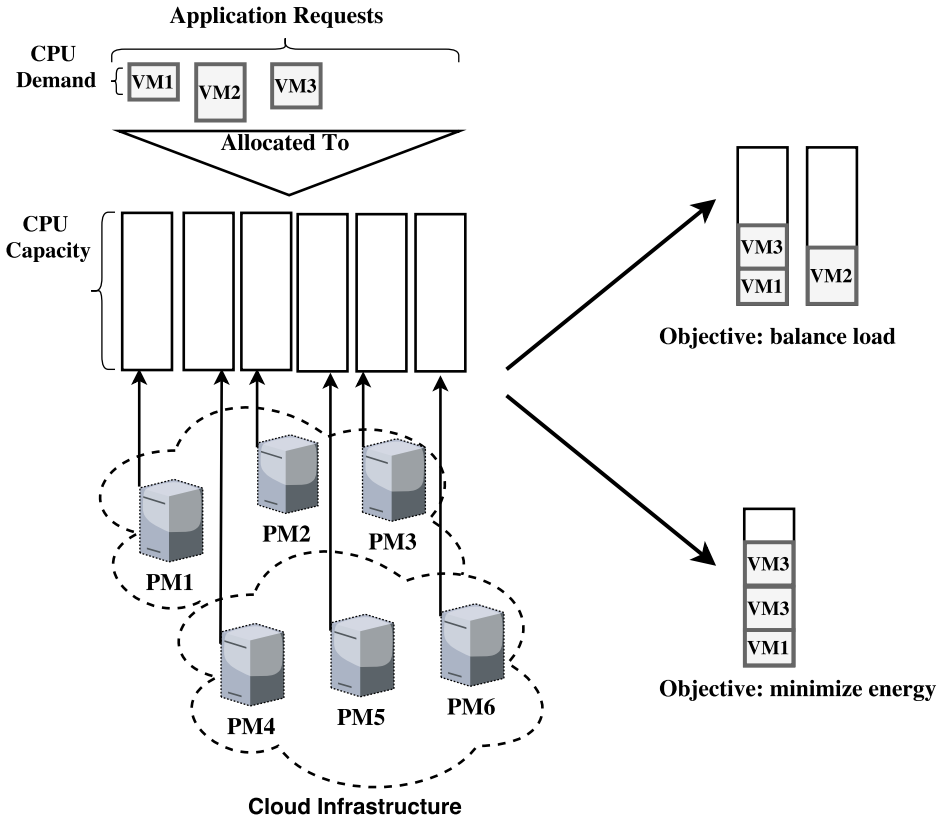

Figure 1: VM placement in a cloud infrastructure, adapted from [106]

pletely idle PM may consume 70% of its maximum power consumption [37]. Therefore, an energy-efficient VM consolidation approach should consider both non-proportional power consumption and under-utilized PMs' power wastage.

Dynamic VM consolidation usually relies on live migration techniques to re-allocate workload across PMs having different goals such as to maximize resource utilization, to manage sudden variation of the workload and to minimize co-located performance interference. However, live VM migration schemes should not violate the QoS requirements of the applications that are hosted on the VMs. For instance, applications hosted on cloud infrastructures have different performance requirements in terms of SLA such as low latency networking support for the ITS applications [76]. But often during live VM migrations, applications' performance are affected in different ways due to high network stress and inefficient network resource allocation for migration traffic. In addition, such migrations may negatively affect the performance of other deployed applications. Therefore, a detailed research is required how to alleviate the network bottlenecking problem during live VM migration and its undesirable consequences in order to improve the network resource management while minimizing applications' performance degradation level.

The second problem addressed in this thesis is how to maximize resource utilization of cloud infrastructures while meeting applications' performance requirements. Cloud infrastructures inefficient resource utilization stems from

three related factors. The workloads for most of the cloud services have different characteristics and are user-driven in nature as their resource demands are variable and non-deterministic. Sudden changes in the workload demand can make the current VMs-to-PMs allocation non-optimal or even infeasible and therefore, requires re-allocation. Focusing on these diverse demands, cloud providers such as Amazon or Google offer various categories of predefined VMs with different set of resources (CPU, Memory, etc.), as well as customized VMs where users can configure the amount of resources. However, the demand variation over time or unpredictable future demands makes it extremely difficult to allocate a precise amount of resources that applications truly require. The second reason behind cloud infrastructures' inefficient resource utilization is the *Performance unpredictability* [34] which may arise from resource contention. This resource contention can be caused by different reasons. For example, workload fluctuations or high competition for the same resource among co-hosted VMs can lead to a situation where the VMs (applications) cannot get the desired amount of resources. In addition, resource sharing of the virtualized underlying infrastructure also may cause performance degradation (known as *Performance Interference* [34]). Therefore, understanding how applications's behaviour change with workload interference is essential. As a consequence, considering the impact of uncertain or non-deterministic resource demand along with applications' sensitivity to different resource types as well as with other co-hosted demands, has significant potential to improve the resource management decisions in virtualized infrastructures.

The third reason behind cloud resource inefficiency is the difficulty to maintain a continually updated view of all the resources available in the different geographically distributed DCs, and make flexible scheduling decisions to satisfy dynamic resource requests from enterprise customers. Customers' requests for resources may arrive and leave at any time and also vary in terms of QoS requirements. For instance, some customers who are running applications like video streaming have strict delay requirements, whereas some customers are running high-performance workloads which typically require significant compute capabilities. In addition, geographically distributed DCs have widely varying cost depending on the location and time such as carbon emission taxes, electricity prices, bandwidth provision cost, etc. Consequently, resource scheduling algorithms are required to dynamically allocate resources in DCs with lower operating cost, where the costs exhibit evident location diversities, while guaranteeing an acceptable level of performance for the applications.

The contribution of this thesis lies in the proposed techniques, algorithms, and methods which help to deal with the above-mentioned challenges. In order to optimize resource utilization and power consumption of the cloud infrastructures while considering the performance requirements of the hosted applications, both mathematical models are formulated and novel algorithmic approaches are proposed. More specifically, one of the main contributions is to make VM allocation decisions aware of dynamic resource demands (in terms of uncertain variation from the expected demand and co-location interference) while reducing power consumption for the infrastructure. This

work also explores the potential benefits of considering different uncertain behaviours in modern cloud infrastructures such as demand variation or VM migrations-related resource overhead. In addition, we investigate different deployment scenarios about how to decrease the possibility of SLA violations without increasing the power consumption significantly (appended Paper VI, VII and VIII). Further, a dynamic and adaptive resource allocation approach is proposed and investigated which can improve both workload and cost distribution for enterprise cloud customers in multiple geographically distributed DC environments (appended Paper V). Furthermore, an OpenStack-based cloud testbed is designed and implemented to give researchers a way of performing experiments in their private cloud infrastructures. Additionally, we focus on the technological considerations that need to be addressed for creating a private cloud from internal resource pools (appended Paper I and II). Finally, several applications are analysed extensively in terms of different QoS requirements such as scalability, mobility, service latency using OpenStack based private cloud infrastructure (appended Paper III and IV).

The thesis contains an introductory summary, followed by re-prints of eight papers in this area, which are co-authored by the author of this thesis. As a further note, this thesis is based on and extending the licentiate thesis that already included three papers (Paper I, Paper III, and Paper IV). To provide a clear understanding of the research problems presented, Section 2 presents an overview of the background material. The research problems of this thesis and contributions are listed in Section 3. The research methods which are employed by the appended papers are discussed in Section 4. Section 5 provides a summary of all the appended papers. Finally, Section 6 concludes the introductory summary of the thesis.

BACKGROUND

The goal of this section is to present underlying concepts and technologies that are investigated in this thesis. The discussion starts with an overview of various cloud computing technological elements, ranging from virtualization technologies to applications resource requirements. Then the discussion proceeds with a brief overview about different cloud platforms including OpenStack. Finally, the section presents multi-objective resource management approaches in cloud infrastructures from infrastructure point of view, including background on combinatorial optimization- and meta-heuristic-based techniques.

2.1 Cloud Computing Paradigm

In the past ten years, cloud computing has been evolving at a rapid pace both in terms of number of hosted services and size of the underlying infrastructure [20]. It has become an essential technology behind innovation in all aspects of endeavour ranging from education to health care [12].Therefore, a large number of popular services such as social networking, video streaming, enterprise management services or even generalized storage platforms [4, 1, 13, 28, 6] are hosted on cloud infrastructures. The main reason behind the popularity of cloud computing is its unique features of on-demand resource provision, high flexibility and quick deployment. Services offered by cloud providers typically fall into one of the three major groups of service models: infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS), which are briefly descried in the next section. Furthermore, there are three different deployment models based on the availability of the cloud infrastructure to the general public. In “Public Cloud”, the cloud is available to the general public. On the other hand, “Private Cloud” restricted its use for a business or organization, and “Community Cloud” ensures limited access to a small number of businesses or organizations who have shared concerns. Since this thesis addresses the issues related to the resource management on the realm of cloud infrastructures, the rest of the section presents only relevant topics in order to facilitate an informed and smooth reading of the complete thesis.

2.2 Cloud-Enabling Technologies

Modern-day clouds are underpinned by a set of primary technological components that collectively enable key features and characteristics associated with contemporary cloud computing. Figure 2 illustrates the generalized architecture with different possible layers for a cloud environment. Typically, these layers are stacked vertically so that each layer interacts only with the upper layer directly and ensures specific services. The “Hardware Layer” is composed of physical resources ranging from servers to storage components, as well as from network equipments to power systems. This is the core layer for all different

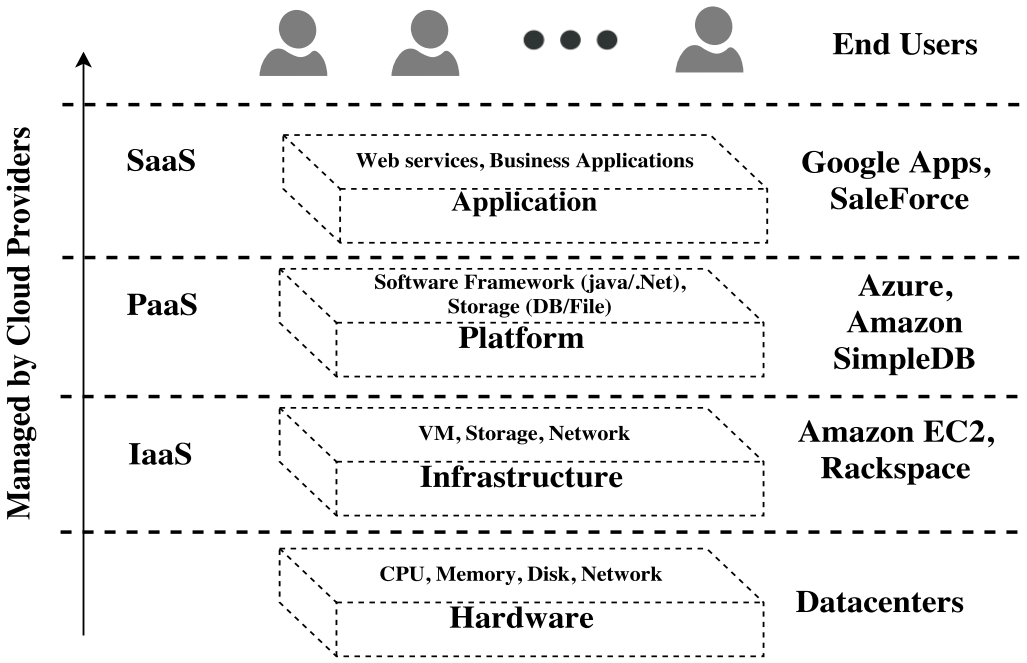

Figure 2: Cloud computing architecture, adapted from [40]

cloud services, and therefore, the associated cost of managing and maintaining the components of this layer represents the CapEx (capital expenditure) and OpEx (operational expenditure) for the cloud providers. The "Infrastructure Layer" mostly deals with on-demand resource provision such as computation, network, storage, etc. while leveraging virtualization technologies [32] to utilize underlying physical resources effectively. Examples of IaaS are Amazon EC2 [4]. The "Platform Layer" provides mostly software frameworks such as Microsoft Azure [13] on top of the infrastructure layer in order to design and deploy cloud services. The "Application Layer" provides cloud applications such as Salesforce [8] or Google App [5] to the end users which is running on top of the platform layer and accessible through the Internet.

2.2.1 Virtualization

Virtualization is a prominent technology which has been used since the 1960s [32]. It allows to run one or more VMs to be co-located on a single PM where the PM is known as the *host* and the VMs are referred to as *guests*. Applications hosted on the VMs are isolated form each other and are not aware of the virtualization of the underlying hardware. A special software layer, called *hypervisor* or *Virtual Machine Monitor (VMM)* [18] is responsible for managing common physical resources among VMs. Unlike emulators, hypervisors use the self-virtualization feature of CPUs instead of emulating them. Most of the recent CPUs from Intel and AMD have incorporated explicit support for a "virtualization extensions" feature which provides hardware acceleration and hence, VMs show improved performance [79].

