3-26-2019

Cloud Workload Allocation Approaches for Quality of Service Guarantee and Cybersecurity Risk Management

soamar homsi
shoms001@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

CLOUD WORKLOAD ALLOCATION FOR QUALITY OF SERVICE GUARANTEE AND CYBERSECURITY RISK MANAGEMENT

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

by

Soamar Homsi

2019

To: Dean John L. Volakis
College of Engineering and Computing

This dissertation, written by Soamar Homsi, and entitled Cloud Workload Allocation for Quality of Service Guarantee and Cybersecurity Risk Management, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Kemal Akkaya

Arif Selcuk Uluagac

Hai Deng

Jason Liu

Gang Quan, Major Professor

Date of Defense: March 01, 2019

The dissertation of Soamar Homsi is approved.

Dean John L. Volakis
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development
and Dean of the University Graduate School

Florida International University, 2019

DEDICATION

To my son, Noah Homsi.

ACKNOWLEDGMENTS

I would like to express my gratitude to my academic advisor, Dr. Gang Quan, for his guidance, and support during the years of my Ph.D. program. I also want to thank my Ph.D. committee members, Dr. Kemal Akkaya, Dr. Arif Selcuk Uluagac, Dr. Hai Deng, and Dr. Jason Liu for their time and valuable insights that significantly improved the quality of this dissertation.

Further, I would love to show my appreciation to my industrial mentor, Dr. Laurent Njilla, from the Air Force Research Laboratory (AFRL), Department of Defense, for inspiring me to explore new challenging research ideas. I want to extend my thanks to Mrs. Tracy DiMeo, Dr. Timothy S. Kroecker, Mr. Steven D Farr, Chief Dr. Douglas Wynne, Mr. Wladimir Tirenin, and Chief Mr. Jame Perreta from the AFRL for the encouragement and commitment in helping me establish and pursue a career.

I would like to thank Ms. Juan Yang and my brother, Dr. Yazan Homsi, as well for their considerable assistance in editing this dissertation. Moreover, my graduate studies wound not have been the same without my colleague, Dr. Gustavo Chaparro-Baquero, who never hesitated to extend a hand whenever I needed help, and without my other colleagues Dr. Shuo Liu, Dr. Qiushi Han, Dr. Tianyi Wang, and Dr. Shi Sha.

I am, most importantly, deeply grateful to my parents, my sisters, Dr. Yara and Dr. Aya Homsi, and my son, Noah, who always supported me with love and encouragement throughout my academic endeavor. Furthermore, I am genuinely thankful and indebted to my brother, Dr. Moetaz Homsi, my second mother, Mrs. Maria Eladia Almonte, and her family, my loving aunt Dr. Rajaa Harfoosh, and my uncle Dr. Rami Arfoosh for their love and support.

Finally, this work is supported in part by (1) The Summer Fellowship Program for Students with the Cyber Assurance Branch of the Air Force Research Laboratory, U.S. Department of Defense, Rome, NY; (2) The seed program from the Florida Center for

Cybersecurity; (3) The National Science Foundation (NSF) under grants CNS-1423137, CNS-1551661, CNS-1565474, and ECCS-1610471.

ABSTRACT OF THE DISSERTATION

CLOUD WORKLOAD ALLOCATION FOR QUALITY OF SERVICE GUARANTEE AND CYBERSECURITY RISK MANAGEMENT

by

Soamar Homsi

Florida International University, 2019

Miami, Florida

Professor Gang Quan, Major Professor

It has become a dominant trend in the industry to adopt the cloud computing technology – thanks to its unique advantages in flexibility, scalability, elasticity and cost efficiency – for providing online cloud services over the internet using large-scale data centers. In the meantime, the relentless increase in demand for affordable and high-quality cloud services, for individuals and businesses, has led to tremendously high power consumption and operating expense levels and, thus, has posed crucial challenges on cloud service providers in finding efficient resource allocation policies. Allowing several services or Virtual Machines (VMs) to commonly share the cloud's infrastructure enables cloud providers to optimize resource usage, power consumption, and operating expense. However, servers sharing among users and VMs causes performance degradation and results in cybersecurity risks. Consequently, how to develop efficient and effective resource management policies that can make the appropriate decisions to optimize the trade-off among resource usage, service quality, and cybersecurity loss plays a vital role in the sustainable future of cloud computing.

In this dissertation, we focus on cloud workload allocation problems for resource usage optimization subject to Quality of Service (*QoS*) guarantee and cybersecurity risk constraints. To facilitate our research, we first develop a cloud computing prototype that we utilize to empirically validate the performance of different proposed cloud resource management schemes under a close to practical, but also isolated and well-controlled, environment. Second, we research the resource management policies for time-sensitive cloud services with *QoS* guarantee. Based on the queuing model with reneging, we establish and formally prove a series of fundamental principles, between the timing characteristics of cloud services and their resource demands, and based on which we introduce several novel resource management algorithms that statically guarantee the *QoS* requirements for cloud users.

Third, we study the problem of mitigating the cybersecurity risks and losses in cloud data centers via proposing secure cloud resource management strategies. We employ the Game theory to model the VM-to-VM interdependent cybersecurity risks in cloud clusters. We conduct a thorough analysis based on our game-theoretic model and establish several algorithms for cybersecurity risk management. Specifically, we start our cybersecurity research from a simple case with only two types of VMs. We next extend it to a more general case with an arbitrary number of VM types. Our intensive numerical and experimental results show that our novel algorithms can significantly outperform the existing methodologies for large-scale cloud data centers in terms of optimizing resource usage, cybersecurity loss, and computational effectiveness.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1

INTRODUCTION

Joining cloud computing market [8–16] has recently become the dominant trend for small and large companies to continuously deliver affordable and high-quality business services over the internet. Cloud computing notably minimizes the Information Technology (*IT*) infrastructure expense by eliminating the necessity for any additional hardware and software purchases and maintenance cost [17]. Cloud computing supplies a commonly shared pool of on-demand, scalable, and elastic *IT* resources and services using a pay-per-use pricing model [18, 19]. In the meantime, the constant increase in desire for affordable cloud services with stringent service qualities and high-security levels has posed critical challenges on cloud service providers in finding efficient resource allocation policies capable of optimizing the trade-off among energy consumption, operating expense, quality of service, and cybersecurity risks [20, 21]. One standard approach that cloud providers adopt to maximize resource usage is Virtual Machine (VM) multiplexing in which they pack several VMs on the same server. Allowing multiple applications and VMs, with different resource and cybersecurity requirements, to share the same cloud infrastructure components, notwithstanding, results in performance degradation of cloud services and interdependent cybersecurity *IC* risks [20–23]. Cloud providers henceforth need efficient resource allocation policies with multi-objective goals in order to thrive in such an ever-expanding and competing cloud computing market [24].

In this chapter, we give a brief introduction about the cloud computing technology, its unique characteristics, and its importance to each sector in our modern life. We then discuss the urgent need of cloud service providers for efficient and effective resource management methods that can provide affordable and secure cloud services with guaranteed Quality of Service (*QoS*) for cloud users. We define our research problem afterward and summarize our contributions. Finally, we describe the organization of this dissertation at the end of this chapter.

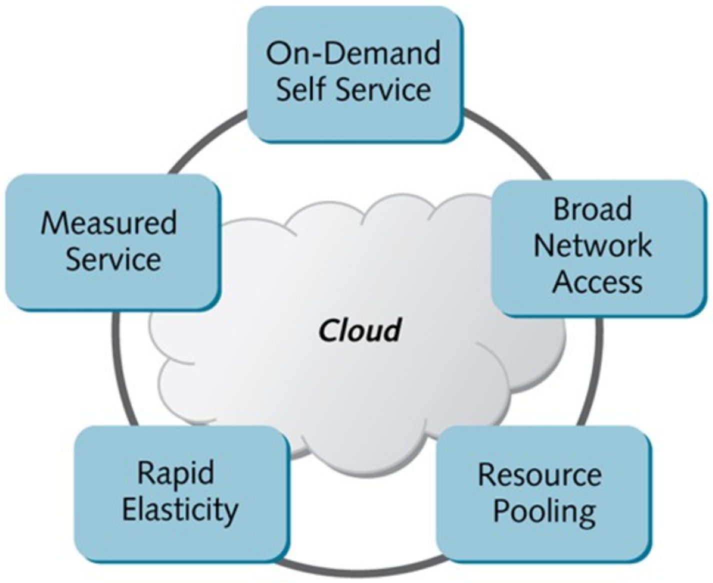
1.1 Cloud computing

Cloud computing offers a commonly shared pool of on-demand, scalable, and elastic *IT* resources and services (e.g., processing, network, software, information, and storage) via a pay-per-use pricing model [18, 19]. Cloud service providers make their services available to cloud consumers using web-based applications, and accessible through a web browser, as if those applications were locally installed on their own computers [25]. Today, cloud computing has grown as one of the most prominent and influencing computing paradigms for managing and delivering services over the internet.

1.1.1 What exactly is cloud computing?

After years of work and more than fifteen drafts, the National Institute of Standards and Technology's (*NIST*) [18] eventually issued the sixteenth and final definition of the cloud computing paradigm in 2011. The *NIST* defines cloud computing (e.g., *NIST* Special Publication 800-145)) 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. This cloud model is composed of five essential characteristics, three service models, and four deployment models."*

As Fig. 1.1 illustrates, the five unique characteristics of cloud computing, according to NIST, are as follwos:



Source: https://slideplayer.com/

Figure 1.1: The characteristics of cloud computing [1]. This figure lists the five major characteristics of cloud computing according to the National Institute of Standards and Technology (NIST).

- *On-demand self-service*

  Cloud users can automatically provision as much computing resources as needed without the service provider's interaction [8, 26].

- *Broad network access*

  Cloud services are made available over any network connected to the internet and are accessible through any computing device, such as mobile devices, personal computers, and servers [27].