In general, virtualization can be classified into three categories [68]: *full-virtualization*, *para-virtualization* and *OS-level virtualization* (see Figure 3). *Full virtualization* provides the opportunity to run multiple VMs on top of the virtualized hardware where the VMs are not aware about the hardware virtualization. Each guest VM will operate in a similar way as it is operating directly over the underlying hardware. Examples of hypervisors providing full virtualization are KVM [52], VMware [100], etc. *Paravirtualization* is a virtualization technique where the guest VMs are modified to inform that they are running over virtualized hardware. In this case, the hypervisor acts as a virtualization layer. Guest VMs make the system calls to the hypervisors in order to share common hardware resources. A typical example of a hypervisor supporting paravirtualization is Xen-PV [93, 18]. *OS-level virtualization* works at the OS layer by allowing a single instance of the operating system for multiple isolated partitions. The OS kernel is running on the physical host and provides OS functionality to each partition, where each partition (called container) acts as a real server. FreeBSD jail [58], LXC (Linux Containers [19]) are examples of OS-level virtualization techniques. Unlike VMs, a container can run as a single process where as a VM requires a full OS for running applications, and hence, containers can result in low performance overhead [19] compared to using VMs .

2.2.2 VM Consolidation

Cloud providers typically make extensive use of virtualization or hypervisor technology which enables VM consolidation by accommodating multiple VMs on a single PM where VMs are running multiple applications. VM consolidation makes it possible to share and manage underlying physical resources more effectively. Infrastructure used by the major cloud providers often involve multi-tenant architecture [65] where all the deployed VMs share processor, storage, memory, and other I/O and network resources. The primary objective behind VM consolidation is to share these physical resources with maximal utilization across a large number of tenants in a massively multi-tenant cloud environment, and hence, reduce the CapEx and OpEx for the cloud providers.

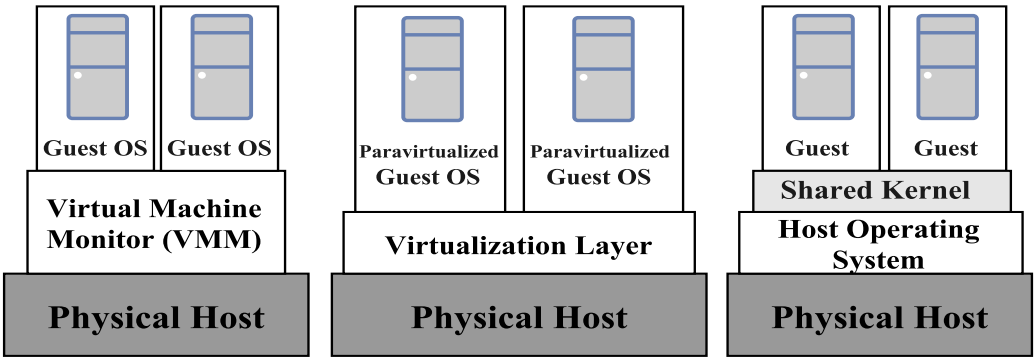

Figure 3: Virtualized infrastructure with Full virtualization, Paravirtualization and OS-level virtualization, adapted from [75]

There are different techniques [69] of VM consolidation based on requirements for optimizing performance and operational costs. In recent years, several research works (extensive surveys are presented in [69, 86]) have been proposed to optimize VM consolidation by a careful allocation of VMs into PMs.

From a broader scope, VM consolidation techniques can be classified into two divisions: *Static and Dynamic VM Conslodation*. In static VM consolidation techniques, cloud providers often have an empty set of resources (PMs, network equipments, etc.), and try to map workload requests (often in the form of VMs) from end users into a proper set of physical resources (PMs). Usually, these techniques work as an initial VM placement strategy, and thus only works when a new workload request arrives without considering re-allocations of the VMs among the PMs. There are several strategies [104] already proposed in this area such as different variations of the first fit algorithms. On the other hand, dynamic VM consolidation makes it possible to re-allocate VMs while considering both current VMs-to-PMs mapping and re-allocation overheads. Dynamic consolidation techniques uaually modify an existing VMs-to-PMs allocation dynamically due to several cases such as changes in customers' requests [89], adapt to critical situations [50], changes in overall system load [87], improve overall decision objectives [69], etc. Such re-allocation mechanisms require one or more actions such as live or cold VM migrations, VM re-sizing, etc.

2.2.3 Live Migration

VM migration is one of the key features offered by a cloud platform. A VM can be migrated from one PM to another by transferring its states such as - CPU, devices, associated memory and storage. Basically, there are two types of VM migration: "Hot (Live) Migration" and "Cold Migration" [30]. In hot or live migration, a VM does not lose its state, therefore, it is not required to shutdown and restart the VM to continue its operation at the destination host. In contrary, cold migrations suspend operating systems and applications on VMs and transfer VMs' states, the associated memory and storage to the destination host and resume VMs' operation at the destination host. The amount of time during which VMs stop executing both at the source and destination host is called VM *downtime*. Today, live migration is supported by most of the common hypervisors such as - Xen, KVM and VMware, but the downtime can vary from few hundred milliseconds to several seconds depending on the hypervisor and workload of the VM.

Three different categories of live migration strategies are discussed in [92]. The outlines for the algorithms of live VM migrations are presented in Figure 4. In the *Precopy* approach [31], VMs transfer their CPU, storage, and memory states before starting their execution at the destination node. This is the most common approach used by most of the common hypervisors such as KVM by default. In general, the *Precopy* approach consists of three steps [33]. In the *Start-up* phase, VMs' memory are transfered form source to destination node over several iterations. But some of the memory pages may change at the source node after transferring to the destination node which require re-transfer

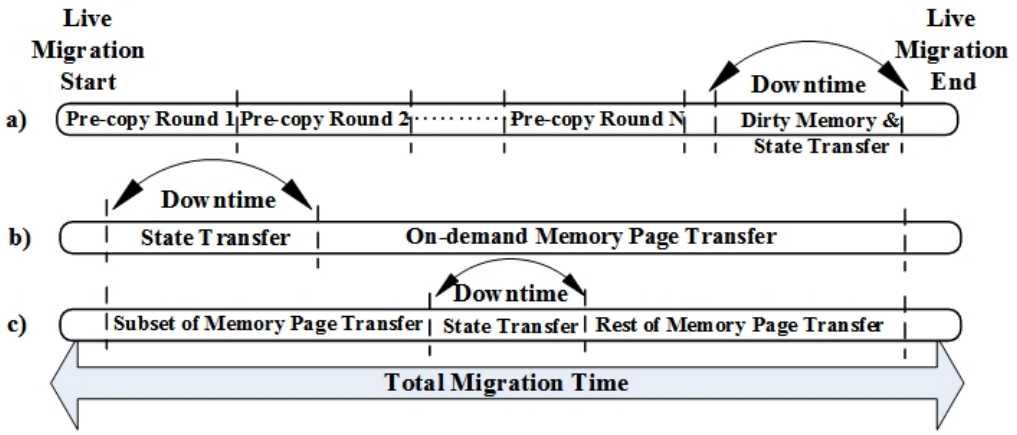

Figure 4: Live migration strategies a) Precopy b) Postcopy c) Hybrid

in order to ensure consistency of the VM. The iterations may run for an infinite amount of time, if memory change rate is higher than the memory transfer rate. Therefore, some conditions are applied in the *Precopy Termination* phase to limit the iterations to a deterministic time. These stop conditions mainly depend on implementations of the hypervisors but can be classified into three main categories. Firstly, the approaches can limit the number of iterations. Secondly, they may limit the amount of transferred memory or finally, they may limit the number of modified pages in the last iterations. In the last phase, called *Stop-and-Copy*, the VMs stop their execution at the source node and transfer CPU and device states and remaining memory pages to the destination node and re-start their operation. In contrary, in the *postcopy* approach [54], at the beginning VMs transfer their CPU states to the destination to start their operation, and afterwards memory pages are transferred from source to destination based on demand. Finally, another strategy called *hybrid approach* [92], combines both *precopy* and *postcopy* approaches. In the *hybrid approach*, at the beginning, the most frequently accessed memory pages are transferred from source to destination and then the migration follows the steps from pure *postcopy approach*.

2.2.4 Elasticity

Elasticity in a cloud-based architecture can be defined as the way of allocating, de-allocating and re-allocating physical resources among the VMs ("VM re-sizing") automatically based on demand. For the dynamic nature of workloads of cloud applications, a cloud platform should be able to allocate and deallocate resources with workload variations, and mitigate peaks as close as possible in unpredictable environments with rapid changes [53].

An elasticity policy can be defined as an approach that decides how to allocate additional resources to an application when workload changes. There are two approaches to achieve elasticity [87] in a cloud environment. In *horizontal elasticity*, scaling is achieved by adding and/or removing the number of allocated VMs for an application. On the contrary, in *vertical elasticity*, scaling

is achieved by allocating and/or deallocating resources to the available VMs for an application. There are two main alternative approaches [45] to implement an elasticity policy for a cloud-based application. The first approach is using a *control theoretic approach* where a decision for resource allocation is based on the current application performance and its projection based on QoS (such as - response time) with given resources. The second approach is called *heuristic, rule-based approach*, where the scaling decision is based on the specification of a set of heuristic rules. For example, when a VM's CPU demand increases more than a fixed threshold either more CPU cores are allocated to the VM or a new VM is spawned up.

2.3 IaaS Cloud Platforms

In order to be considered as IaaS, cloud providers must satisfy some specific set of characteristics. IaaS is a step closer than PaaS or SaaS to the metal in terms of running consumers' applications and responsible for keeping underlying resources up and running, but it's up to consumers' how they deploy and run their applications. According to NIST [71], there are some essential characteristics for IaaS such as physical resources are pooled to serve multiple consumers using a multi-tenant model; allows applications to dynamically scale up or down; provides utility based pricing models; and consumers should be able to access the services through standard computers over the Internet. Currently, there are many IaaS providers such as Amazon's EC2 which are public. On the other hand, there are private IaaS providers [43] such as OpenStack [83], CloudStack, Open Nebula, Eucalyptus. Although there are many benefits of public cloud platforms such as automated deployments, greater flexibility and reliability; generally, they do not provide low-level access to infrastructure resources. For example, consumers cannot migrate their VMs provided by the public IaaS providers. However, private cloud platforms often are open source and give full customization ability which allows academics to conduct cloud computing research at the infrastructure level. As this thesis uses OpenStack cloud platform for several experiments, a detailed description of OpenStack is presented in the following section.

2.3.1 OpenStack Infrastructure

OpenStack is a collection of open source software to build and manage private, public and hybrid cloud platforms. OpenStack falls into the category of IaaS and provides the flexibility to the end users to manage the infrastructure. OpenStack consists of different components and its open nature provides the opportunity to include additional components depending on the requirements. In general the OpenStack community has considered nine different components as the core components.

Nova: It provides the compute service and is considered as the core computing engine behind OpenStack. It is responsible for creating and managing VMs and other instances such as containers for handling computing tasks. *Nova* consists of six different sub components to provide different services. *Nova API*

provides an interface for the end users or system administrators to carry out the management related tasks. *Nova Compute* is responsible for maintaining the life cycles of VMs such as - creating, deleting or moving VMs within the cloud platform. *Nova Conductor* implements a new layer which allows *Nova Compute* to offload its database operations. *Nova Scheduler* handles the coordination between resource requests and resource allocations. *Nova Network* provides the network connectivity for the VMs within OpenStack. *Active Message Queuing Protocol (AMQP)* is used for communication between different services through extensive use of messaging and *MySQL* is used for storing all the metadata.

Neutron: It is the networking component of OpenStack which is a pluggable, scalable and API-driven system for managing network related service such as L3 forwarding, NAT, VPN, edge firewall, etc. for the OpenStack VMs. Unlike *Nova Network*, it is based on software-defined networking (SDN) and offers the opportunity to configure advanced virtual network topologies such as tenant-based network traffic segregation. As a part of creating a VM within OpenStack, *Nova Compute* interacts with *Neutron* through API calls to enable network connection for the VM.

Swift: It offers cloud storage software for objects and files . It uses unique identifiers for referencing files or other pieces of information rather than the location on the storage. Users can store and retrieve information through an API which is ideal for storing unstructured data in a scalable way.

Cinder: It is a block storage component for OpenStack VMs. It uses a more traditional approach than *Swift* to access files based on its location on the disks. In addition, it provides the option to write images to a block storage device for *Compute* to use as a bootable persistent instance.

Keystone: It provides the authentication and role-based access control services for OpenStack. It maps all the users against all the available services in the cloud platform and decides about access over those services. It also provides an OpenStack identity API for the developers.

Glance: It provides the image services within OpenStack. It allows system administrators to use images and snapshots as a template for launching VMs within OpenStack.

Ceilometer: It provides the services related to billing. It keeps track of the users' usage of cloud resources and allows cloud providers to offer billing services based on the usage reporting.

Heat: It is the orchestration component of OpenStack to manage the requirements such as - resources and configurations for deploying services on top of the cloud platform.

Horizon: It provides a web-based interface to the system administrators to manage the cloud platform. For delivering the core support for the OpenStack services it classifies the interface into three dashboards, a "User Dashboard", a "System Dashboard", and a "Settings Dashboard". Further, it provides a set of API abstractions so that developers can work with a consistent set of reusable methods without getting familiar with the APIs of each OpenStack project.

Figure 5 provides an overview of the steps necessary to create a VM within OpenStack. First, a command is received through *Nova API* or OpenStack

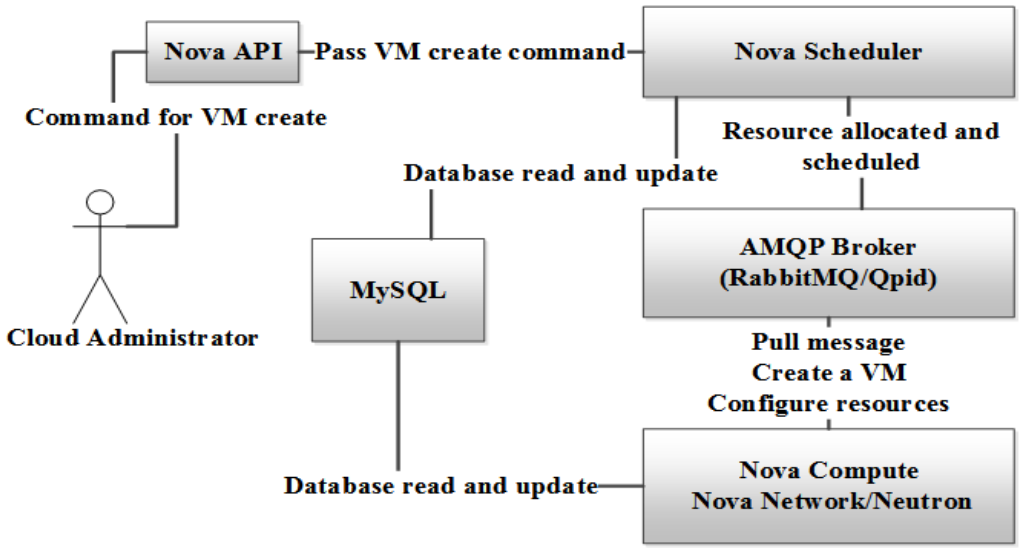

Figure 5: Overview of VM creation within OpenStack cloud platform, adapted from [75]

Dashboard from cloud administrators, then the *Nova Scheduler* decides about resource allocation and placing the VM on a suitable *Compute Node*. For example, the nova-scheduler uses different filers [83] such as RamFilter (to check a host has sufficient RAM available), ComputeFilter (to check a host has sufficient CPU resources), ImagePropertiesFilter (to satisfy any specified architecture, hypervisor type, or virtual machine mode properties) etc. to select the most suitable host for placing a VM. For AMQP broker, either RabbitMQ [98] or Qpid [16] is used for managing the communication between *Nova Scheduler* and *Nova Compute*. Finally, *Nova Compute* launches the VM and *Glance* provides the VM image. Then, either *Nova Network* or *Neutron* is used for setting up network connectivity for the VM. All the metadata regarding the VM is stored in *MySQL* database in order to manage the consistency of the cloud platform. *Keystone* acts as the central entity for checking credentials for all the services.

OpenStack supports different live VM migration methods [83] which can be classified into three categories - "migrations with shared storage known as *Live Migration*", "migrations without shared storage known as *Block Live Migration*", and "migrations with an attached volume (block storage) known as *Volume-backed Live Migration*". In OpenStack, live migration mainly depends on its hypervisor. For example, KVM is the ephemeral hypervisor for OpenStack which supports QEMU [21] to utilize hardware virtualization on different architectures. Libvirt [26] provides an API to manage both KVM and QEMU. The live migration technique is faster than block live migration as for the prior case the hypervisor only transfers the VM's state while for block live migrations the entire virtual disk is required to be transferred. In volume-backed live migration, VMs use block storage rather than ephemeral disk and migration is possible without moving and copying a volume. Rather, it requires a handshake between *Nova* and *Cinder* to attach/detach the volume from

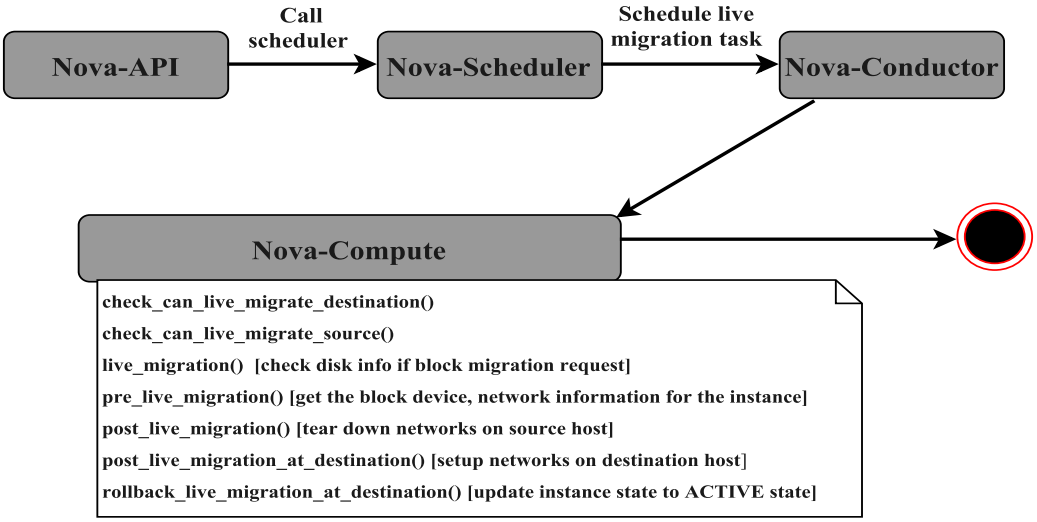

Figure 6: OpenStack VM block live migration workflow

one host to another. The detailed steps for OpenStack block live migration are presented in Figure 6. At the beginning, a request for live migration is provided through *Nova Compute API*. Then, *Nova Scheduler* schedules the task and forwards this to *Nova Conductor*. *Nova Conductor* checks the storage availability and compatibility of source and destination nodes for the migration. *Nova Compute* creates an empty disk and an instance directory and transfers missing files through *Glance* to the destination node. Finally, *Nova Compute* initiates the actual migration on the source node and a libvirt XML file is created on the destination node for storing metadata about the VM. The actual migration process can use the default pre-copy approach or can be configured to use the post-copy or hybrid approach described in Section 2.2.3.

2.4 Cloud Applications and Resource Requirements

Due to the increasing flexibility and cost effectiveness, IaaS platforms are hosting a wide variety of applications with respect to their resource requirements. While some applications such as real-time traffic information on the realm of Intelligent Transport Systems (ITS) [77] has stringent requirements in terms of end-to-end latency, other services such as background analytics and logging operations are less sensitive to latency. On the other hand, some composite services are communication intensive such as Virtual Network Function (VNF) [70], where different network functions are generally deployed over multiple VMs and perform sequential operations across a set of VMs. Consequently, the amount of inter-VM traffic within IaaS platform is growing with rapid pace.

In general, cloud applications can be classified into three main categories [59] depending on their workload types. The first category in this classification is known as “Data-Intensive Workload” where VMs have high data transfers but require low amount of computational resources. For example, a video-on-demand application, generates only a new video streaming process for each new client. In order to maintain the quality of such applications, the network

resources such as network equipments and network paths need to be allocated carefully. The second category, "Computationally-Intensive Workload" focuses mainly on solving complex and computational jobs. These kind of applications have very high demand for computational resources (CPU, memory). For example, numerical simulations to solve real world problems such as surface rendering can ask for huge amounts of computational resources but in general, not require to transfer high volumes of data. The third category is known as "Balanced Workload" [40] where applications require both computational and communication resources. For example, an ITS application [77] requires high computational resources for building graphs with geographical and transportation data, and also transfers noticeable amounts of data in order to provide real time feed to clients.

The performance of an application is not only affected by its hardware and software platforms, but also its resource demands and its sensitivity [78] to the allocated resources. [38] states that understanding workloads and determining appropriate resources is more important than designing scheduling algorithms for placing workloads. The main objective of an IaaS platform is to share resources among different applications, but resource sharing can negatively affect the performance of the co-located applications. Therefore, understanding the accurate sensitivity of an application to different resource types and the intensity of performance degradation due to resource contention, is one of the critical dimension for resource management. Figures 7 and 8 illustrate the relative performance of different applications (selected from Phoronix Test Suites [11]) for application co-location and different resource reservations. For instance, figure 7 highlights the relative performance (compared with isolated run) of eight different applications when they are co-located with another application which is unzipping large files. All the applications are hosted in a VM with 2vCPUs, 2GB of vRAM, and 20GB of vDisk. It is evident that amount of degradation differs across the applications but if 10% degradation is tolerable then its possible to deploy any of the five applications that meet this requirement together with the unzipping application on the same PM.

Further, it is very important to quantify the minimum amount of required

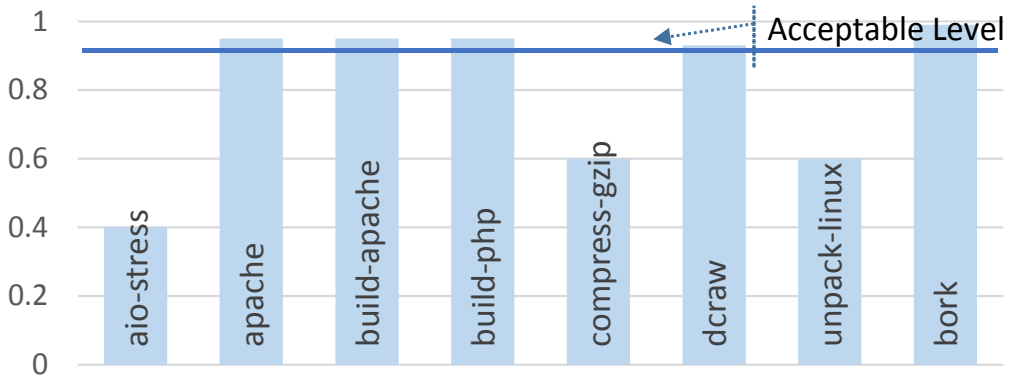

Figure 7: Relative performance of applications due to co-located application, adapted from [78]

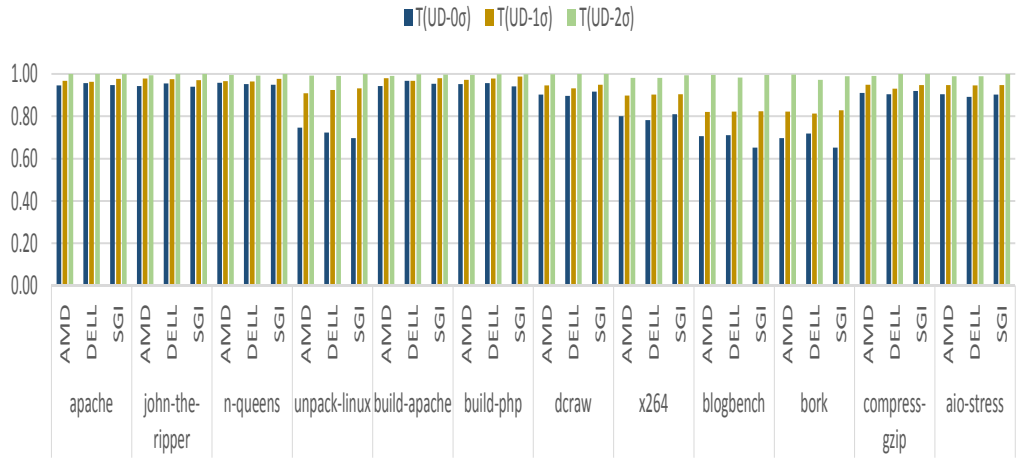

Figure 8: Relative performance of applications for different resource demands, adapted from [94]

resources to achieve predetermined QoS levels for cloud applications. Figure 8 reflects how performance of applications change with three different amount of resource allocation. In this case, the experiments are performed in three different PMs - one AMD (AMD), one Intel Xeon (DELL), and one Intel-i7(CGI) in order to detach the findings of the experiment from its underlying platform. The results clearly reveal that amount of resource reservation has noticeable impact on applications performance, although the amount of degradation is different across applications. For instance, the average throughput for most of the applications is 87% when resources are allocated according to T(UD-0σ) (minimum level of resource allocation), whereas some applications like blogbench has throughput of 67%.

While resource assignment and the relation among different applications requiring same resource are two important parameters for workload management, the associated management decisions are becoming particularly complex [41, 74] by the time-variability of the application demands both within single PM and across the whole infrastructure. For example, in real IaaS platforms workload fluctuates widely as user traffic is higher during the day and drops during the night [34]. This is the most common scenario for the interactive applications such as email, enterprise applications, and social networking. Besides that other applications such as analytic applications also show variations in terms of input size [34]. For instance, figure 9 illustrates the CPU usage of six different applications at Karlstad University (KAU) over a period of one week at the beginning of June 2015. “DC-S8” denotes *Active Directory* for student domain, “Raindance” runs the internal economy system,“Filemaker13” acts as a backend database for two different system, “Titan” represents one student *Mail Server*, “TS-DV” denotes the terminal server for CS domain, and “Passman-a1” represents one part of the authentication system. It is clearly shown that different applications have different CPU demands which vary differently over time. Another important information is highlighted from the graph that all

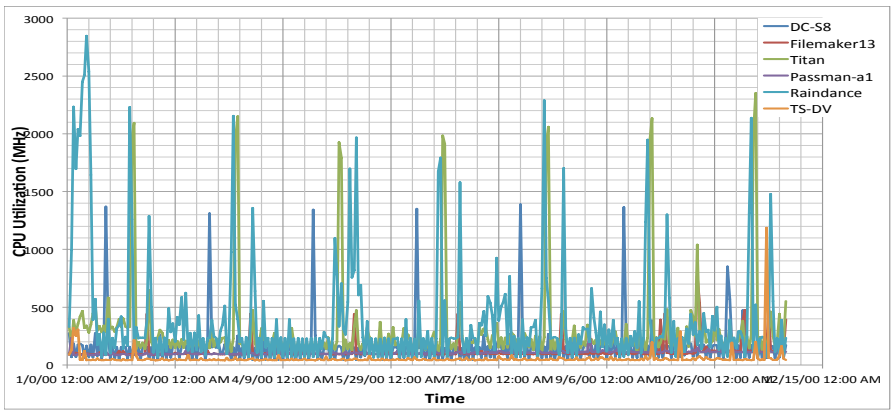

Figure 9: CPU usage of 6 different applications over one week at KAU

the applications are not asking for the highest amount of CPU resources at the same time. This also highlights the fact that although a large number of works [49, 66, 85] have modelled and characterize resource demands, understanding resource variation over time of an application accurately is a very challenging problem.