- *Resource pooling*

  Cloud service providers aggregate cloud computing physical and virtual resources into resource pools. Providers subsequently assign or reassign each pool to a group of cloud users, i.e., a multi-tenant model. Cloud users have no information about the actual geographical location of their resource pools [28].

- *Rapid elasticity*

  Service providers can resiliently allocate and release computing resources to accommodate consumers' fluctuating demands according to peak business hours. This rapid provisioning process not only saves cloud users the need to invest in a private *IT* infrastructure but also grants them access to a virtually unlimited amount of computing resources anytime and anywhere [29].

- *Measured service*

  Cloud providers implement online metering and pricing tools that are capable of dynamically monitoring and measuring the usage of resources, i.e., a pay-per-use pricing model. Such metering and pricing tools provide transparency for both service providers and users about resource utilization and billing information, respectively [30].

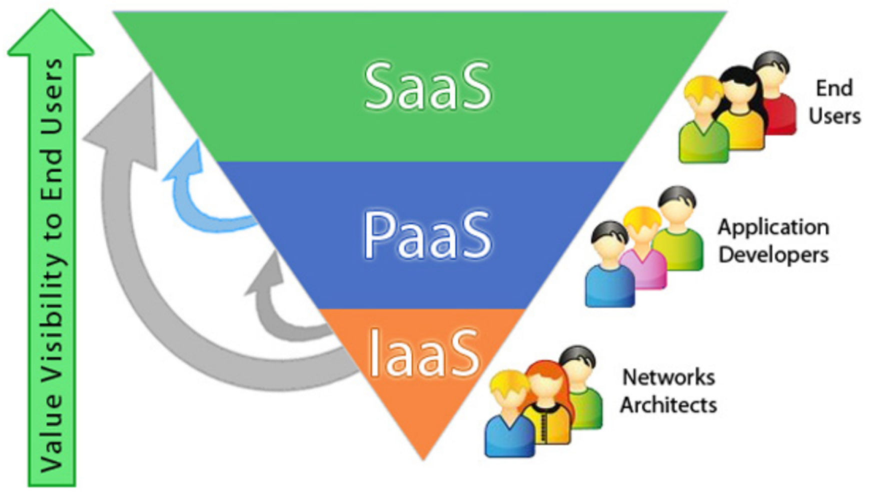
Cloud platforms can be classified according to its service model or according to its deployment model. The major cloud service models, that are illustrated in Fig. 1.2, are:

- *Infrastructure as a Service (IaaS)*

  Service providers offer businesses and individuals on-demand computing resources (e.g., storage, processing, etc.) accessible over a network to optimize the IT costs of cloud users [12].

- *Platform as a Service (PaaS)*

  In the platform service model, cloud service providers grant cloud users access to online development environments with powerful and capable supporting infrastructure. The *PaaS* services thus enable programmers and developers to affordably produce and deliver quality software solutions, from their homes, without incurring any powerful servers or costly software licenses [31].



Source: http://www.smechannels.com

Figure 1.2: The cloud service models [2]. This figure illustrates the three cloud service models, i.e., Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

- *Software as a Service (SaaS)*

  End-users can leverage the on-demand cloud services that are available via web pages or through Application Program Interfaces (*APIs*) and are accessible over the Internet [32].

In this dissertation, we concentrate on designing efficient and effective cloud allocation strategies in order to optimize cloud resource usage while providing cloud services for cloud users with guaranteed quality levels, regardless of the cloud deployment model in-use. The *NIST* definition lists the following four general cloud deployment models [18]:

- *Public Clouds*

  A public cloud is an openly available cloud platform that is owned and operated by a third-party [33]. Examples of public clouds include Amazon Elastic Com-

Figure 1.3: Examples of public cloud service providers [3]

pute (*EC*2) [31], IBM's Blue Cloud, Sun Cloud, Google AppEngine, and Windows Azure Services Platform, i.e., Fig. 1.3.

- *Community Clouds*

  A community cloud is a public cloud, but its access is limited to a particular collection of cloud consumers [8].

- *Private Clouds*

  A private cloud is a cloud platform that is owned and managed by a single organization. Access to private cloud services is limited to the employees and clients of the organization that runs and owns the private cloud platform [34].

- *Hybrid Clouds*

  A hybrid cloud is a platform that combines two or more different deployment models. For instance, an organization can store its sensitive data on its own cloud platform, while it can offer its business services to its client using public cloud platforms [17].

To understand the significant role of cloud computing in today's *IT*-based market, we should first examine the unique features of cloud computing as opposed to those from other computing paradigms.

1.1.2 The uniqueness of cloud computing and its market potential

We have learned so far that cloud computing offers plenty of advantages for individuals and organizations. Moreover, the unique features of cloud computing cause the annual growth of the cloud computing market to continuously increase year after year compared to the market of other computing paradigms [4].

Unlike the application-oriented-based supercomputing and cluster computing, cloud computing is a service-oriented utility computing with virtualization technology [34]. Utility computing is a business model that attracts cloud customers using a pay-per-use pricing model [35], [36]. Cloud computing, thus, supplies consumers with on-demand computing resources similar to those services provided by public utilities, such as electricity [17, 19]. Cloud providers hence equip each cloud cluster with innovative metering, monitoring, and billing tools. Amazon, Sun, and IBM are examples of companies that utilize their cloud infrastructure to offer Utility computing services publicly [34]. For instance, Amazon, i.e., AWS, offers pay-by-hour utility compute services using $EC2$ and utility pay-by-usage storage services using the Simple Storage Service ($S3$) [37].

Cloud consumers can quickly provision as much computing resources as needed without the service provider's direct intervention, knowledge about the infrastructure, or skills to utilize cloud services [9, 34]. The process of allocating cloud resources to consumers seems similar to hosting online applications using the service-oriented web hosting scheme over distributed computing platforms [17]. However, distributed computing does not rely on vastly scalable computing infrastructure that is spread across several countries or continents, as the cloud computing infrastructure that is supported by several large-scale cloud data centers [17].

Several scholars argue that cloud computing emerged out of grid computing as a result of the transfer from the focus on providing physical computing resources with high performance to the emphasis on delivering economy-based abstracted resources and services [17]. Furthermore, grid computing is different from cloud computing in its foundation and structure, i.e., the grid technology relies on standard, well-documented protocols, and accessible interfaces to deliver notable qualities of computing service [38].

Resource sharing in cloud data centers significantly reduces energy cost and operating expense for cloud consumers [19]. Governments and businesses replace their private data centers with the cloud's remote infrastructure [17]. The U.S. Office of Management and Budget (*OMB*) has issued guidance to reduce the number of federal, private data centers starting in 2010, as reported in [39]. The goal is to shift most of the critical and non-critical *IT* workload to cloud data centers to save energy costs via the adoption of proficient cloud resource management policies. Unwise workload allocation practices in traditional data centers result in low server utilization and high power consumption per consumer, compared to (65%) cloud's server utilization and (84%) power consumption per cloud's consumer as reported by the Data Center Efficiency Assessment from the Natural Resources Defense Council in 2014. Traditional data centers are "duplicative, costly and complex", as reported in [40], causing the annual electricity cost to increase by 2.5% each year over the last 20 years [41]. The scalable and flexible cloud infrastructure optimizes resource usage and server utilization by adopting adequate resource-sharing policies, such as server consolidation.

The scalable and elastic cloud infrastructure likewise reliably and affordably fulfills the increasing demands for cloud users. For example, cloud computing provides small

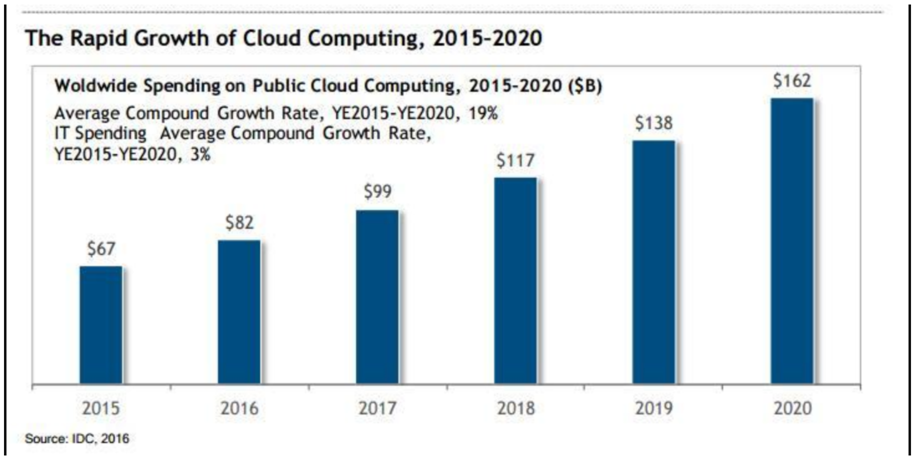


Figure 1.4: Rapid growth of cloud computing [4]. This figure shows the average compound growth of cloud computing between the years 2015 and 2020.

businesses with the tools required to compete with middle-sized and large-sized organizations via saving them the cost of investing in a private *IT* infrastructure [34].

The unique features of cloud computing, which we described earlier, have been attracting more and more businesses and organizations to make the transition to the cloud-based flourishing market.

It is unsurprising, thereupon, that several studies predict major growth in the adoption of cloud computing in the next several years. According to a study done by Forbes [4], the cloud computing market is expected to reach $162B in 2020 attaining a Compound Annual Growth Rate (*CAGR*) of 19% compared to the $67B in 2015 at 3% CARG, as illustrated in Fig. 1.4.

On the other hand, the rapid growth of the cloud market calls for comparable expansion in the size and spending of the cloud infrastructure. Fig. 1.5 illustrates data center's *IT* spending according to deployment type [5]. The *IT* spending in cloud data centers has been increasing by 15.3% each year, and the total *IT* spending of cloud data centers is expected to reach $41.7B in 2021, according to IDC [5].