2.5 Multi-objective Resource Management in Cloud Environment

Resource management is critical for achieving high performance in cloud infrastructures. Different from traditional infrastructures like parallel and distributed systems, resource management in cloud infrastructures involves managing VMs, since all modern IaaS providers make extensive use of virtualization. Ideally, cloud resource management should have three desirable properties [34]: 1) each workload demand should be allocated the proper amount of resources so that it can achieve acceptable performance; 2) each workload need to be encapsulated in VMs and allocated carefully on available PMs to achieve high infrastructure utilization; 3) workloads need to be re-allocated dynamically in order to handle system failure or to cope with varying workload, but re-allocation overhead needs to be minimized in order to reduce negative affects on applications' performance. In general, VM placement/consolidation is inherently a multi-objective problem [44, 97, 103]. Figure 10 presents a list of typical objectives of resource management from the IaaS provider point of view [69]. Of course, not all the objectives are independent from each other, a large number of them are overlapping with each other such as energy-related objectives can be transformed into monetary objectives. However, there are some objectives which are independent and contradict with each other such as minimizing the number of migrations and minimizing the energy consumption of the infrastructure. Therefore, in this situation, if the IaaS provider has

multiple objectives then it is reasonable to balance properly among the different conflicting objectives.

2.5.1 Optimization Model and Uncertainty Programming

Various mathematical approaches have been used to model cloud computing. This is because a mathematical model provides an understanding of the interdependencies involved in cloud computing. These models are suitable for identifying optimal values and equilibria and understanding behaviour. Most of the exisiting approaches formulate cloud environment as optimization problems that aim to analytically identify VMs-to-PMs mappings that would optimize quality of services, performance or operational cost such as energy consumption [81, 22, 99, 80, 64, 46]. In general, analysis of these models are done through numerical evaluations, probably using one of the state-of-the-art optimization solvers such as CPLEX [10], Matlab [7], etc.

The problem of mapping a number of VMs into a set of PMs has its origins in the multi-dimensional, multi-constrained *Bin Packing Problem* [60]. The problem of VMs-to-PMs placements or allocations or re-allocations (known as VM consolidation) are multi-dimensional as the VMs (considered as items) need to be allocated into a number of PMs where the demand requirements and available capacity both include resources from different dimensions. For example, the VMs' may demand CPU (mostly expressed in MHz or in cores), memory (amount in MB or GB), bandwidth (amount Mbps or Gbps) and the PMs are also equipped with a certain amount of CPU, memory and bandwidth. Usually, these VMs-to-PMs allocations are aimed to minimize or maximize some objectives such as minimize the energy consumption and are subject to fulfilling a number of constraints. For example, the demands of all VMs allocated to a PM must not exceed the capacity of the PM (unless overbooking is allowed), which can be treated as a budget constraint. The original bin packing problem can be described as [60]: "*Given a set of bins with a defined capacity and a number of items, each characterized by a well-known size, the objective of the problem is to find the mapping between items and bins which minimizes the number of used bins, without overloading the available capacity of the bins.*"

The basic bin packing problem is NP hard [60] and usually, the solution approaches for this problem are based on Mixed Integer Programming (MIP) or heuristic algorithms. Because of its similar structure, the bin packing problem can be easily extended to the VM consolidation problem. However, for the dynamic consolidation it is necessary to consider carefully the live migration technique of the VMs as the migration process may create additional overhead [102, 88]. A simple formulation for the static VM consolidation for n VMs, k PMs where the resource dimension is 1 (e.g. only memory is considered) is presented below. For instance, $R = \{r_1, r_2, ..., r_n\}$ denotes the VMs' resource demands. PMs' resource capacity are presented as $S = \{s_1, s_2, ..., s_k\}$. In addition, the set of active PMs (PMs that are hosting at least one VM) are presented by a decision variable, p_j. Further, the decision variable a_{ji} is 1, when VM i is allocated to PM j, otherwise it is set to 0. The objective is to minimize the

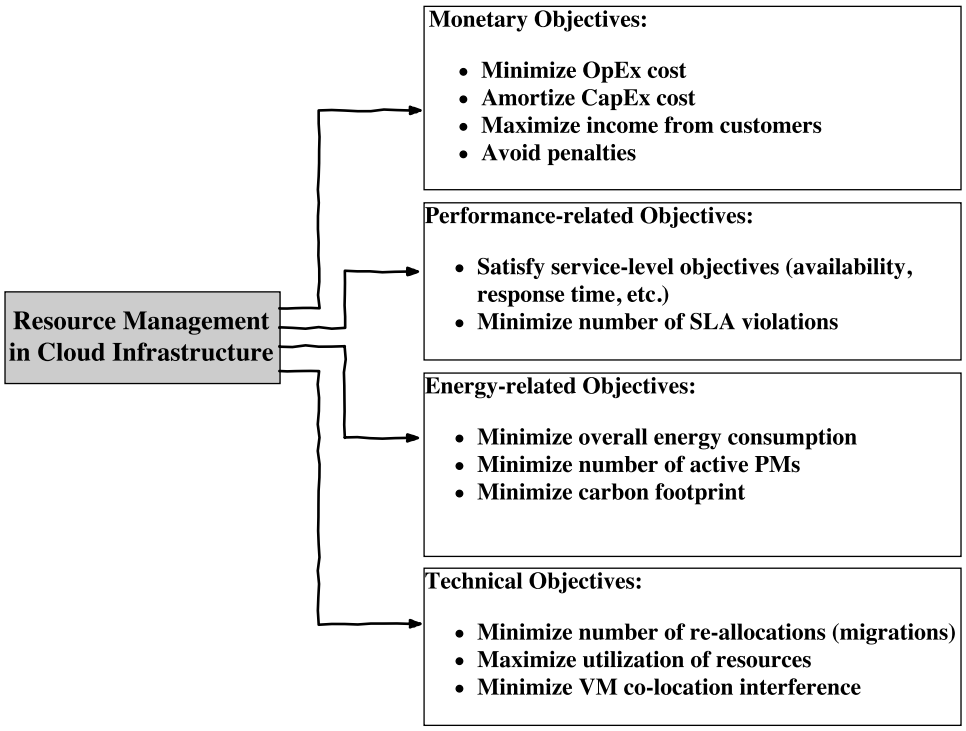

Figure 10: Different objectives for a general VM placement/consolidation, adapted from [69]

number of active PMs and their associated operating cost (presented as o_j). The objective and the most common constraints for this problem can be written as:

$$\begin{aligned} minimize \quad & f = \sum_{j}^{k} o_j \cdot p_j \\ s.t. \quad & \sum_{j}^{k} a_{ji} = 1, \qquad \forall i \\ & \sum_{i}^{n} r_i \cdot a_{ji} \leq s_j \cdot p_j, \qquad \forall j \end{aligned} \tag{1}$$

In the formulation, the first constraint ensures that each VM should be hosted on at most one PM, and the second constraint ensures that all the VMs demand mapped to a PM should not exceed its available resource capacity.

It is discussed before that estimating VMs resource demand is the main challenge in modern IaaS platforms due to diversity of the applications and unpredictable resource demands over time. It is already proven [24] that paying little or no attention to uncertainty in optimization problems may lead to unexpected results. For example, a small deviation in the input parameters may compromise the optimality of the solution or even turn feasible solutions to infeasible ones [24]. The following example can help to illustrate how

neglecting uncertainty on VMs' resource demand can lead to an unfavourable situation for the VM consolidation problem discussed above. For instance, assume that the expected resource (e.g. amount of RAM) demands for two VMs are 2 GB (r_1) and 4 GB (r_2). One PM has the capacity of 6 GB of RAM and hosts both of the VMs. This allocation neglects the fact that VMs' resource demands may vary from the expected or estimated value. Therefore, if one of the VMs' demand increases, the capacity of the PM will not be enough to fulfil both of the VMs' demands unless the other VM reduces its demand. In this situation, either migration will be required to trigger to move away one VM to another PM if there is a available PM; or performance of the VMs is penalized severely if migration is not possible and the performance of the VMs depend on RAM. In summary, it can be said that the proposed allocation plan is not very robust with regard to changes in resource demand.

The above example makes clear that it is not a good idea to neglect resource demand uncertainty, otherwise there is a potential risk that the solution approaches will turn out to be infeasible or shows poor quality when implemented. On the other hand, "Robust Optimization (RO)" [23, 25, 24] is a methodology to improve decision making process in uncertain scenarios by allowing to add uncertainty that is present in the model directly. In recent years, RO has gained a lot more interest than "Stochastic Programming" because of its accessibility, computational tractability and not assuming knowledge of the distribution of the uncertainty [24]. Specially, RO deals with the fact that the formulation contains a defined *uncertainty set* to reflect risk aversion and to identify deviation of the coefficients against which the solution approach need to be protected against i.e. the uncertainty set cuts off all the feasible solutions that may become infeasible due to deviation present in the uncertainty set. Within different possible uncertainty sets [51], the cardinality constraint uncertainty set, defines a family of polyhedral uncertainty sets [51], is practically relevant as it presents a budget of uncertainty i.e. the maximum number of parameters that are allowed to deviate from their expected value (mostly denoted as "nominal value") known as "so-called row-wise uncertainty" or "Γ-robustness".

From the previous discussion about workload characterization in IaaS platforms, it is noticeable that it is very unlikely that all the VMs will deviate to their maximum at the same time. Therefore, for the VM consolidation problem, a RO model should define the uncertainty sets carefully so that it does not increase the operational cost unnecessarily while guaranteeing a satisfying level of performance for the hosted VMs. For instance, for the VM consolidation problem presented above, if the resource demand of the VMs (r_i) are not known precisely, only their nominal value ($\bar{r}_i$) and the maximum possible deviation ($\Delta r_i \geq 0$) are known i.e. $r_i \in [\bar{r}_i - \Delta r_i, \bar{r}_i + \Delta r_i]$, then the second constraint can be written as follows.

$$\sum_{i}^{n} \bar{r}_i \cdot a_{ji} + DEV_j(\Gamma, a) \leq s_j \cdot p_j, \qquad \forall j \tag{2}$$

Here, the additional variable, $DEV_j(\Gamma, a)$ presents the worst deviation that the

left side of the constraint may experience under Γ-robustness for an allocation vector a, when at most Γ coefficients deviate from their nominal value $\bar{r}_i$.

Therefore, now the VMs-to-PMs allocation scheme will be feasible even when at most Γ resource demands, deviate from their nominal value $\bar{r}_i$ to $+\Delta r_i$ (the most positive deviation indeed entails the highest increase in a resource demand). Clearly, ensuring protection against resource demand variation through hard constraints increases the operational cost for the IaaS providers (as now more PMs are required to host the VMs, hence increases the overall energy consumption). This phenomena is known as "Price of Robustness" [24] which defines the increase in the cost of a robust solution compared to the cost of the deterministic solution. Such price indicates the characteristics of the uncertainty set i.e. uncertainty sets that present higher risk aversion consider more unlikely and severe deviations and provide higher protection, but also lead to higher price of robustness; On the other hand, uncertainty sets that present risky attitudes usually neglects unexpected deviations and provide lower protection, but also decrease the price of robustness.

2.5.2 Meta-heuristic based Heuristics

Formalizing the VM consolidation problem as an optimization problem and solving it through a state-of-art optimization solver is very challenging because the large number of integer variables in the model lead to very high computational complexity. Therefore, finding an exact/optimal VMs-to-PMs allocation within a reasonable amount of time for medium to large sized cloud infrastructures is not possible, as [95] presents how the search time of an exact algorithm increases with the size of the problem. As the VM consolidation problem is NP-hard [86] i.e. it requires exponential time (unless P=NP) to be solved optimally, therefore, metaheuristics are an interesting alternative to solve this problem [95]. Unlike exact methods, metaheuristics allow to tackle large-size problem instances by delivering satisfactory solutions within a reasonable amount of time. A metaheuristic is a high-level problem-independent framework that can be used as a guiding strategy in designing underlying heuristics to solve specific optimization problems [95]. Two important characteristics of metaheuristics are intensification and diversification [105]. Intensification intends to search better solutions around the current best solutions and select the best candidates or solutions. Diversification makes sure that the algorithm can explore the search space more efficiently, often by applying randomization. As the underlying foundations of different metaheuristics vary significantly, two fundamental classes of metaheuristics can be distinguished [48]. Single-solution metaheuristics [95] (e.g. Local search, Simulated Annealing, Tabu Search, etc.) iteratively make small changes to a single solution and try to find good quality solutions in each step by means of a neighbourhood search. Population-based metaheuristics [95] (e.g. Genetic Algorithm, Swarm Optimization, etc.) iteratively combine solutions into new ones. However, these classes are not mutually exclusive and many metaheuristic algorithms combine ideas from different classes, or involve constructive metaheuristics to construct initial solution from their constituting elements rather than improving complete solutions. This is achieved by adding

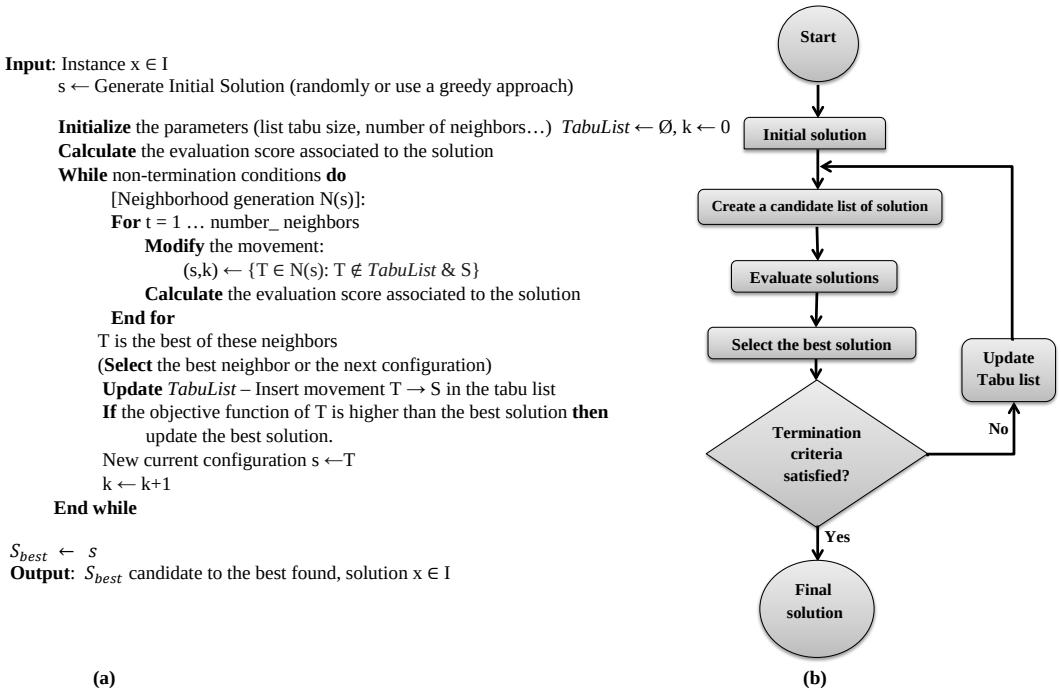

Figure 11: Basic TS procedure a)Pseudocode b)Flowchart, adapted from [39]

one element at a time to a partial solution. Constructive metaheuristics are often adaptations of greedy algorithms that add the best possible element at each iteration. Although the choice of metaheuristic algorithm depends on the underlying problems, Tabu Search (TS) has gained interests due to its performance with regard to solution time [63] and guaranteed stable optimization results for multiple iterations [15].

The TS algorithm processes only one solution in each iteration and explores the neighbourhood of the current solution by flexible use of memory [47]. In each iteration, TS explores the entire neighbourhood of the current solution and selects the best solution by using an evaluation function. The evaluation function selects the move in the neighbourhood of the current solution that produces the most suitable move or least deterioration in the objective function. In order to prohibit the algorithm from cycling around the local optimum, it manages a memory of the recent solutions or movements known as *tabu list*. Figure 11 illustrates the pseudocode and the flow chart of the basic TS algorithm. Basically, TS consists of seven core elements: 1) the move, 2) the neighbourhood, 3) initial solution, 4) a search strategy, 5) a memory, 6) an evaluation function (objective function) and 7) termination criteria.

For instance, in the dynamic VM consolidation problem, TS would start with a initial VMs-to-PMs allocation. The initial allocation may be random or follows a greedy approach like first fit. Within an iteration, in every step one of the VMs are selected and migrated to a new PM (move), and new allocations (neighbourhood) are created. The objective function is evaluated for each neighbour (e.g. the objective is to minimize the energy consumption) and the

VM allocation associated with the best score is selected. Then, the migrations are added to the tabu list. The iteration is continuing until the termination criteria is satisfied. Different termination criteria can be used, for example a time limit or maximum number of iterations. However, some useful moves (which help to achieve better objective) may be rejected because of being in the tabu list. Therefore, some movements need to be re-considered although they are enlisted in the tabu list. These movements are known as *aspired moves*, and the criteria that triggers these movements are called *aspiration criteria* (the most common aspiration criteria are to accept moves that can give better evaluation score).

It is discussed before that robustness is one of the most important characteristics that need to be considered carefully during searching for a solution. Therefore, the basic TS algorithm needs to be modified so that in each iteration, it searches for solutions which not only ensure good quality but also present robustness with regard to input parameter changes. A solution is called *quality robust* [91] if the quality of the solution does not deteriorate with changes in input. The main advantage of making tabu search robust is that it only requires modification in the evaluation function rather than changes in any TS-specific features. For instance, the evaluation function $f(x^{'})$ may be changed to $f_r(x^{'})$ to support robustness based on following basic principles [91].

Principle 1: "For every solution, some noise may be added (that reflects the expected changes to the input) before calculating its evaluation score. Thus, the robust TS evaluates $x^* = x^{'} + \delta$ instead of $x^{'}$, where x^* can be a derived solution. Derived solutions may or may not belong to the original solution space."

Principle 2: "The new robust evaluation function, denoted as $f_r(x^{'})$ is responsible for evaluating several derived solutions and combined into a single function value rather than evaluating a single point."

Based on these principles, a robust evaluation function [91] may be written as $f_r(x^{'}) = \frac{1}{n} \sum_{i=1}^{n} w_i f(x^{'} + \delta_i)$, where n presents the total number of derived solutions that are evaluated and w_i indicates the weighting factor. Here, $f(x^{'} + \delta_i)$ presents the function for the derived solution, $x^{'} + \delta_i$ at point i. Therefore, by evaluating several derived solutions and combing these evaluations into a robust evaluation function, a TS algorithm can offer the robustness feature to handle perturbation in the input data effectively.

RESEARCH PROBLEMS AND PROPOSED APPROACHES

The key challenge addressed in this thesis is to design and implement cost-efficient and application-aware resource management techniques for virtualized cloud infrastructures. While inefficiencies exist across the hardware and software stack, application performance, infrastructure utilization, and high energy cost are to a large extent determined by the adopted resource management techniques; in particular, the system that orchestrates resource allocation and maps VMs-to-PMs in a virtualized environment. To this end, the contributions of this thesis focus on decreasing energy cost for the cloud infrastructures by improved resource allocation decisions, while guaranteeing that each VM (application) satisfies its performance requirements. This is the overarching research question of this thesis. It is broken down into three more detailed questions. Further, these questions are divided into smaller challenges for different articles that this thesis is comprised of.

How to advance experimental studies involving different cloud computing architectures and provide reconfigurable experimental features for the cloud environment?

Problem: Cloud computing is primarily commercially-driven and hence, system level researchers who are designing techniques and infrastructures to support cloud find it difficult to obtain low level access to cloud infrastructure resources. In addition, application level researchers also need to control the deployment and resource consumption of hosted services across a distributed cloud environment to design and deploy new application models. Therefore, a cloud testbed is required to be designed or deployed to support research into design, provision and management of cloud infrastructure and also into the design and deployment of large scale services that use such an environment.

Approaches: Paper I aims to study the fundamental questions associated with a virtualized cloud environment: 1) how much overhead does virtualization impose on the performance of the applications hosted on IaaS environments? 2) what are the most relevant factors that affect the interactions between virtual and physical resource allocations? The goal of this work is to deploy applications from early prototype to investigate the level of performance degradation imposed by the virtualization that are realistic enough to understand the necessary system level support for the hosted applications. Further, a nested virtualization technique is employed in order to highlight the difference in applications' performance with increasing level of virtualization. Finally, the experiments provide a better understanding of designing, deploying and configuring an OpenStack-based private cloud environment.

The aim of Paper II is to investigate the question: how to give researchers the potential to rapidly build their own private infrastructure as a representative of real world cloud DCs to evaluate their algorithms, policies, applications and technologies without excessive hardware investment? The paper presents a programmable cloud computing research testbed, called OpenStackEmu which encompasses an OpenStack-based private cloud composed of a set of compute nodes, networking hardware and emulator CORE (Common Open Research Emulator) [14] and the software, a SDN based controller [82], to innovate both at and above the infrastructure layer. OpenStackEmu provides a repeatable and controllable environment that supports experiments with different topologies such as distributed infrastructure topology and study the behaviour of OpenStack VMs as well as services hosted on those VMs under different background loads and SDN routing policies. The paper provides insights into the practical aspects regarding the setup and implementation of a cloud testbed utilizing open source solutions and also elaborates practical cases of performing research by experimentation in the cloud infrastructures.

How to schedule and deploy low latency applications for the cloud?

Problem: Many cloud services are commonly soft real time in nature and service latency is usually one of their major requirement. In general, a late response does not cause any catastrophic consequences for these applications, but the quality of the applications are affected negatively and may result in revenue losses for the cloud providers (stated by Amazon [55], 100 ms delay in web page loading time may drop 1% of their sales). Therefore, managing effectively low latency cloud services with the goal of maximizing a cloud provider's profit is crucial for the sustainable development of cloud computing. This is particularly challenging since these applications may be composed of multiple components with complex interactions among them.

Approaches: Paper III conducts extensive measurements and tries to answer the question: how a low latency application performs when deployed in a cloud infrastructure? The paper selects a core application from the ITS domain as its workload is typically very demanding in terms of CPU, memory and network I/O and runs on OpenStack VMs. The paper provides insights into resource sharing techniques of the cloud platform and scalability of the overall system under real-life workload scenarios. Further, it identifies the challenges application designers face in architecting scalable and latency sensitive applications for cloud environment while meeting their constraints such as cost, delays, etc.

Paper IV addresses the problem of adverse impacts of live VM migrations on the application performance, specially, migrating a VM when it is serving a number of interactive clients, and focuses on how to utilize available network resources effectively to improve live migrations' performance. The paper uses the ITS application proposed in Paper III and discusses different innovative

approaches to provide QoS for these types of low latency applications. The testbed is designed to model a typical cloud deployment scenario where PMs are connected via multiple network paths for separating management and data traffic. The paper proposes different network-centric approaches to achieve optimal trade-offs between resource utilization and QoS which can be effective for both cloud providers and application performance. Further, extensive experimental evaluations in a private cloud testbed under a large number of realistic scenarios showcase better ways of utilizing cloud resources to reduce service delay for the soft real time applications.

Paper V tackles the research question: how to schedule dynamically applications with different QoS requirements (such as latency) in multiple geographically distributed cloud infrastructures? In particular, a request from an enterprise customer is mapped to a virtual network (VN) and allocated, which requires both computational (CPU) and communication-related (bandwidth, latency) resources. The paper proposes an allocation scheme over multiple time periods with the goal of minimizing operational cost. More specifically, the work presents a novel MILP formulation which aims at solving dynamic traffic engineering problems by flexibly splitting different demand requests across multiple infrastructures based on the time varying location dependent operating costs. Further, comprehensive evaluations based on real-world traces for DC locations, operational costs, demand patterns and resource provision costs help to gain insight in the relationship among requests arrival rates, applications QoS requirements and operating costs of the cloud providers.

How to model uncertainty phenomena in cloud infrastructures resource provisioning?

Problem: There are different types of uncertainties associated with cloud environments [96, 78] such as performance variation due to applications interference, time varying resource demand, inaccuracy of applications run time estimation, variation of processing times and data transmission, workload uncertainty, and other phenomena. Uncertainty is one of the main obstacles in cloud infrastructures bringing additional challenges to the cloud providers as well as to the end users. Most of the available cloud perfromance models do not satisfactorily capture uncertainty and dynamic performance changes related to shared infrastructures [96]. Therefore, further research is needed to understand the consequences of uncertainty and design novel resource management strategies to handle uncertainty in an effective way. In particular, the effect of uncertainty in resource and service provision strategies and corresponding operational cost for the cloud providers; and applications co-allocation models and corresponding performance interference need to be investigated thoroughly.

Approaches: Paper VI addresses the problem of uncertain or unpredictable performance of applications due to their co-location with other applications on the same underlying hardware. The resource sharing nature of virtualization technology can lead to several scenarios ranging from smooth

co-existence to catastrophic situations, where performance of one or more applications degrade significantly due to contention for the same resource. Therefore, it becomes imperative to detect and quantify the amount of performance degradation to maintain a desired level of performance. The paper tries to solve the following research question: how to mitigate the impact of interference without increasing energy consumption for the cloud environments? A novel dynamic VM consolidation approach is proposed while considering resource sensitivity values of the applications. The work uses quantitative measures that reflect how much a VM (application) is sensitive to its requested resources such as CPU, memory, disk etc. and uses this information to achieve better VM allocation decisions with respect to interference for a given limit on maximum energy consumption. Further, our evaluation covers a wide range of scenarios including different overbooking ratios of physical resources and provides insights into how considering sensitivity information of the applications can lead to significant performance gain for cloud infrastructures.

There are other sources of uncertainty which are common in modern cloud infrastructures but crucial to understand in order to manage resources in an efficient way. For example, cloud providers may not have exact knowledge about the system and applications such as servers power consumption, quantity of the resources that need to allocated, prediction of resource overhead due to migrating VMs from one PM to another. Therefore, paper VII tries to answer the following research questions: 1) how to analyse and model cloud behaviour under uncertainties and specific constraints? 2) how to mitigate the impact of uncertainty on the performance, reliability and robustness of the cloud environment? A multi-objective mathematical model based on the theory of robust optimization is formulated to solve the VM consolidation problem under non-deterministic resource demand and migration overhead. Further, the article presents an extensive study on different uncertain parameters which illustrates the trade-off between cloud infrastructures' energy cost and corresponding SLA violations for the hosted applications.