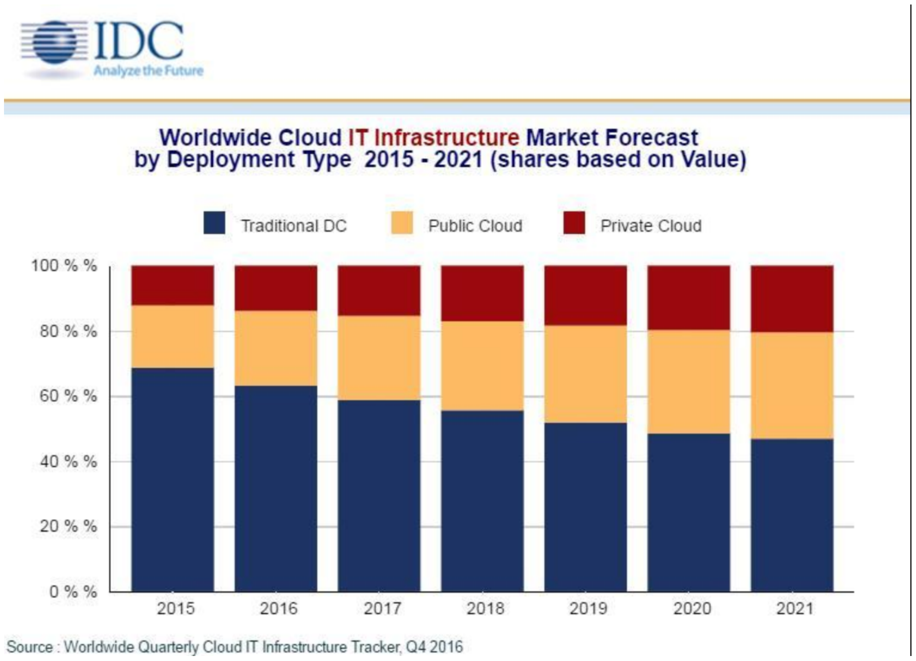


Figure 1.5: Data center's IT spending according to the deployment type [5]. This figure shows that the IT spending in cloud data centers has been increasing by 15.3% each year and the total IT spending of cloud data centers is expected to reach $41.7B in 2021 according to reports by IDC [5].

Although cloud computing provides irresistible benefits, it raises many concerns including high power consumption, low resource utilization, performance degradation as a result of resource sharing, and cybersecurity risks. Those challenges arise as a result of the continuous expansion of cloud data centers and the limitations of the current cloud resource management policies, as we discuss in the next section [42–44]. Specifically, we first define cloud resource management. We then argue the critical role of efficient cloud workload allocation strategies in providing cloud services with *QoS* guarantee and minimized cybersecurity risks.

1.1.3 Cloud computing challenges

To continue offering compelling advantages and cost-saving options for cloud users, cloud service providers have to face the following major cloud computing's challenges:

- *Power consumption and energy cost*

  Excessive power consumption and energy cost are among the principal concerns in cloud data centers. Cloud service providers endeavor to supply the markets with more reliable, yet less costly, services to contribute to a modern world that demands Everything as a Service ($EaaS$) [15]. Whereas the price of online services decreases and the performance of computing systems increases at a fast rate, the performance-per-watt of computing components increases at a much slower rate [45]. As an example, in 2013, the annual electricity consumption of data centers in the United States was close to 91 billion KiloWatt Hour (KWH), which is larger than the annual amount of electrical power required by most countries [46]. Thereupon, service providers are taxed by looming energy bills as they try to produce satisfactory QoS guarantee. Such a consumption rate of electricity is not only a cost problem but also a serious threat to the environment as a result of the large amounts of carbon dioxide emissions during powering and cooling cloud data centers [46]. It is imperative thereupon for service cloud providers to adopt energy-aware resource allocation strategies that minimize power consumption and electricity cost in cloud data centers [47].

- *IT-based operating expense*

  The provider's *IT* operating spending in cloud data centers (e.g., servers, software, licenses, network equipment, etc.) is expected to increase 15.3% each year, after 2017, to reach $41.7B in 2021, according to the Worldwide Cloud *IT* Infrastructure Market Forecast, as illustrated in Fig. 1.5. Server consolidation allows several services or VMs to share the same server to optimize resource usage [48]. Sharing the same resource type, among several applications and users, unfortunately, causes performance degradation which affects the quality levels of cloud services and results in *IC* risks [33].

- *Service Level Agreement (SLA)*

  The cloud service provider and consumers negotiate preferred quality levels for cloud services (i.e., *QoS*), under which their satisfaction is guaranteed. The negotiation eventually ends in a Service Level Agreement (*SLA*) [49]. Therefore, the cloud service provider must guarantee *QoS* conditions for his/her consumers, i.e., provides a *QoS* guarantee. Maintaining redundant computing resources cannot always ensure the *QoS* conditions for cloud users since it cannot adaptively scale up or down according to the abrupt changes in the cloud workload arrival and execution rates [32,47,49]. It can also lead to tremendously high power consumption and energy cost due to the low resource utilization [24]. Cloud service providers consequently need sound resource management policies that can swiftly and judiciously make allocation decisions while ensuring the *QoS* guarantee for cloud consumers.

- *Low resource utilization*

  Low resource utilization [50] is a prevailing problem in cloud data centers, and is a major leading factor to their high power consumption and increased operating cost [45]. For example, the utilization of Google's servers is less than 50% on average [21]. Maximizing resource utilization becomes more critical when performance must meet a defined satisfactory level of service given by *QoS* conditions. The challenge is hence how to allocate the cloud's workload in a way that maximizes resource usage and guarantees the *QoS* requirements.

- *Cybersecurity risks*

  Although cloud computing provides interesting solutions to individuals and organizations, cloud computing, like any other *IT* technology, is not completely safe from cyber attacks. Precisely, the commonly shared infrastructure of the cloud among VMs and users undeviatingly exposes cloud applications and VMs to several cybersecurity risks, such as side-channel attacks [51–53] and VM-to-VM *IC* risks [39].

Consequently, how to develop efficient and effective resource management policies to face those challenges and optimize the trade-off among resource utilization, *QoS* guarantee, and cybersecurity risks plays a vital role in the sustainable future of cloud computing.

1.2 Cloud resource management

The benefits of cloud computing enable businesses to compete and thrive by taking advantage of the scalable and agile cloud computing platforms. In the meantime, the constant increase in desire for affordable and high-quality cloud services has led to several significant concerns that forced cloud service providers to find new efficient and effective resource allocation policies.

1.2.1 What is cloud resource management?

Resource management is a fundamental process for any cloud platform. It is a core function which effectively and efficiently allocates and releases computing resource to meet the demand of cloud users while providing them with satisfactory *QoS* [54]. Efficient resource management allows cloud service providers to share cloud resources among cloud services and users with high availability and optimized utilization [20–22]. It is, yet, a very complicated task for cloud service providers to provide all the required resources to avoid *SLA* violations due to the limited capacities of available resources. On the contrary, inefficient resource management drastically affects the performance and cost of a cloud platform [23].

Cloud service providers need effective resource management strategies that quickly and intelligently make allocation decisions without violating the *SLA* conditions between the cloud service provider and cloud users [24]. Maintaining excessive computing resources cannot adequately guarantee the *QoS* conditions for cloud users as it can lead to tremendously high power consumption and low resource utilization [47]. Besides, allowing several applications, VMs, and users to commonly share a pool of cloud resources brings about many cybersecurity risks, such as the side-channel attacks [51, 52], and the VM-to-VM *IC* risks [33, 55].

Next, we emphasize the crucial role of efficient and effective resource allocation policies to provide *QoS* guarantee and to minimize cybersecurity risks.

1.2.2 The need for cloud resource management to provide QoS guarantee

Cloud service providers offer cloud services to the cloud consumers who expect the cloud system to operate without service interruption and to perform according to an *SLA* in a cloud Service Oriented Architecture (*SOA*) [56]. However, It is difficult, for the cloud service provider, to ensure the ideal satisfaction of all cloud consumers. The cloud service provider, thus, negotiates a feasible balance with the consumers in the form of agreements. These agreements are referred to as *SLA*s and evaluated based on *QoS* criteria. The term *QoS* is a metric which measures the service performance, in terms of predefined criteria that characterize specific attributes (e.g., processing rates [57], latency [58, 59], etc.) of the cloud platform, to ensure the degree of cloud consumers' satisfaction and to enforce *SLA*s [60]. Providing *QoS* guarantees nonetheless is a challenging task when considering the allocation of time-sensitive cloud services while aiming to optimize cloud resource usage and to minimize the average response time to avoid *SLA* violations.

Hard real-time schedulers must always provide guarantees on preventing requests from missing deadlines. Therefore, real-time requests must have deterministic and well-defined parameters (e.g., arrival time, response time, etc.). On the other hand, the cloud workload is random. Cloud interactive services [61], as online gaming [21] and multimedia streaming services [62], are described by soft-timely constraints [61]. Cloud service providers must guarantee that a predefined percentage of them meet their deadlines, i.e., deadline miss ratio [63, 64]. The cloud service provider, otherwise, violates his/her *SLA* with the cloud users [20]. Therefore, probabilistic or stochastic cloud allocation policies are necessary to accommodate requests with random or statistical time parameters [65].

Furthermore, scheduling real-time requests in embedded system is either using a single-core [66, 67] platform or multi-core platform [68–71]. On the other hand, a cloud provider can schedule requests on a single server [72–74], on a cluster of servers [75, 76], on a single data center [77, 78], or on a set of geographically distributed data centers [79]. The cloud service provider consequently has to consider other interference related parameters, in addition to the timing constraints, before making a scheduling decision, such as network delay, communication contention, etc, [80], [81].