The VM consolidation problem falls in the category of NP-hard combinatorial optimization problems and therefore, its worst case run time is exponential with respect to the input size, and hence, solving a large-scale instance takes too long. Therefore, a more scalable approach is required to obtain solutions that can balance between quality and time. Paper VIII addresses this issue by designing and implementing a novel Tabu Search based robust algorithm. The proposed algorithm can effectively generate VM placement plans for uncertain resource demands in a reasonable amount of time, even for large size instances. Further, the algorithm can produce VM placement decisions that can cope with sudden workload fluctuations in real-time to ensure required SLA.

RESEARCH METHODOLOGY AND RELATED ISSUES

In general, scientific methods are a common and useful way of describing science. Computer science is younger than most academic disciplines and can be divided into "Theoretical Computer Science" and "Experimental Computer Science" [36]. It can be considered as empirical science where the scientists observe phenomena, state hypothesis, and test it or prove it. In most cases, scientific methods refer to an iterative cycle of literature review, problem formulation, hypothesis building or description, verification and analysis. The methods used for hypothesis verification include - real-world measurement, simulation, emulation or analytical methods. When using analytical methods, a mathematical model is used to represent a system. Computer simulations involve more detailed features of the underlying system but often based on many assumptions and artificial modeling in order to reach a certain realistic degree. Real-world experiments have the lowest level of abstraction, but it is difficult to design the setup and hard to control the environmental factors. Due to lack of control over the experimental environment, this method is often time consuming to verify a hypothesis. As both simulation and real-world experiments have shortcomings, a hybrid method called emulation is becoming more common. In this case, some components of the experimental setup are abstracted and some components run within a real environment. It is possible to combine any of these approaches, and as a best practice [56], it is typically worth to use multiple methods to validate hypothesis.

In this thesis all three methods - analytical models, simulations and real experiments are applied. Apart from being less demanding on hardware and software investment, mathematical modelling and analysis are also attractive for gaining insight into interdependencies involved in cloud computing. It is particularly suitable for identifying optimal values and equilibria and predicting behaviour. For instance, Paper V presents a mathematical model to investigate the dynamic resource allocation scheme for geo-graphically distributed cloud environment, because getting full control over three geographically distributed real infrastructures in order to perform experiments is extremely difficult and expensive. Further, the mathematical model proposed in Paper VI helps to reveal the relationship between a given allocation of resources to an application and resulting QoS. Furthermore, in Paper VII, a mathematical model is used to understand the impact of uncertainty on cloud providers operational cost and applications' performance level. Simulation is often an alternative method in the area of cloud computing [29] when system properties make analytical modelling difficult. Paper VIII employs the simulation method in order to investigate the performance of the proposed algorithm. In this case, simulation allows to determine the correctness and efficiency of the proposed algorithm by comparing alternative designs, and investigating the impacts of different

design parameters. For instance, the algorithm is investigated under 7 different scenarios with 10 to 100 repetitions, and some design parameters have a range of possible values; all together, this leads to large number of combinations which may require lengthy experiments over expensive testbeds. In most cases this length or expense is not feasible, and so fewer scenarios are examined, and/or fewer replications are made. Evaluating fewer scenarios or making fewer replications might result in potentially leading to wrong insights about performance of the algorithm. To overcome these challenges, simulation gives the benefit of repeating experiments and reproducing results.

Parts of this thesis also uses a testbed to run experiments in a controlled environment. We take synthetic traffic as input to mimic large numbers of complex use case scenarios and different traffic characteristics. To evaluate all the experimental scenarios, a physical testbed is deployed for hosting a cloud platform. For the cloud platform, OpenStack [84] is used (Paper I - IV). OpenStack is selected because it continues to demonstrate extensive growth in development and participation of a large number of vendors. Further, OpenStack is designed to provide massive scalability by loose coupling among different components.

However, there are a large number of challenges associated with physical testbed deployment such as OpenStack and experiments on it. First, although OpenStack is an open source software and does not charge any licensing fee, frequent changes to the source code make it hard to maintain. One of the major problems is that the latest code is always unstable and may contain some critical bugs which are difficult to identify before deployment. Additionally, a configuration management issue can cause problems at several points in the cloud platform which requires troubleshooting. This troubleshooting can be a very difficult and time consuming task for a large scale complex software like OpenStack. Second, the architecture of OpenStack is modular and different components work together as mentioned in Section 2, and hence, the deployment is challenging as the underlying hardware must be designed at the beginning. It requires a sophisticated plan to deal with everything from cabling to designing networks, storage, etc. based on the use cases and workload types. Furthermore, a large number of OpenStack deployment tools such as - Juju-MAAS (Metal as a Service) [67], DevStack [42], Fuel [90], TripleO [61], Puppet-openstack [62], etc. impose difficulty regarding product selection for the testbed setup. In addition, these automation tools are often associated with different technologies which add extra complication for most of the cases. Third, cross-domain technical expertise such as - IP networking, hypervisor resource management, storage architectures, source code management, security, distributed application architecture, etc. is mandatory to deploy and maintain OpenStack as it spans compute, network, storage, security and other domains.

Additional challenges are related with ranges of involved technologies, rapid progress on the design and development, diverse implementation teams for different components. Therefore, there is a steep learning curve to deploying and working with OpenStack. Finally, there are challenges involved in experimental evaluation on a cloud platform to provide the flexibility of repeating

the experiments and reproducing results. For some cases, experimental results vary due to different hidden factors of OpenStack such as - underlying hardware, hypervisor types and related parameters, OS parameter settings, and specially, the communication overhead among different components of OpenStack. Therefore, discovering, and possibly reducing these factors in order to improve evaluation methodologies is also a complex and time consuming task. Paper I-IV present the hardware and software configurations and the tuning of the parameters for the experiments in order to achieve repeatability. For instance, in Paper I, we enabled "nested virtualiztion" on the physical host to maximize the performance of VMs, and loaded the VHostNet kernel module to improve network performance of the VMs. Further in Paper II, we used "TUN/TAP" interfaces in order to allow user space networking i.e. allow user space programs to control raw network traffic. Again, in Paper III, we tuned some parameters to enable parallel connections for a large number of subscribers such as adjusted hard and soft limits for the file descriptors, increased the value for the maximum number of requests queued to a listen socket, etc. Further, in paper IV, we investigated different factors that can affect the VM migration performance such as - hypervisor specific parameters that can be used for reserving bandwidth and setting tolerable downtime for the VMs, performance impacts due to software switching (OpenvSwitch), data transfer rates of the attached storage, etc.

SUMMARY OF APPENDED PAPERS

Paper I : Deploying OpenStack: Virtual Infrastructure or Dedicated Hardware

With increasing adoption of cloud computing in different sectors, virtualization has become the core technology for offering economical on-demand computational resources. Additionally, nested virtualization or two-level virtualization (a nested hypervisor inside a VM) gives full control to the end users over the VMs so that they can migrate their VMs provided by the IaaS providers. However, virtualization can degrade the performance of on-top applications due to different factors, such as - CPU sharing, hypervisor scheduling, etc. This paper investigates how the underlying virtualization affects the performance of a private cloud environment. In particular, we have conducted some basic experiments to check the performance of CPU, network and storage throughput while shifting the underlying hardware from physical servers to a virtualized setup for OpenStack deployment. The results suggest that shifting from one-level virtualization to two-level virtualization may significantly degrade the performance of CPU, disk and network. In the worst case, the performance of the resources can degrade up to seven times. Moreover, we shed some light on practical issues related to OpenStack deployment such as installation of different OpenStack components, balance between hardware amount and deployment requirements for a private cloud setup, hypervisor performance tuning parameters and their impacts.

Paper II : OpenStackEmu - A Cloud Testbed Combining Network Emulation with OpenStack and SDN

OpenStackEmu is a programmable cloud computing research platform which integrates physical Openstack deployment with the large-scale and distributed Network Emulator CORE, a SDN controller and the Datacenter Traffic Generator (DCT^{2}Gen) [101]. The aim of OpenStackEmu is to support researchers to controlled cloud infrastructure and networking experiments and study the effect of new algorithms and protocols on the performance of real cloud applications that run inside the VMs. The SDN controller and the compute nodes of the OpenStack infrastructure are connected with the CORE emulator node via TUN/TAP interfaces where both control e.g. migration traffic and data e.g. VM-to-VM traffic are pushed over a customizable network topology such as fat tree. Further, a number of use case scenarios are presented in order to demonstrate how researchers can pursue their experiments in a realistic environment under a range of realistic traffic patterns. For instance, one scenario is to investigate how VM migration time in OpenStack is affected when the level of network congestion changes due to other cross-traffic.

Paper III : Mobile Publish/Subscribe System for Intelligent Transport Systems over a Cloud Environment

This paper discusses the design and implementation of a real-time low latency application from the ITS domain - *Real-time Public Transit Tracking* while taking into consideration two recent technologies - Publish/Subscribe system and Cloud Computing. During the design and implementation of the application, two major challenges are addressed - continuous handling of large volumes of real-time data, and automated context aware push-based communication for a large number of mobile consumers. This paper makes several contributions. First, the implementation of a soft real-time application which has the potential to be deployed in a cloud environment. Second, a detailed evaluation of the application for different use case scenarios and diverse workloads is presented in order to reveal several insights such as analysing and understanding the performance of such applications. Finally, a practical insight of resource scheduling strategy used in OpenStack for equal and unequal workload on guest VMs while keeping the same aggregated workload on the physical machine, is analyzed as OpenStack is used as the cloud platform provider.

Paper IV : Network Centric Performance Improvement for Live VM Migration

Live migration is an important feature offered by most of the cloud platforms in order to optimize resource utilization while ensuring the service level agreements for the hosted applications. However, the default live migration strategy (pre-copy) in OpenStack suffers from large VM downtime and migration time, which introduces unpredictible latency for the applications running inside the VMs. In this paper, different network centric techniques are introduced to improve the performance of the block live migration technique of OpenStack. First of all, we propose to take advantage of utilizing multiple network paths instead of a single path during sending migration traffic through data center networks. Secondly, evaluation of queue management schemes, such as Hierarchical Token Bucket (HTB) [17] and Fair Queuing with Controlled Delay (FQ_CoDEL) [57], for the latency-sensitive application presented in paper III explores different ways of providing QoS guarantee for the cloud services. Moreover, an extensive evaluation of the block live migration technique of OpenStack for different workload patterns and traffic characteristics is presented.

Paper V : Cost-Efficient Resource scheduling with QoS Constraints for Geo-Distributed Data Centers

This paper aims to develop an advanced scheduling approach for virtualized reservation-oriented services in multiple geographically distributed cloud infrastructures. Due to the spatial diversity of cloud infrastructures, the cost of using

different infrastructures varies over time. We factor in the operational cost variation for the infrastructures, depending on their power consumption cost, carbon taxes, and bandwidth cost. Paper V presents a unifying optimization framework for minimizing time average operational cost of geo-distributed IaaS, while guaranteeing the SLA requirements (such as latency bounds) for the requests. The MILP formulation proposed in this paper offers the flexibility of allocating requests, arriving at a particular review point to a available infrastructure. Further, the MILP model considers penalty costs by associating it with blocked requests in order to highlight the potential loss in revenue. Finally, a comprehensive study is presented to provide an insight about how different classes of Virtual Network (VN) customers are affected in terms of resource requirements and corresponding request arrival rates.

Paper VI : Optimizing Virtual Machine Consolidation in Virtualized Datacenters Using Resource Sensitivity

In virtualized datacenters (vDCs), dynamic consolidation of VMs is used as the most common technique to achieve both energy-efficiency and load balancing among different PMs. Although live migrations of VMs can greatly contribute to realize dynamic consolidation, often uncontrolled migrations can create high contention for resources among co-located VMs and lead to non-negligible performance degradation for the applications running inside the VMs. Based on the observation of performance interference, we aim at designing a VM consolidation approach while making migration decisions based on applications' sensitivity to their demanded resources. We propose a novel mathematical model for the VM consolidation problem in order to reach a balance between energy consumption of a vDC and performance of the hosted applications. The proposed approach is validated through an extensive numerical evaluation using five familiar benchmarks which are different in terms of resource consumption. The results show that our approach can reduce performance degradation of applications by 9 - 12%, on average with allocation fairness (in terms of throughput among applications) even for higher level of overbooking.

Paper VII : Robust Optimization for Energy-Efficient Virtual Machine Consolidation in Modern Datacenters

This paper investigates the role of uncertainty and associated challenges in the context of designing dynamic resource allocation techniques in modern cloud infrastructures. We propose a novel energy efficient VM consolidation optimization model based on the theory of robust optimization and the concept of Γ-robustness, which includes also the overbooking of resources. Our model takes into account uncertainty of several input parameters such as VM resource demands, migration related overhead or the power consumption model of the deployed PMs. Further, our extensive numerical evaluation illustrates the set of trade-offs a decision maker can deal with: by calculating more robust solutions,

the resulting VM consolidation scheme leads to higher energy consumption (called "Price of robustness"), while being more risky the cloud provider can save more energy at the expense of potentially more unsatisfied users (i.e. higher probability of SLA violation). Previous models mostly assume perfect knowledge of input parameters to the model. As a result, their solutions may be highly infeasible in practice, while the solutions that our model provide are of high relevance to practical deployments where it is very difficult to estimate model parameters precisely.