Ensuring the *QoS* of cloud services yet becomes a more challenging process when several cloud consumers can have different *SLA* with various *QoS* attributes [82]. The cloud service provider's common practice has been to host requests with the same *QoS* on the same VM to optimize resource usage [83]. Although this method simplifies the resource allocation process, it does not provide *QoS* guarantees and it increases energy cost and operating expense [26, 84].

Cloud service providers not only need to provide a competing and guaranteed *QoS* for their cloud consumers but also have to minimize the energy cost and operating expense to increase their profit. Power-saving techniques (e.g., Server consolidation [85], VM migration [86], and server dynamic configuration [87]) usually, if not always, cause a degraded computing performance and, as a result, *SLA* violations. Whereas resource over-provisioning is a common and simple solution to avoid *SLA* violations, resource over-provisioning is an expensive method by which resources are drastically underutilized, and it cannot provide *QoS* guarantees under the fluctuating cloud workload [26, 84].

Cloud service providers, thus, need to develop efficient cloud resource management policies with timing *QoS* guarantees that ensure the satisfaction of the cloud consumers and optimize the energy cost and operating expense of cloud data centers.

1.2.3 The need for cloud resource management to minimize cybersecurity risks

Cybersecurity is the most crucial concern when adopting cloud computing [42, 44]. The commonly shared infrastructure of the cloud among VMs and users, unfortunately, exposes cloud workload and users to several cybersecurity risks, such as side-channel attacks [51–53] and VM-to-VM *IC* risks [39].

The side-channel attacks are carried on using VM collocation. The successful collocation, of the attacker's VM with a victim VM, allows the attacker to build different kinds of side channels to extract private information about the victim's VM (e.g., workload's traffic rate [52], cryptographic keys [51], etc.) that eventually enables him/her to launch a successful attack. Several software-based and hardware-based methods were proposed to address this type of attack [88, 89].

Software-level methods are usually limited in their capabilities to the boundaries of the hosting VM or are unable to keep up with all the new attack techniques proposed by hackers [90]. On the other hand, hardware-level methods proposed to countermeasure side-channel attacks are costly and impractical, as they required modifying the architecture of the cloud infrastructure [88, 89]. Furthermore, dynamic VM allocation can significantly minimize the attacker chances of colocating his/her VM with the victim VM [90, 91] and maximizes resource utilization [92] through live VM migrations. However, live migration comes with a migration overhead results in performance degradation and *SLA* violations [49].

In the VM-to-VM *IC* attacks, an attacker can compromise the hypervisor after a successful attack on one of its vulnerable VMs [33, 55]. He/she consequently can compromise all other coexisted secure VMs on that hypervisor. VM multiplexing methods, which are extensively used in cloud data centers, expose VMs with sensitive data or high value to *IC* risks. Those risks allow the attackers to bypass any security measures applied to those critical VMs and to indirectly compromise them using a less secured colocated VMs [33, 52]. While allocating VMs with different security requirements to different servers incurs lower security risks, it exacerbates resource usage and energy cost. Hence, the challenging question is –how can the providers optimize the trade-off between minimizing the *IC* risks and improving the energy cost and operating expense? –.

In the next section, we introduce our research problem.

1.3 Research problem

In this dissertation, we study the research problem of developing efficient and effective cloud resource management policies and techniques with the focus on optimizing the trade-off among optimizing resource utilization, providing *QoS* assurance, and minimizing cybersecurity risks. Specifically, our research problem can be formulated as the following:

Given

- *The performance characteristic of a cloud cluster,*

  - number of servers, resource performance, cybersecurity performance, power consumption performance, etc.

- *The specifications of a set of cloud services or a set of VMs,*

  - timing specifications, workload characteristics, QoS *requirements, cybersecurity specifications, etc.*

- *The design constraints,*

  - response time, deadline miss ratios, cybersecurity loss, power consumption, etc.

- *The design objectives,*

  - optimize performance criteria, such as timing, cybersecurity, cost, etc.

Determine

where and how to allocate the requests or VMs so that the design objectives are optimized.

1.4 Our contributions

Towards this research problem, we make the following contributions:

1. To facilitate our experiment and validation work, we first develop a cloud computing prototype that closely mirrors industry-compatible cloud platforms. Our prototype can provide cloud services as any middle-sized cloud service provider does. The prototype generates and runs several general and cloud-specific benchmarks under an isolated and well-controlled environment. We further incorporate new

workload scheduling, resource provisioning, and performance monitoring schemes that we proposed in this dissertation into the platform.

2. Different from previous studies that employ separate VMs for hosting requests with different *QoS* requirements, we develop a cloud service multiplexing method, based on the queuing model with reneging, that enables requests of the same service type, but with different *QoS* constraints, to share the same VM. To our best knowledge, this is the first approach by which different requests with different *QoS* requirements can be hosted on a single node to increase resource utilization. We also devise a novel methodology that correctly discards potential failure requests as soon as possible to minimize processing rate demands, and to reduce total power consumption with statistically guaranteed *QoS*. We introduce a packing and consolidation algorithm that statistically ensures the *QoS* requirements of cloud requests in terms of deadline miss ratios. In addition to the analytical validation of our proposed methods, we experimentally verify them, under general and cloud-specific workloads, using our cloud platform. For example, we use the Data Caching benchmark that emulates the behavior of a Twitter caching server and assumes strict quality of service guarantees, such as, 95% of the request must finish within 200 ms. Extensive experimentation results show that our proposed methods widely outperform existing approaches in terms of *QoS* satisfaction, power consumption efficiency, resource demand minimization, and electricity cost saving.

3. A major limitation of consolidating VMs of different security requirements onto a single server is that it can result in VM-to-VM interdependent cybersecurity *IC* risks. For example, the odds of successfully compromising a secure critical VM are high when an attacker compromises the hosting hypervisor after a successful direct attack on one of its less secured, non-critical VMs. Therefore, we formulate the allocation problem with cybersecurity awareness into a non-cooperative, zero-

sum theoretical game model between an attacker and the service cloud provider. We develop a set of new conditions to identify the existence of an equilibrium allocation strategy quickly. We mean by an –equilibrium– allocation strategy that the allocation policy in which neither the provider nor the attacker can benefit from unitarily deviating from their allocation or attack decisions, respectively. We then incorporate several resource usage parameters into a non-zero-sum game model. We identify the cases under which several static and dynamic equilibrium allocation strategies exist. We also derive the lower-bound and upper-bound of the *IC* risks.

4. Finally, we extend our game models to include VMs with more general cybersecurity and resource requirements. We focus on the static VM allocation problem to study how to (1) minimize the provider's worst potential cybersecurity loss under constrained resource usage and how to (2) optimize cloud resource usage while ensuring that the worst potential cybersecurity loss is always less than a given cybersecurity threshold. We show later in this dissertation that a constrained-allocation problem is a typical NP-hard problem, and, thus, we formulate the security-constrained and resource-constrained VM allocation problems using the Mathematical Programming (*MP*) approach to obtain the optimal solutions, which will be used as a comparison baseline against other proposed approaches in this dissertation and when the problem size is small. We formally model the resource and security-constrained allocation problems as a non-cooperative two-player zero-sum game. We conduct a thorough analysis of the characteristics of the pure Nash Equilibrium (*NE*) strategy profiles in our game model, which we formulate as a series of lemmas and theorems. Based on the insights of our analysis, we develop several effective and computationally efficient algorithms to allocate VMs, of different resource and security requirements, with resource usage and security loss optimized. We have implemented our algorithms and studied their efficiency and effectiveness. Our ex-

tensive simulation results show that our novel approaches are good trade-offs when compared with the computational-intensive approaches, such as the ones based on the *MP* approaches, the existing *NE* search methods, or the computationally efficient multi-dimensional bin-packing methods.

1.5 Organization

We organize the rest of this dissertation as follows. Chapter 2 discusses the related works to our research. Chapter 3 introduces our cloud computing platform which is used to empirically validate our proposed methods and algorithms in this dissertation. Chapter 4 presents our research on workload consolidation for cloud data centers to optimize their resource usage and electricity cost while providing guaranteed *QoS* for cloud users via the early reneging of failed requests. Chapter 5 studies how to collocate critical VMs and non-critical VMs onto cloud clusters with or without resource and energy constraints to minimize the *IC* risks. Chapter 6 extends our game models and studies how to mitigate the provider's worst potential cybersecurity loss under constrained resource usage and how to optimize cloud resource usage while guaranteeing that the worst possible cybersecurity loss is always less than a given cybersecurity threshold. Finally, Chapter 7 concludes this dissertation and discusses our future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

We present the related works to our research in this chapter. We start by briefly describing real-time scheduling and the differences between real-time request scheduling in embedded systems and time-sensitive request scheduling in cloud platforms. We further discuss the current research on time-sensitive requests scheduling in cloud platforms. We additionally describe the related work to VM allocation in cloud data centers, with and without security awareness. Finally, we summarize our discussion at the end of the chapter.

2.1 Real-time scheduling

Real-time systems are usually reactive systems that must comply with deadlines. In the real-time systems, the correctness of the execution of a task does not only depend on the computational results, but also on completing the task before its deadline [93, 94].

A real-time system is a finite collection of independent or dependent services, each of which generates a potentially infinite sequence of requests [95]. Each request is characterized by a Worst-Case Execution Time ($WCET$) requirement, an applicable deadline, and a period. Each service generates a potentially infinite sequence of requests. Successive requests arrive with a non-zero number of units of time apart.

Real-time scheduling is the process that decides where to allocate arriving requests and when to start and/or stop the execution of every request so that the timing constraints of all requests are met and other performance optimization criteria, if there exist any, are also achieved.

Real-time system can be classified according the timing constraints into soft [96–98] and hard real-time systems [63, 99]. Soft real-time systems allow requests to miss a few deadlines. However, missing more than a certain number of request's deadlines within a specific period (e.g., deadline miss ratio) degrades the performance of the soft real-time system. On the other hand, catastrophic effects may occur when violating a single deadline in hard real-time systems, such as a nuclear reactor monitoring system or traffic control software [63, 64].

Scholars also classify real-time scheduling, according to the time at which the scheduling decisions are made, into static [70, 100] and dynamic [101–103] real-time scheduling. Static real-time scheduling algorithms make the scheduling decisions offline, according to the timing constraints of requests (e.g., arrival time, deadline, response time, etc.). The static scheduling decisions never change until all requests complete their executions. Rate Monotonic Scheduling (*RMS*) is an example of static real-time scheduling methods. On the other hand, dynamic scheduling approaches make the scheduling decisions online according to changes in the request's run-time information (e.g., earliest deadline, most recent arrivals, etc.) An example of this type of real-time scheduling is the Earliest Deadline First (*EDF*).

In the non-preemptive scheduling, high priority requests cannot interrupt lower priority requests until the last finishes its execution. Static real-time scheduling policies are, thereupon, non-preemptive [74, 104]. Dynamic real-time scheduling, on the other hand, can be non-preemptive [74, 104] or preemptive [105, 106].

Real-time schedulers can be a single-core [66, 67] or on a multi-core [68–71]. A single-core scheduler has to decide the sequence of requests execution. On the other hand, a multi-core scheduler has to decide where to allocate each request and when each request is to be executed.

Regardless of the type of the real-time scheduler, It must always guarantee the timing constraints of the real-time tasks to improve the predictability of the system, which could result in performance degradation and a low throughput [107]. It is, thus, necessary to ensure that sufficient resources are available for each real-time request all the time [108].

2.2 Scheduling time-sensitive services in cloud platforms

Although cloud platforms are capable of delivering real-time services [109, 110], service scheduling in cloud platforms is different from real-time scheduling in embedded systems due to several factors, such as the nature of cloud workload, type of cloud platforms, and the design objectives.

- *Cloud workload*

  Real-time requests should have deterministic and well-defined parameters (e.g., arrival time, response time, etc.) so that hard real-time schedulers can assure that no request will miss its deadline. On the contrary, cloud workload arrival and execution times are often characterized by general distributions and hence cloud services need probabilistic or stochastic schedulers to accommodate cloud requests with random or statistical time requirements [65].

- *Cloud platforms*

  Scheduling real-time requests in embedded systems occurs on a single core's level [66,67] or on a multi-core platform's level [68–71]. On the other hand, cloud schedulers can schedule requests onto a single server [72–74], a cluster of servers [75,76], a single data center [77, 78], or a set of geographically distributed data centers [79]. The cloud service provider accordingly has to consider other external parameters before making a scheduling decision, such as network delay, communication cost, power consumption, etc.

- *Design objectives*

  Real-time scheduling relies on the worst-case analysis to avoid missing any deadline which could be catastrophic in several systems, e.g., missing a deadline in an automatic braking system in modern cars. Contrarily, cloud scheduling aims to provide statistical guarantees (e.g., 95% of cloud requests must complete before a deadline). Moreover, regardless of the type of scheduler adopted in a real-time system, all schedulers must guarantee the timing constraints to improve the predictability of the system even if the scheduling decision results in performance degradation [107]. On the contrary, the goal of cloud schedulers is to optimize the utilization and other performance metrics of the whole system as the system throughput and average response time [48, 111].

Next, we discuss the research on the power-aware scheduling of time-sensitive requests in the cloud. We categorize those types of research into power-aware cloud resource management and cloud resource allocation with *QoS*-awareness.

2.2.1 Power-aware cloud resource management

Countless efforts have been made to reduce power and energy consumption in service-oriented computing systems [112]. We can categorize those researches into different abstraction levels and/or according to different criteria. For example, according to the scale/type of the computing systems, Cai et al. in [50] categorized the energy-aware techniques applicable for servers [68, 113], clusters [57, 59, 114], data centers [45, 115] and the cloud [116].

Power/energy aware approaches can also be classified according to the different resource types, such as CPUs [45, 117], memory [84], storage devices [118], and/or network [30]. Since CPUs usually acquire the highest power consumption among all resource types [115], we focus on reviewing the works that focus on improving power consumption and energy cost of CPUs in cloud data centers using techniques including Dynamic Voltage Scaling and Dynamic Frequency Scaling (DVS/DFS), virtualization, and server consolidation [119].

Dynamic Voltage Scaling and Dynamic Frequency Scaling (DVS/DFS)

Dynamic Voltage Scaling and Dynamic Frequency Scaling (DVS/DFS) has been a powerful conventional technique for adaptive performance and power dissipation adjustment to achieve power efficiency [57, 114]. Hwang et al. showed that the maximum energy savings in virtualized multi-core servers can be achieved when combining the DVS/DFS methods and the consolidation algorithms [68]. Beloglazov et al. [116] introduced a global-and-local layer approach to make virtualized servers more power-efficient by adjusting the frequency and voltage of processors according to VM's utilization. Likewise, Kim et al. [114] proposed *DVFS*-enabled, with both time-shared and space-shared, cluster scheduling policies for a bag of tasks to reduce power consumption and to meet the deadline requirements of end-users.

Although many researchers and engineers acknowledge that *DVFS* scheduling algorithms are robust and energy-saving solutions on the server's level, they cause many challenges in the current cloud data centers. For instance, *DVFS* are architecture-dependent and they may not achieve their best power/energy-saving when applied to the current heterogeneous cloud data centers.

Virtualization and VM migration

As virtualization technology evolved as a norm in today's data centers to amplify resource usage through running multiple VMs on a single server, VM migration has been widely employed to optimize server utilization and to reduce power consumption [29]. In [120], Mastroianni et al. statistically modeled and analyzed the effects of VMs allocation and migration on minimizing the number of powered-on servers, and on reducing power consumption in data centers. Zhen et al. [29] introduced the concept of skewness to measure the unevenness in the servers' multidimensional-resource utilization.

VM live migration, nonetheless, requires a delay and service interruption that can degrade the overall system performance and availability, and consequently leads to *SLA* violations [121].

Server consolidation

In conjunction with VM migration, server consolidation is of a special interest among efficient resource allocation policies [85]. Server consolidation, comparatively to *DVFS*, improves resource utilization without demanding excessive hardware resources, and it is easy to implement and to deploy [120].

Now that, server power consumption is not exactly proportional to its utilization, and a server may waste a substantial amount of power even when it is idle [120], server consolidation methods pack running VMs on a smaller number of physical servers and/or turn off the rest, to minimize the total power consumed by those servers [58, 113, 120]. In [113], Verma et al. presented a two-dimensional, i.e., memory-based and CPU-based, consolidation strategy in which decisions are made based on the correlation among different workloads. In [113], Pinheiro presented an algorithm to dynamically turn servers on and off according to the imposed load in computing clusters. Chase et al. [122] reduced the energy consumption of server clusters by degrading services according to their *SLA*, when power consumption or thermal dissipation exceeds a certain limit.

Next, we discuss the works related to cloud resource allocation with *QoS*-awareness.

2.2.2 Cloud resource allocation with QoS-awareness

Whereas saving power/energy is important, service providers must also ensure that their services can satisfy the *QoS* requirements of cloud users, such as response time and/or deadline miss ratios. For instance, a recent report has found that a 100ms extra delay costs Amazon 1% of sales revenue [123]. The problem becomes more challenging with interactive workload types [61] like online gaming [21] and multimedia streaming services [62]. These online interactive services have soft-timely constraints [61], in that service providers must guarantee that a predefined percentage of them meet their deadlines. Otherwise, service providers fail to keep up with their *SLA* [20].

VM placement methods with performance-interference awareness were introduced in [124] to improve the performance of VMs and the utilization of servers. Resource overbooking, i.e., allocating more resources than the actual available capacity to raise service provider profit, with different real-time constraints is presented in [61]. Energy-aware resource allocations with response time and end-to-end time guarantees were introduced in [58] and [59], respectively. Greenberg et al. [30] studied the costs of cloud data centers and recommended developing new management systems within and across geographically distributed data centers with the focus on network agility to improve their efficiency and end-to-end performance.

SLA-aware workload consolidation methods have been proposed to achieve higher dynamic power efficiency and to overcome the under-utilization problems resulting from applying the over-provisioning policies [86]. Lee et al. [125] developed a pricing model, based on the queue model M/M/1/PS, and used it to establish profit-driven scheduling algorithms with *SLA* for the cloud-dependent services.

To assure the *QoS* requirements of cloud services with requests from different classes of *QoS* demands, it has been a common approach (e.g., [59]) to serve requests with the same *QoS* requirements on the same VM. When all requests on the same machine have the same *QoS* requirement, different types of *QoS* can be captured by a single variable, such as provisioned resources (e.g., [113] [120] [116]), required processing speed (e.g., [57]), or latency (e.g., [58] [59]). Although this approach simplifies the resource management problem to guarantee one specified *QoS* criteria, it excludes requests that can share resources, and the overall resource usage can be rather inefficient, as illustrated later in this dissertation.

Almost all previously mentioned works implicitly assumed that all accepted requests must receive service, even if they do not meet their *QoS* conditions. We show in this dissertation that if we can judiciously discard the requests that are likely to miss their deadlines, we can significantly improve resource usage without compromising *QoS* conditions.

Next, we discuss the related works to VM allocation in cloud platforms with and without security awareness.

2.3 VM allocation in cloud platforms

As cloud services entered in each sector of our personal and professional lives, maximizing resource usage and minimizing power consumption in cloud data centers using efficient allocation policies became a necessity. Many allocation approaches, based either on traditional optimization methods (e.g., Mathematical Programming (*MP*) [126], evolutionary programming method [127], and fuzzy control [128]) or on a variety of different heuristics (e.g., [129]), have been proposed for resource allocation in cloud platforms for applications with different characteristics, requirements, and optimization goals. For example, Beloglazov et al. [126] suggested several VM migration algorithms to improve CPU utilization. When a server has a CPU usage below a predefined threshold, all VMs hosted on that server must be migrated to other servers. They also introduced several more algorithms to minimize the number of VM migrations using upper utilization thresholds as the Minimization of Migrations (*MM*) policy, the Highest Potential Growth (*HPG*) policy, and the Random Choice (*RC*) policy. Their experimental validation showed that increasing the lower-utilization thresholds, increases the *SLA* violations, while reducing the server's energy consumption. Proposing VM consolidating policies that optimize resource usage is vital for cloud providers, yet those allocation policies are hypothetical because they ignore the cybersecurity effects of the consolidation decisions.