Paper VIII : A Robust Tabu Search Heuristic for VM Consolidation under Demand Uncertainty in Virtualized Datacenters

From the algorithmic point of view, the dynamic robust VM consolidation problem proposed in Paper VII requires a solution to come up with a number of VM migration decisions among all possible set of migrations within the cloud infrastructure that would minimize both energy consumption and the number of migrations for uncertain resource demands. This makes the problem search space increase exponentially with the increase in the number of VMs and PMs. Thus, due to high computational complexity of the MILP-based exact method, Paper VIII proposes a novel heuristic based on Tabu Search by applying robust optimization theory and adapting Γ-robustness. The main idea of this heuristic is to navigate the search space efficiently within polynomial time while trying to achieve an energy-efficient VM allocation plan and to manage resources in response to changing demand patterns dynamically. Further, an extensive comparison with the exact model under different scenarios highlights the suitability of the heuristic in the context of large-scale cloud infrastructures. Most importantly, the proposed approach for robustness is not limited to Tabu Search, rather it can be easily applied to any local search algorithms such as Simulated Annealing, Hill climbing, etc.

SUMMARY AND OUTLOOK

The rapidly growing adoption rate of cloud infrastructures for deploying diverse applications has accelerated the number and size of cloud infrastructures. These infrastructures incur very high investment and operating cost for the computational resources as well as for the associated power consumption. Much of the progress in infrastructures efficiency over the past five years has occurred in the area of facility and equipment efficiency. However, little progress has been achieved in server operation efficiency, particularly in terms of server utilization. Most significantly, servers are being used very inefficiently consuming power 24/7 due to the their peak provisioning configuration while doing little work most of the time. Further, the progress remains uneven across development of innovative management of cloud resources so that applications can maintain their performance requirement dynamically with demand variation over time, while minimizing the power consumption.

Major research goals of this thesis are to devise scheduling and dynamic resource management techniques that enable resource-efficient and performance-aware virtualized cloud infrastructures. More specifically, this work has proposed solutions that combat inefficient utilization of cloud resources and hence, minimize power consumption in the context of complex shared virtualized environments. Several approaches related to VM placement and consolidation have been presented in order to optimize both power and performance.

First, an OpenStack-based private cloud testbed is deployed in physical infrastructure as well as in virtualized environment in order to explore the performance of different applications which are diverse in terms of performance requirements. In addition, a testbed combining OpenStack infrastructure with a SDN based controller such as OpenDaylight and a large-scale network emulator, CORE is developed that allows to perform experimental driven research with a focus of cloud networking. Next, as the live VM migrations have non-negligible impact on the hosted applications, different network centric techniques have been presented to improve the live migration scheme in the OpenStack platform. Further, considering dynamic resource demands from enterprise customers, a scheduling approach is presented for a multi-located DC environment. The proposed scheme has considered different cost components associated with cloud infrastructures such as carbon emission costs, bandwidth provision costs, energy costs and shown a significant improvement in resource utilization while keeping a balance between cost and SLA requirements.

Additionally, a novel mathematical formulation is developed to characterize the uncertain behaviours in modern virtualized infrastructures due to non precisely specified resource demands or migration overhead. The mathematical model is formulated as a multi-objective optimization based on the theory of robust optimization in order to improve the VMs-to-PMs allocation decisions. In other words, the robust model allows to calculate the risk of SLA violations and the additional cost to protect against such risk for the cloud applications,

if their actual resource usage pattern or prediction about future demand is not accurate enough. The robust model takes long time to solve for large infrastructures because the search space grows exponentially with the number of VMs and PMs. Therefore, a novel robust heuristic based on Tabu Search is designed that can generate good quality solutions within polynomial time. In addition, co-located interference and corresponding performance degradation for diverse cloud applications is investigated carefully, and explored how this can improve the VM placement and allocation decisions in a shared virtualized environment. Particularly, an optimization model is proposed to highlight the importance of using resource sensitivity information of applications during migrations. The proposed approach improves the VM migration decisions by significantly reducing performance degradation of co-located VMs especially under the presence of high resource contention.

The research presented in this thesis seems to have raised more questions that it has answered. There are several lines of research arising from this work which can be investigated further. Firstly, most of the work presented in this thesis consider applications that can be run independently on VMs and hence, do not consider communication among the VMs. However, latency among communicating computing units (e.g. VMs or VNF components (VNFC)) is an important requirement for deploying and managing composite applications such as network function virtualization (NFV), which needs further research. The major challenge associated with such applications are accurate prediction of communication requirements and patterns. The problem is further complicated as these applications (such as VNFs) still largely rely on traditional TCP/ IP protocol stack, which may impose limitations on the performance because of extra latency added by network device virtualization. However, novel programmable packet forwarding planes such as P4 and their implementation in hardware may limit the underlying impact of virtualization. Further, as multiple computing units (e.g. VNFCs) need to exchange large volume of data over the network under capacity and latency constraints, the network sharing in virtualized environments also plays an important part.

A second line of research, which follows from the scalability issues associated with an centralized resource allocation scheme, is to investigate the performance of decentralized resource allocation schemes with respect to infrastructure size, management objectives, service level agreements, resource efficiency, and management overhead. As the centralized design has only one decision maker, the resulting decisions are more likely to be highly consistent but may suffer from scalability and fault tolerant issues. On the other hand, decentralized design can parallelize the decision making processes and divide the task over multiple decision makers which may improve the scalability and make the decision making process faster, but individual decisions may have negative impact on the overall state of the system.

Finally, elasticity (auto-scaling) is one of the major concerns for any application deployed over cloud infrastructure because of dynamic and heterogeneous workload patterns and sharing of underlying cloud resources, which has not been investigated thoroughly in this thesis. For instance, to automate, the

dynamic scaling of the available resources is complicated because the solution should be scalable, robust, adaptive, and capable of making rapid decisions. Further, accurate and adaptive scaling decisions are required so that they can improve overall resource utilization while reducing the operating cost for the cloud providers. Two major research questions are centered around this issue: when and how much scaling is required to achieve the expected performance? Finally, a large amount of performance unpredictability comes from many levels of indirection in the software stack. Therefore, managing a feedback control system for the application designers in order to shed light on the application level inefficiencies can improve the performance predictability of the cloud applications.

References

[1] Computing Community Consortium (CCC). Challenges and Opportunities with Big Data, Available: http://cra.org/ccc/docs/init/bigdatawhitepaper.pdf. Last Accessed 2017-02-05.

[2] Host Server CPU Utilization in Amazon EC2 Cloud, Available: http://huanliu.wordpress.com/2012/02/17/host-server-cpu-utilization-in-amazon-ec2-cloud/. Last Accessed 2017-02-05.

[3] Netflix, Available: http://netflix.com. Last Accessed 2017-02-05.

[4] Amazon EC2, Available: https://aws.amazon.com/ec2/. Last Accessed 2017-02-05.

[5] Google App Engine, Available: https://cloud.google.com/appengine/. Last Accessed 2017-02-05.

[6] Google Compute Engine, Available: https://developers.google.com/compute/. Last Accessed 2017-02-05.

[7] MATLAB (2014b): The Language of Technical Computing, Available: https://www.mathworks.com/products/matlab.html. Last Accessed 2017-01-27.

[8] Salesforce, Available: https://www.salesforce.com/. Last Accessed 2017-02-05.

[9] Dropbox, Available: http://www.dropbox.com/. Last Accessed 2017-02-05.

[10] ILOG CPLEX: High-performance software for mathematical programming and optimization, Available: http://www.ilog.com/products/cplex/. Last Accessed 2017-01-27.

[11] Phoronix Test Suite: Open-Source Automated Benchmarking, Available: http://www.phoronix-test-suite.com/. Last Accessed 2017-01-27.

[12] Presidents Council of Advisors on Science and Technology (PCAST). Designing a digital future: Federally funded research and development networking and information technology, Available: http://www.whitehouse.gov/sites/default/files/microsites/ostp/pcast-nitrdreport-2010.pdf. Last Accessed 2017-02-05.

[13] Windows azure, Available: http://www.windowsazure.com/. Last Accessed 2017-02-05.

[14] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim. CORE: A Real-time Network Emulator. In *MILCOM 2008 - 2008 IEEE Military Communications Conference*, pages 1–7, Nov 2008.

[15] M. Antosiewicz, G. Koloch, and B. Kamiński. Choice of Best Possible Metaheuristic Algorithm for the Travelling Salesman Problem with Limited Computational Time: Quality, Uncertainty and Speed. *Journal of Theoretical and Applied Computer Science*, 7:46–55, 2013.

[16] R. Bajpai, K. K. Dhara, and V. Krishnaswamy. QPID: A Distributed Priority Queue with Item Locality. In *Parallel and Distributed Processing with Applications, 2008. ISPA '08. International Symposium on*, pages 215–223, Dec 2008.

[17] D. G. Balan and D. A. Potorac. Linux HTB Queuing Discipline Implementations. In *Networked Digital Technologies, 2009. NDT '09. First International Conference on*, pages 122–126, July 2009.

[18] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, pages 164–177, New York, NY, USA, 2003.

[19] R. K. Barik, R. K. Lenka, K. R. Rao, and D. Ghose. Performance Analysis of Virtual Machines and Containers in Cloud Computing. In *2016 International Conference on Computing, Communication and Automation (ICCCA)*, pages 1204–1210, April 2016.

[20] L. A. Barroso, J. Clidaras, and U. Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second Edition*. 2013.

[21] F. Bellard. QEMU Open Source Processor Emulator, Available: `http://www.qemu.org`. Last Accessed 2017-04-05.

[22] A. Beloglazov and R. Buyya. Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers under Quality of Service Constraints. *IEEE Transactions on Parallel and Distributed Systems*, 24(7):1366–1379, July 2013.

[23] A. Ben-Tal, L. E. Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, Princeton Series in Applied Mathematics, Princeton, USA, 2009.

[24] D. Bertsimas and M. Sim. The Price of Robustness. *Operations Research*, 52(1):35–53, 2004.

[25] D. Bertsimas and A. Thiele. *Robust and Data-Driven Optimization: Modern Decision Making Under Uncertainty*, chapter 4, pages 95–122. 2006.

[26] M. Bolte, M. Sievers, G. Birkenheuer, O. Niehorster, and A. Brinkmann. Non-intrusive Virtualization Management using Libvirt. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pages 574–579, March 2010.

[27] R. Buyya, A. Beloglazov, and J. H. Abawajy. Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges. *CoRR*, abs/1006.0308, 2010.

[28] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci, J. Haridas, C. Uddaraju, H. Khatri, A. Edwards, V. Bedekar, S. Mainali, R. Abbasi, A. Agarwal, M. F. Haq, M. I. Haq, D. Bhardwaj, S. Dayanand, A. Adusumilli, M. McNett, S. Sankaran, K. Manivannan, and L. Rigas. Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 143–157, New York, NY, USA, 2011.

[29] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Softw. Pract. Exper.*, 41(1):23–50, January 2011.

[30] A. Celesti, F. Tusa, M. Villari, and A. Puliafito. Improving Virtual Machine Migration in Federated Cloud Environments. In *Evolving Internet (INTERNET), 2010 Second International Conference on*, pages 61–67, Sept 2010.

[31] 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*, NSDI'05, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[32] R. J. Creasy. The Origin of the VM/370 Time-sharing System. *IBM J. Res. Dev.*, 25(5):483–490, September 1981.

[33] W. Dargie. Estimation of the Cost of VM Migration. In *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*, pages 1–8, Aug 2014.

[34] C. Delimitrou. *Improving Resource Efficiency in Cloud Computing*. PhD thesis, Stanford University, Stanford University, 8 2016. `oai:purl.stanford.edu:mz539pv2699`.

[35] C. Delimitrou and C. Kozyrakis. Quasar: Resource-Efficient and QoS-Aware Cluster Management. In *Proceedings of the 19th International*

Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '14, pages 127–144, New York, NY, USA, 2014. ACM.

[36] P. J. Denning. ACM President's Letter: What is Experimental Computer Science? *Commun. ACM*, 23(10):543–544, October 1980.

[37] X. Fan, W. D. Weber, and L. A. Barroso. Power Provisioning for a Warehouse-sized Computer. In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ISCA '07, pages 13–23, New York, NY, USA, 2007.

[38] D. G. Feitelson. *Workload Modeling for Computer Systems Performance Evaluation*. Cambridge University Press, New York, NY, USA, 1st edition, 2015.

[39] M. Fera, F. Fruggiero, A. Lambiase, G. Martino, and M. E. Nenni. *Production Scheduling Approaches for Operations Management*, chapter 5. InTech, Rijeka, Croatia, 2013.

[40] Md. H. Ferdaus. *Multi-objective Virtual Machine Management in Cloud Data Centers*. PhD thesis, Monash University, 2016. `http://www.cloudbus.org/students/HasanulPhDThesis2016.pdf`.

[41] A. Forestiero, C. Mastroianni, M. Meo, G. Papuzzo, and M. Sheikhalishahi. Hierarchical Approach for Efficient Workload Management in Geo-distributed Data Centers. *IEEE Transactions on Green Communications and Networking*, PP(99):1–1, 2016.

[42] OpenStack Foundation. DevStack - an OpenStack Community Production, Available: `http://docs.openstack.org/developer/devstack/`. Last Accessed 2015-03-27.

[43] G. R. Gangadharan. Open Source Solutions for Cloud Computing. *Computer*, 50(1):66–70, January 2017.

[44] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences*, 79(8):1230 – 1242, 2013.

[45] H. Ghanbari, B. Simmons, M. Litoiu, and G. Iszlai. Exploring Alternative Approaches to Implement an Elasticity Policy. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 716–723, July 2011.

[46] C. Ghribi, M. Hadji, and D. Zeghlache. Energy Efficient VM Scheduling for Cloud Data Centers: Exact Allocation and Migration Algorithms. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pages 671–678, May 2013.

[47] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[48] F. Glover and K. Sörensen. Metaheuristics. *Scholarpedia*, 10(4):6532, 2015. revision #149834.

[49] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. Workload Analysis and Demand Prediction of Enterprise Data Center Applications. In *2007 IEEE 10th International Symposium on Workload Characterization*, pages 171–180, Sept 2007.

[50] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. Resource Pool Management: Reactive versus Proactive or Let's be Friends. *Computer Networks*, 53(17):2905 – 2922, 2009.

[51] B. L. Gorissen, I. Yanikoglu, and D. D. Hertog. A Practical Guide to Robust Optimization. *Omega*, 53:124–137, 2015.

[52] Red Hat. KVM-Kernel Based Virtual Machine, 2009. Available: `http://www.redhat.com/en/resources/kvm---kernel-based-virtual-machine`. Last Accessed 2015-02-27.

[53] N. R. Herbst, S. Kounev, and R. Reussner. Elasticity in Cloud Computing: What It Is, and What It Is Not. In *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*, pages 23–27, San Jose, CA, 2013.

[54] 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*, VEE '09, pages 51–60, New York, NY, USA, 2009.

[55] T. Hoff. Latency Is Everywhere And It Costs You Sales - How To Crush It, Available: `http://highscalability.com/latency-everywhere-and-it-costs-you-sales-how-crush-it`. Last Accessed 2017-02-27.

[56] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Willey, New York, 1991.

[57] T. H. Jørgensen, P. McKenney, D. Taht, J. Gettys, and E. Dumazet. FlowQueue-Codel. Internet Draft (informational), December 2014. Available: `http://tools.ietf.org/html/draft-ietf-aqm-fq-codel-00`.

[58] P. H. Kamp and R. N. Watson. Jails: Confining the Omnipotent Root. In *Proceedings of the 2nd International SANE Conference*, volume 43, page 116, 2000.

[59] D. Kliazovich, P. Bouvry, and S. U. Khan. DENS: Data Center Energy-Efficient Network-Aware Scheduling. *Cluster Computing*, 16(1):65–75, 2013.

[60] L. T. Kou and G. Markowsky. Multidimensional Bin Packing Algorithms. *IBM J. Res. Dev.*, 21(5):443–448, September 1977.

[61] R. Kumar and B. B. P. Parashar. Dynamic Resource Allocation and Management Using OpenStack. *National Conference on Emerging Technologies in Computer Engineering (NCETCE)*, pages 21–26, November 2014.

[62] Puppet Labs. Puppet and OpenStack, Available: `https://puppetlabs.com/solutions/cloud-automation/compute/openstack`. Last Accessed 2015-03-27.

[63] F. Larumbe and B. Sansò. A Tabu Search Algorithm for the Location of Data Centers and Software Components in Green Cloud Computing Networks. *IEEE Transactions on Cloud Computing*, 1(1):22–35, Jan 2013.

[64] X. Li, Z. Qian, S. Lu, and J. Wu. Energy Efficient Virtual Machine Placement Algorithm with Balanced and Improved Resource Utilization in a Data Center . *Mathematical and Computer Modelling*, 58(5-6):1222 – 1235, 2013.

[65] A. D. Lin, H. Franke, C. S. Li, and W. Liao. Toward Performance Optimization with CPU Offloading for Virtualized Multi-tenant Data Center Networks. *IEEE Network*, 30(3):59–63, May 2016.

[66] Z. Liu, A. Wierman, Y. Chen, B. Razon, and N. Chen. Data Center Demand Response: Avoiding the Coincident Peak via Workload Shifting and Local Generation. *Performance Evaluation*, 70(10):770 – 791, 2013.

[67] Canonical Ltd. Installing the Canonical Distribution of Ubuntu OpenStack, Available: `http://www.ubuntu.com/download/cloud/install-ubuntu-openstack`. Last Accessed 2015-03-27.

[68] M. Mahjoub, A. Mdhaffar, R. B. Halima, and M. Jmaiel. A Comparative Study of the Current Cloud Computing Technologies and Offers. In *Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on*, pages 131–134, Nov 2011.

[69] Z. Á. Mann. Allocation of Virtual Machines in Cloud Data Centers - A Survey of Problem Models and Optimization Algorithms. *ACM Comput. Surv.*, 48(1):11:1–11:34, August 2015.

[70] M. Mechtri, C. Ghribi, and D. Zeghlache. VNF Placement and Chaining in Distributed Cloud. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 376–383, June 2016.

[71] P. Mell and T. Grance. The NIST Definition of Cloud Computing, Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf. Last Accessed 2015-03-27.

[72] R. Miller. Inside Amazon's Cloud Computing Infrastructure, Available: http://datacenterfrontier.com/inside-amazon-cloud-computing-infrastructure/. Last Accessed 2017-02-05.

[73] M. P. Mills. The Cloud begins with Coal: Big Data, Big Networks, Big Infrastructure, and Big Power - An Overview of the Electricity Used by the Global Digital Ecosystem, Available: http://www.tech-pundit.com/wp-content/uploads/2013/07/Cloud_Begins_With_Coal.pdf?c761ac\&c761ac. Last Accessed 2017-02-05.

[74] M. Mishra and U. Bellur. Whither Tightness of Packing? The Case for Stable VM Placement. *IEEE Transactions on Cloud Computing*, 4(4):481–494, Oct 2016.

[75] R. Nasim and A. Kassler. Deploying OpenStack: Virtual Infrastructure or Dedicated Hardware. In *IEEE 38th International Computer Software and Applications Conference Workshops (COMPSACW)*, pages 84–89, 2014.

[76] R. Nasim and A. Kassler. Network-centric Performance Improvement for Live VM Migration. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 106–113, June 2015.

[77] R. Nasim, A. J. Kassler, I. P. Ẑarko, and A. Antonic̀. Mobile Publish/Subscribe System for Intelligent Transport Systems over a Cloud Environment. In *2014 International Conference on Cloud and Autonomic Computing*, pages 187–195, Sept 2014.

[78] R. Nasim, J. Taheri, and A. J. Kassler. Optimizing Virtual Machine Consolidation in Virtualized Datacenters Using Resource Sensitivity. In *2016 IEEE 8th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 168–175, Dec 2016.

[79] G. Neiger, A. Santoni, F. Leung, D. Rodgers, and R. Uhlig. Intel Virtualization Technology: Hardware support for efficient processor virtualization, August, 2006. Available: https://www.ece.cmu.edu/ece845/docs/vt-overview-itj06.pdf. Last Accessed 2017-02-05.

[80] A. Ngenzi. Applying mathematical models in cloud computing: A survey. *CoRR*, abs/1403.3649, 2014.

[81] S. Ohta. Strict and Heuristic Optimization of Virtual Machine Placement and Migration. In *Proceedings of the 9th International Conference on Computer Engineering and Applications (CEA '15)*, February 22-24 2015.

[82] OpenDaylight. OpenDaylight: Open Source SDN Platform, Available: https://www.opendaylight.org/. Last Accessed 2017-02-27.

[83] OpenStack Foundation. OpenStack Cloud Administrator Guide, Available: http://docs.openstack.org/admin-guide-cloud/content/. Last Accessed 2015-02-27.

[84] K. Pepple. *Deploying OpenStack*. O'Reilly Media, Inc., 2011.

[85] A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam, and N. Sharma. Towards autonomic workload provisioning for enterprise Grids and clouds. In *2009 10th IEEE/ACM International Conference on Grid Computing*, pages 50–57, October 2009.

[86] R. W. Ahmad and A. Gani and S. H. Hamid and M. Shiraz and A. Yousafzai and F. Xia. A survey on Virtual Machine Migration and Server Consolidation Frameworks for Cloud Data Centers. *Journal of Network and Computer Applications* , 52:11 – 25, 2015.

[87] M. Sedaghat, F. Hernandez-Rodriguez, and E. Elmroth. A Virtual Machine Re-packing Approach to the Horizontal vs. Vertical Elasticity Trade-off for Cloud Autoscaling. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, CAC '13, pages 6:1–6:10, Miami, Florida, USA, 2013.

[88] T. Setzer and A. Wolke. Virtual machine re-assignment considering migration overhead. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 631–634, April 2012.

[89] L. Shi, J. Furlong, and R. Wang. Empirical Evaluation of Vector Bin Packing Algorithms for Energy Efficient Data Centers. In *2013 IEEE Symposium on Computers and Communications (ISCC)*, pages 000009–000015, July 2013.

[90] V. Sobeslav and A. Komarek. OpenSource Automation in Cloud Computing. In *Proceedings of the 4th International Conference on Computer Engineering and Networks*, pages 805–812. Springer, 2015.

[91] K. Sörensen. Tabu Searching for Robust Solutions. In *Proceedings of the 4th Metaheuristics International Conference*, pages 16–20, 2001.

[92] P. Svärd, J. Tordsson, and E. Elmroth. Noble Art of Live VM Migration - Principles and performance of precopy, postcopy and hybrid migration of demanding workloads. *Technical Report, UMINF-12.11*, 2014.

[93] I. Tafa, E. Beqiri, H. Paci, E. Kajo, and A. Xhuvani. The Evaluation of Transfer Time, CPU Consumption and Memory Utilization in XEN-PV, XEN-HVM, OpenVZ, KVM-FV and KVM-PV Hypervisors Using FTP and HTTP Approaches. In *Intelligent Networking and Collaborative Systems (INCoS), 2011 Third International Conference on*, pages 502–507, Nov 2011.

[94] J. Taheri, A. Y. Zomaya, and A. Kassler. vmBBProfiler: A Black-box Profiling Approach to Quantify Sensitivity of Virtual Machines to Shared Cloud Resources. *Computing*, pages 1–29, 2017.

[95] E. G. Talbi. *Metaheuristics: from design to implementation*. John Wiley & Sons, Hoboken, New Jersey, USA, 2009.

[96] A. Tchernykh, U. Schwiegelsohn, V. Alexandrov, and E. G. Talbi. Towards Understanding Uncertainty in Cloud Computing Resource Provisioning. *Procedia Computer Science*, 51:1772 – 1781, 2015.

[97] E. Tsamoura, A. Gounaris, and K. Tsichlas. Multi-objective Optimization of Data Flows in a Multi-cloud Environment. In *Proceedings of the Second Workshop on Data Analytics in the Cloud*, DanaC '13, pages 6–10, New York, NY, USA, 2013.

[98] A. Videla and J. J. Williams. *RabbitMQ in Action: Distributed Messaging for Everyone*. Manning, 2012.

[99] A. D. F. Vigliotti and D. M. Batista. A Green Network-Aware VMs Placement Mechanism. In *2014 IEEE Global Communications Conference*, pages 2530–2535, Dec 2014.

[100] C. A. Waldspurger. Memory Resource Management in VMware ESX Server. *ACM SIGOPS Operating Systems Review*, 36(SI):181–194, 2002.

[101] P. Wette and H. Karl. DCT2Gen: A Versatile TCP Traffic Generator for Data Centers. *CoRR*, abs/1409.2246, 2014.

[102] Z. Xiao, W. Song, and Q. Chen. Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment. *Parallel and Distributed Systems, IEEE Transactions on*, 24(6):1107–1117, June 2013.

[103] J. Xu and J. A. B. Fortes. Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments. In *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, pages 179–188, Dec 2010.

[104] Y. C. Shim. Performance Evaluation of Static VM Consolidation Algorithms for Cloud-Based Data Centers Considering Inter-VM Performance Interference. *International Journal of Applied Engineering Research*, 11:11794 – 11802, 2016.

[105] X. S. Yang. *Harmony Search as a Metaheuristic Algorithm*, pages 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[106] R. Yanggratoke. *Data-driven Performance Prediction and Resource Allocation for Cloud Services*. PhD thesis, KTH, 8 2016. `urn:nbn:se:kth:diva-184601`.

Cost- and Performance-Aware Resource Management in Cloud Infrastructures

High availability, cost effectiveness and ease of application deployment have accelerated the adoption rate of cloud computing. This fast proliferation of cloud computing promotes the rapid development of large-scale infrastructures. However, large cloud datacenters (DCs) require infrastructure, design, deployment, scalability and reliability and need better management techniques to achieve sustainable design benefits. Resources inside cloud infrastructures often operate at low utilization, rarely exceeding 20-30%, which increases the operational cost significantly, especially due to energy consumption. To reduce operational cost without affecting quality of service (QoS) requirements, cloud applications should be allocated just enough resources to minimize their completion time or to maximize utilization.

The focus of this thesis is to enable resource-efficient and performance-aware cloud infrastructures by addressing above mentioned cost and performance related challenges. In particular, we propose algorithms, techniques, and deployment strategies for improving the dynamic allocation of virtual machines (VMs) into physical machines (PMs).

ISBN 978-91-7063-783-4 (print)

ISBN 978-91-7063-784-1 (pdf)

ISSN 1403-8099

DOCTORAL THESIS | Karlstad University Studies | 2017:21