In this section, we review the research related to VM allocation methodologies in cloud platforms. We can classify those related works into four major categories that are directly related to our research focus in this dissertation: Vector Bin Packing (*VBP*) based cloud resource allocation, game theory-based cloud resource management, cloud cybersecurity countermeasures using the Game theory, and security-aware cloud resource allocation approaches.

2.3.1 Vector Bin Packing (VBP) based cloud resource allocation

The VM allocation problem is a well-known NP-hard problem, as we show later on, and hence heuristics, such as Vector Bin Packing (*VBP*), have been adopted heavily to solve similar problems with multiple optimization goals [130]. In a *VBP* heuristic, there is a weight function applied to the items so that each item is assigned a single scalar, based on which, the standard bin packing can be used to sort those items. Wood et al. [92] introduced a multi-dimensional First Fit Decreasing (FFD) approach to lively migrate VMs out of overloaded servers. Their approach considers multiple CPU, memory, and network resource demands of each VM. Panigrahy et al. [131] proposed several *VBP*-based methods for VM allocation with different resource demands and optimization goals. Beloglazov et al. [126] suggested several VM migration algorithms to improve CPU utilization and demonstrated that increasing the lower-utilization threshold increases the Service Level Agreement (*SLA*) violations while reduces the server's energy consumption. In [128], the authors used combinatorial and multi-objective optimizations to optimize resource usage in a two-level control system that allocates workloads from virtual to physical resources. Using a local and a global fuzzy controller, they tried to minimize power consumption, thermal dissipation, and a peak temperature of the system. In [132], the authors introduced an application placement controller that consolidates applications according to the ratio of their CPU to memory demands. Microsoft's Virtual Machine Manager used in Azure applies the Dot-Product and Norm-based Greedy heuristics [133]. The authors in [133] proposed new geometric heuristics that run nearly as fast as *FFD*. We are, however, not aware of any prior work that employs *VBP* to deal with security requirements during a VM allocation process.

2.3.2 Game theory-based cloud resource management

Researchers use the Game theory to study and understand the interaction among economic, social, or military entities. The goal is to find a stable pair of static (e.g., pure) or mixed (e.g., dynamic) strategies which are called a Nash Equilibrium (*NE*) strategy profile. We mean by –stable– is that none of the entities involved in the game has any incentive to deviate from its equilibrium strategy. According to Nash 1950 [134], any game with a limited number of strategies must have at least one such equilibrium. The Game theory thereupon can be used to formulate VM allocation problems into a non-cooperative two-player game framework. Our rationale is that the Game theory offers "mathematical models of conflict and cooperation between cooperative and noncooperative intelligent rational decision-makers", [135]. Although the Game theory is a theoretical approach in the sense that it assumes the attacker has a comprehensive knowledge about all the information related to the infrastructure that helps him/her maximize his/her cyber gain (e.g., informative game), we can positively use it to model the scenarios in which the worst potential cyber-attack occurs on the clusters of cloud providers.

The three essential criteria to consider when designing an equilibrium-based VM consolidation algorithm are soundness, completeness, and computational efficiency [136]. First, the algorithm is sound if any solution it returns is, in fact, a solution. The algorithm is complete if there exists only a single solution; the algorithm finds it [137]. The third essential criteria for our VM consolidation problem is computational efficiency; that is the algorithm's worst-case computation time is polynomial for any number of VMs and servers.

Several works employed the Game theory to model the resource optimization problem in cloud platforms [127, 129, 138]. Wei et al. [127] adopted the Game theory to solve a *QoS* constrained resource allocation problem across a cloud-based network. Kunsemoller et al. [129] elaborated on cloud economics benefits for businesses using a game-based cloud model of an *IaaS* economy including the dynamics of pricing and usage. Pillai et al. [139] proposed a VM allocation policy onto cloud platforms, based on the principles of coalition formation and the uncertainty principle of the Game theory. They illustrated that the coalition-formation of the VMs leads to higher resource utilization and higher request satisfaction. Teng et al. [140] suggested a new Bayesian pure *NE*-based resource management policy assumes heterogeneous and distributed resources, cognitive behaviors of cloud consumers, non-perfect information, and dynamic successive allocation. They proved that the resource price would converge to the optimal rate at the end of the gambling sequence. Jalaparti et al. [138] similarly employed the Game theory to optimize resource efficiency, and pricing policies by modeling the client-provider and client-client interactions, respectively. They introduced multiple heuristic algorithms with near-optimal allocation and pricing policies compared to the fixed-pricing strategies used today by cloud providers, such as in Amazon EC2. However, all the above ignored the cybersecurity effects of their allocation methods.

2.3.3 Cloud cybersecurity countermeasures using the Game theory

As more and more organizations, companies, and private users move their computing facility to cloud data centers, there have been increasing interests and concerns in the cybersecurity problem in the cloud (e.g., [90, 141]).

Several works focused only on studying the types of cybersecurity attacks that are resulted from the commonly shared infrastructure of the cloud among several applications and users. Side-channel attacks are one of the most popular types of cybersecurity attacks on cloud infrastructure. Several countermeasures were proposed in the literature to mitigate or prevent side-channel attacks [51–53]. The proposed methods include modifying or tuning up the infrastructure to prevent hackers from extracting information about private keys (i.e., secret key extraction attack [51]), preventing attackers from verifying co-residence with the victim's VM [52], or introducing a new infrastructure design (e.g., mitigate the threat of timing channels by eliminating high-resolution clocks [142], or adding latency to potentially malicious operations [52, 142]).

Kamhoua et al. [33] used a non-zero-sum game framework to model the VM-to-VM interdependent cybersecurity risks between an attacker and two users in the cloud. They showed that the existence of *NE* strategy profiles depends on the probability that the hypervisor is compromised, after a successful attack on one of the users, and the total cost of the user's security investments. They also irrationally concluded that there exists no *NE* strategy profile when all the users in the cloud fully invest in cybersecurity countermeasures. Et al. Kwiat [55] applied the same cybersecurity model introduced by Kamhoua to a different allocation problem. They considered a game between an attacker and three users. The first user never invests in security and is always allocated to the first insecure hypervisor. The second user invests in security countermeasures and is always assigned to the second secure hypervisor. In this dissertation, we use a zero-sum game framework between an attacker and a provider to model and analyze the VM-to-VM mutual interdependent cybersecurity risks in cloud data centers.

Unlike [33, 55], wherein the defender strategies are to invest or not to invest, we assume that all VMs have different resource requirements and cybersecurity countermeasures installed. The provider's strategy is to choose the allocation policy that minimizes his/her loss under a worst-case cybersecurity attack on a cloud cluster with a limited set of resources.

2.3.4 Cybersecurity-aware cloud resource allocation

Cybersecurity is the most critical concern when adopting cloud computing [42, 44]. The commonly shared infrastructure of the cloud among VMs and users, unfortunately, exposes cloud workloads to several cybersecurity risks, such as side-channel attacks [51–53] and VM-to-VM interdependent cybersecurity risks [39].

The side-channel attacks occur using virtual machine collocation. The successful collocation allows the attacker to build different kinds of side channels to extract private information about the victim's VMs (e.g., the victim's workload, traffic rate [52], cryptographic keys [51], etc.) that eventually enables him/her to launch a successful attack. Several software-based and hardware-based methods were proposed to address this type of attacks [88, 89]. Software level methods are usually limited in their capabilities to boundaries of the hosting VM or unable to keep up with all the new attack techniques proposed by hackers [90]. Rao et al. [143] used the Game theory to search the ability of a cloud computing provider to guarantee a given capacity C with a particular probability P given a physical or cybersecurity attack on his/her data center. They proposed the use of reinforcement strategies to decrease the attacker's utility. While many works focused on VM migration to maximize resource utilization and minimize power consumption, et al.

Zhang [90] is the first to develop a formal and quantified migration strategy in the cloud to improve cybersecurity against collocation attacks and with accepted costs.

[141] used the Game theory to model the co-location attack between attackers and a provider. The attackers try to collocate their VMs with target VMs on the same physical server and exploit side-channel attacks to extract private information from the VMs of victims. The provider aims to minimize the attackers' possibility of collocating their VMs with the target VMs while maintaining a satisfactory workload balance and low power consumption for the system. The provider strategy is to choose among four different allocation policies: servers with the least number of VMs, servers with the most number of VMs, random, and round-robin.

Hardware-level methods proposed to face side-channel attacks are costly and impractical as they required modifying the architecture of the cloud infrastructure [88, 89]. Furthermore, dynamic VM allocation, i.e., VM live migration, can significantly minimize the attacker chances of colocating his/her VM with the victim VM [90, 91]. Although dynamic VM allocation maximizes resource utilization [92] and minimizes cybersecurity risks resulted from side-channel attacks [51] through live VM migrations, it brings about a migration overhead results in performance degradation and *SLA* violations [49].

In the VM-to-VM Interdependent Cybersecurity (*IC*) attacks, an attacker can compromise the hypervisor after a successful attack on one of its vulnerable VMs [33, 55]. He/she consequently can compromise all other coexisted secure VMs on that hypervisor. VM multiplexing methods, which are extensively used in cloud data centers, expose VMs with sensitive data or high values to interdependent cybersecurity risks which allow the attackers to bypass any security measures applied to those critical VMs by indirectly compromising them using a less secured colocated VMs [33, 52]. While allocating VMs with different security requirements to different servers incurs lower security risks, it exacerbates resource usage and energy cost. The challenging question accordingly is –how can the providers optimize the trade-off between minimizing the *IC* risks and improving his/her energy cost and operating expense?–.

2.4 Summary

In this section, we discuss the difference between real-time and cloud scheduling for time-sensitive requests. We also discuss the works about the power-aware scheduling of time-sensitive cloud requests which we categorize into cloud resource management without assuring *QoS* guarantees and cloud resource allocation with *QoS*-awareness. Finally, we review the research about our VM allocation methodologies in cloud platforms. We further classify those works into Vector Bin Packing (*VBP*) based cloud resource allocation, game theoretic-based cloud resource management, cloud cybersecurity measures using the Game theory, and security-aware cloud resource allocation approaches.

In the next section, we describe our cloud prototype which we designed and implemented to validate our findings in this dissertation.

CHAPTER 3

THE GREEN CLOUD COMPUTING PROTOTYPE (GCCP)

In this section, we first state our motivation and shed light on the importance of building an experimental cloud platform. We then describe the hardware and software components of the platform. Second, we show how to use our platform in analyzing the performance of actual cloud clusters. Finally, we list various types of open-source benchmarks that the platform can run and conclude the chapter.

3.1 Motivation

Purely theoretical analysis and study of the performance of cloud platforms and the behavior of virtualized systems undergo numerous challenges that prevent scholars from providing realistic and applicable conclusions [144]. Variations in Virtual Machine (VM) Managers, hypervisors, virtualization scenarios, and cloud workload types makes it almost impossible to develop a single universal benchmark tool that can carry out performance testing tasks on any cloud platform. Furthermore, new simulation models were developed to study cloud data centers. However, those models fall short in considering the performance interference and interdependence among hosting hypervisors, VMs, and applications that compete over the shared resources (e.g., processing, memory, storage, and network). Furthermore, simulation develops a margin of errors that contribute to inaccurate validation results.

We hence realize the need to build an experimental validation platform that allows us to generate synthetic or to import open-source cloud-specific benchmarks when validating our proposed resource management schemes.

3.2 Platform characteristic

In this section, we introduce the Green Cloud Computing Prototype (*GCCP*) that conforms to the industrial standards applied in practice. Fig. 3.1 shows the different components of which the *GCCP* consists. We built *GCCP* so that it can have the following characteristics:

- *Industry compatibility*

  The architecture model of *GCCP* conforms to the model introduced by IBM in [25], whereas the organization of its infrastructure model conforms to the well-known cloud providers, such as Google Compute Engine, Amazon EC2, Rackspace, and Microsoft Windows Azure. The prototype can import and integrate any of the industrial-compatible cloud software or benchmarks (e.g., the cloud orchestrator OpenStack, different hypervisors, Memchashed benchmarks, etc.) as we illustrate later in this chapter.

- *Modularity*

  Our cloud prototype consists of individual functional modules that are implemented in Java and running on management nodes.

- *Full automation for better usability*

  The system workflow is fully automated and controlled using Python scripts at the system level, and bash scripts at the operating system level which makes using the prototype easy for users from any background.

- *Versatility*

  The prototype is equipped with a dedicated workload submodule that is capable of generating cloud or general-purpose workload with different characteristics and functionality allowing it to be used for several purposes.

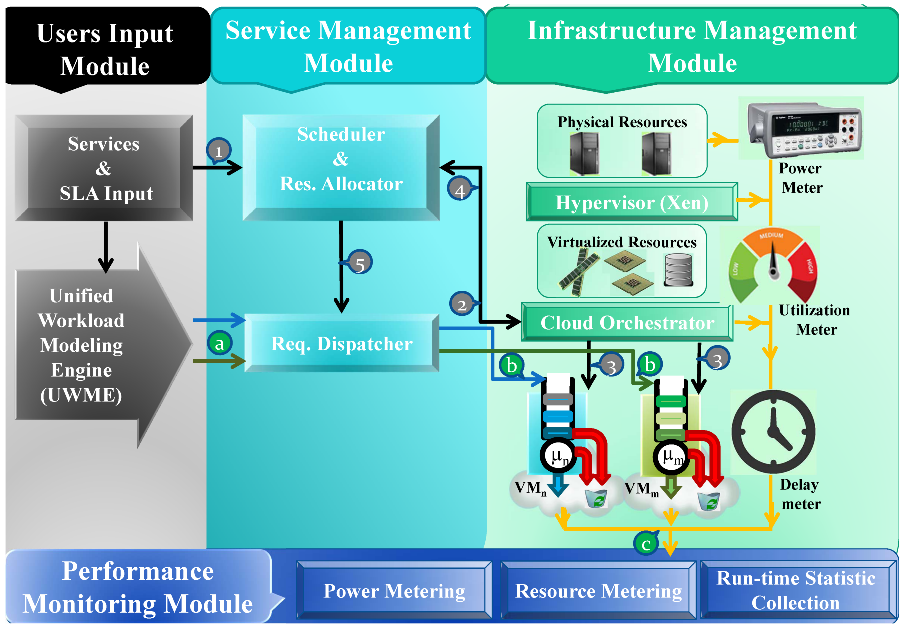


Figure 3.1: High-level representation of the Green Cloud Computing Prototype (*GCCP*). The cloud prototype consists of four functional modules that are implemented in Java and running on management nodes. The system workflow is automated and controlled using Python scripts at the system level, and bash scripts at the operating system level.

- *Scalability*

  In this chapter we show how to deploy *GCCP* software components over several personal desktop computers inside a small classroom or over a typical cloud cluster, which consists of several powerful rack servers, exists in a server room or actual data center.

We describe each of the hardware and software components of *GCCP* next.

3.2.1 GCCP's hardware components

GCCP consists of two management and two compute physical machines. The two management nodes (e.g., VMs) are launched using the open-source Kernel Virtual Machine (*KVM*) hypervisor and managed using the open-source Webvirtmgr. *KVM* hypervisor is hosted onto two Dell Precision T1500 machines with Quad-Core Intel i-5 CPU, 16 GB, 1333 MHz DDR3 memory, and a 300 GB SATA Disk Drive. The management VMs run Ubuntu Server Linux 12.04.5 LTS Precise Pangolin release with a kernel version 3.2.0.76.

We installed Citrix XenServer 6.5 platform, which is based on the hypervisor Xen [145], to manage the physical resources onto two compute nodes. Each compute node is an HP Workstation Z800 with two Intel Xeon Six-Core E5645 (2.40 GHz, 12 MB cache), 1333 MHz DDR3 memory of size (32 GB), and 1 TB disk space.

3.2.2 GCCP's software components

GCCP consists of four functional modules implemented in Java and running on management nodes. The system workflow is automated and controlled using Python scripts at the system level, and bash scripts at the Linux nodes level.

Next, we describe each of the functional modules in detail.

User input module

The $UserInput$ module allows users to define service types having request classes with different *QoS* requirements. The synthetic request classes imitate the requests of a cloud broker under different Service Level Agreements *SLAs* via the *Service and SLA Input* submodule. On the other hand, the submodule *Unified Workload Modeling Engine (*UWME*)* models and generates request instances of a specific service type and class according to parameters and *QoS* constraints defined in the *Service and SLA Input* submodule. Specifically, *UWME* is a Java tool that generates workloads with desired timing, functional and computational characteristics. As shown in Fig. 3.2, *UWME* is essentially a server/client model program that can generate workloads with different stress levels, which can be identified by the number of processes to be launched at a specified VM. The *UWME*

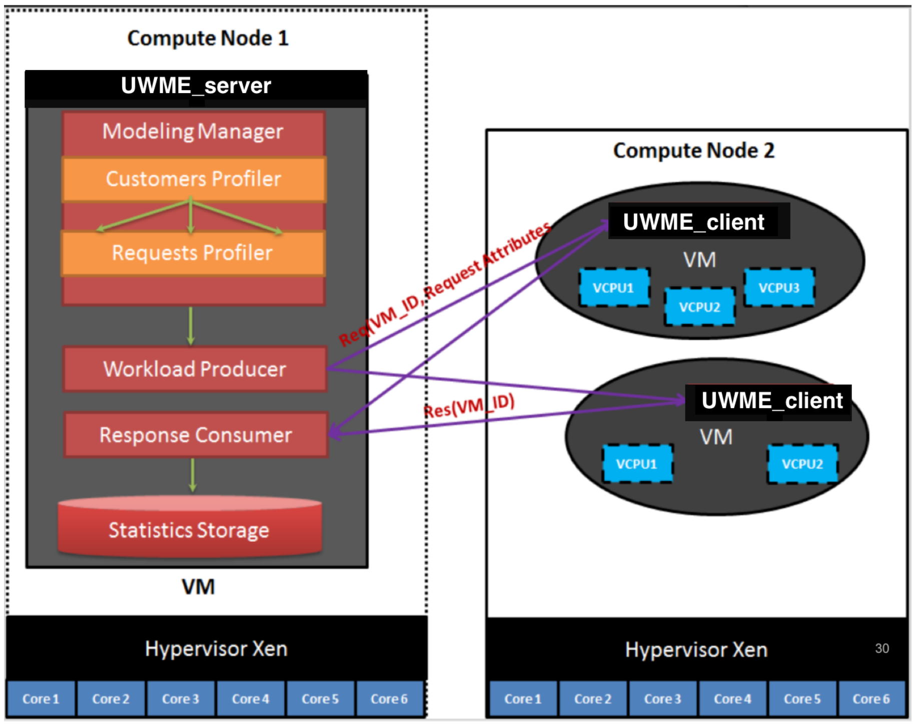


Figure 3.2: Unified Workload Modeling Engine (UWME). UWME is essentially a server/ client model program that can generate workloads with different stress levels.

server is a VM residing in a management compute node and contains four main modules, i.e., Modeling Manager, Producers, Consumer, and Database.

All the submodules in the user input module run on allocated VMs that enable potential failure reneging and allow results logging.

In the following chapters, we employ *UWME* to generate several scientific and cloud workloads. For example, we utilize *UWME* to generate memory-intensive cloud services, which is a Matrix MULtiplication (*MMUL*) Java application, using the open source lightweight Apache Common Mathematics Library [146]. We also employ *UWME* to produce CPU-intensive cloud services, shaped with a one-dimensional Fast Fourier Transform (*1-D FFT*) Java application, using the open source multi-threaded (*FFT*) library Jtransforms [147].

In addition to our newly developed synthetic workloads, we employ *UWME* to incorporate cloud-specific cloud benchmarks. As we discuss later in this dissertation, we use the *UWME* submodule's server and clients to mimic the behavior of a Memcached server and clients. A Memcached server is a distributed memory object caching system that speeds up dynamic web applications by alleviating the delay of a single database worker (e.g., a single execution queue) that is capable of producing the data caching requests, i.e., *Get* or *Set*. The Memcached clients receive the caching requests to schedule and process them according to our proposed scheduling algorithms. The *Performance Monitoring Module* then collects the results on all VMs.

Service management module

The *ServiceManagement* module accepts inputs from the *User Input Module* and schedules/dispatches cloud workloads onto the available computing resources using the *Scheduler and Resource Allocator/Request Dispatcher* submodules. Different resource provisioning and workload scheduling algorithms can run in the *Scheduler and Resource Allocator* sub-module.

Infrastructure management module

The $InfrastructureManagement$ module is responsible for operating the hardware and software computing resources of the cloud cluster. We employ Citrix XenServer 6.5 platform, which is based on the hypervisor Xen [145], to manage the physical resources on both HP Workstations Z800 with two Intel Xeon Six-Core E5645 with a 2.40 GHz and 12 MB cache CPU, 1333 MHz DDR3 memory of size 32 GB, and 1 TB disk space. We also developed a *Cloud Orchestrator* submodule that communicates with the *Scheduler and Resource Allocator* and XenServer to automate the processes of creating VMs, provisioning VMs, configuration VMs, and to make VMs available online for the *Request Dispatcher* onto which requests are forwarded from the *UWME* submodule.

Performance monitoring module

The $PerformanceMonitoring$ module monitors the performance of the cloud cluster, collects performance statistics, and measures resource usage and power consumption via three major submodules (e.g., the *Power Metering*, *Resource Metering*, and *Run-time statistic collection* sub-modules).

The *Power Metering* submodule measures the static and dynamic power consumption of the server pool under different configurations and running conditions. To measure the actual power consumption, we used an AC/DC Fluke i410 current clamp meter with an output of 1 mVolts(mV)/Amps(A), connected to an Agilent 34401, which is a multimeter with a resolution of +/- 120 mWatts (mW). The *Power Metering* submodule automates the power reading process using a *C* program running within Ubuntu Linux 12.04.5 LTS on a dedicated Dell desktop that communicates with the multimeter through a serial cable to automatically record electrical current readings.

Consider a server pool $SP_i = \{VM_{i,1}, ..., VM_{i,m_i}\}$ allocated to physical cores $\{PCPU_{i,1}, ..., PCPU_{i,m_i}\}$, respectively. Recall that a single workstation in *GCCP* has at most 10 available physical cores –Note that we reserve 2 cores for Xen's Domain-0–. SP_i must be hosted by $\lceil \frac{m_i}{10} \rceil$ workstations, and the power consumption for SP_i is the total power consumption of these workstations. Now assume that $m_i \leq 10$. The power consumption P_i consumed by SP_i can be formulated as

$$P_i = \sum_{j=1}^{m_i} (P_{i,j}^d \varphi_{i,j}) + P_i^s, \tag{3.1}$$

where $P_{i,j}^d$ is the dynamic power when each core is 100% utilized, $\varphi_{i,j}$ is the utilization of $VM_{i,j}$, and P_i^s is the static power of the HP workstation when m_i physical cores are allocated to the VMs.

To calculate P_i^s, we measured the drawn AC current by the workstation, i.e, I^s, when all m_i VMs are idle. The corresponding static power consequently is $P_i^s = I^s \times 120V$.

To calculate $P_{i,j}^d$, we used the $UWME$ submodule to generate enough request instances such that all VMs were kept busy and achieved 100% utilization, and we afterward measured the drawn AC current by the workstation to calculate the total power consumption, i.e., static and dynamic power. The difference accordingly, between the full and static measured power, is the overall dynamic power consumption of m_i cores with 100% utilization.

We further assume that the cores that are hosting the same service type (e.g., S_i) with the same capacity size (e.g., C_i) consume the same amount of dynamic power. We thereupon divide the total dynamic power by the number of cores m_i to get $P_{i,j}^d$. As an illustration, let a server pool with a single VM. We measured its static power as $186W$. It's dynamic power consumption when running memory-intensive workload and CPU-intensive workload on a fully utilized core are $16.8W$ and $18W$, respectively.

The *Resource Metering* submodule leverages the available system tools and our newly developed user-level tools to collect and measure the computing resource usage. For example, to measure the amount of CPU usage, i.e., the processing rate of a CPU, consumed by each VM in MegaHertz (*MHz*), we can use the command $xentop$. To measure the CPU usage in Instruction Per Second (*IPS*), we can use our newly developed scripts that parse the run-time logs generated by the *Run-time Statistic Collection* submodule.

The *Run-time statistic collection* submodule collects and stores run-time statistics of the dispatched request instances in log files based on the light-weight RAM Filesystem ($Ramfs$). Information obtained by this submodule includes IDs of service types, request classes, instances and hosting VMs, along with with the instances start times, finish times, *QoS* violations, completion ratios, and so on.

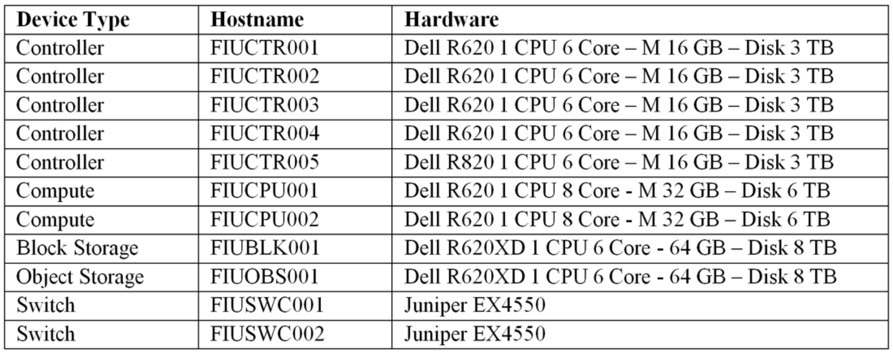
In Fig. 3.1, steps one to five illustrate the processes of initiating a new service and its request classes (step one), running a packing and allocation algorithm (step two), spawning and configuring VMs (step three), returning the IDs of a VMs to the *ServiceManager* module to be available to the server the request instances (steps four and five). Furthermore, steps *a*, *b* and *c* illustrate the processes of generating instances of a request type, dispatching them to the allocated VMs, and collecting the system performance readings upon their completion.

3.3 How to deploy and utilize GCCP with actual cloud clusters

In this section, we show how to extend the usage of *GCCP* to include analyzing the performance of actual cloud clusters. Specifically, we will replace the four physical nodes used in the implementation of *GCCP* with real rack servers used in cloud clusters. We will further replace our cloud orchestrator with the open-source cloud orchestrator OpenStack, which is adopted heavily in cloud data centers [148].

We can replace the four nodes used in *GCCP* with seven or nine rack servers (e.g., five controller nodes, two or four compute nodes, and two storage nodes), and a Juniper EX2200 Ethernet Switch [149]. The cluster should be moved to one of FIU's server rooms where reliable power source and cooling are provided. Fig. 3.3 shows the physical specification and the role of each server. There are five controller nodes (e.g., two compute nodes, a block storage node, and an object storage node [148]).

There are five different networks configured over the cluster [6,7,148]. First, a public network connects all nodes in the cluster. Second, an internal private network serves the VMs and Quantum plugins control it. Third, a management network connects all OpenStack components [148]. Fourth, an object storage network connects the object storage node to the server running the load balancer (HAproxy) [150]. Finally, a block storage



Device Type	Hostname	Hardware
Controller	FIUCTR001	Dell R620 1 CPU 6 Core – M 16 GB – Disk 3 TB
Controller	FIUCTR002	Dell R620 1 CPU 6 Core – M 16 GB – Disk 3 TB
Controller	FIUCTR003	Dell R620 1 CPU 6 Core – M 16 GB – Disk 3 TB
Controller	FIUCTR004	Dell R620 1 CPU 6 Core – M 16 GB – Disk 3 TB
Controller	FIUCTR005	Dell R820 1 CPU 6 Core – M 16 GB – Disk 3 TB
Compute	FIUCPU001	Dell R620 1 CPU 8 Core - M 32 GB – Disk 6 TB
Compute	FIUCPU002	Dell R620 1 CPU 8 Core - M 32 GB – Disk 6 TB
Block Storage	FIUBLK001	Dell R620XD 1 CPU 6 Core - 64 GB – Disk 8 TB
Object Storage	FIUOBS001	Dell R620XD 1 CPU 6 Core - 64 GB – Disk 8 TB
Switch	FIUSWC001	Juniper EX4550
Switch	FIUSWC002	Juniper EX4550

Figure 3.3: Physical infrastructure specifications for a typical middle-size cloud cluster

network that connects compute nodes to the block storage node. Controller, compute and block storage nodes are connected to the management and the public networks via two bonded (10) Gigabit Ethernet interfaces.

Each controller node has a single 6-core CPU, (16) MB of RAM and two (1) TB hard disks with Redundant Array of Inexpensive Disks (*RAID*) controller uses a *RAID*-5 level. the compute nodes use a *RAID*-5 level too. They, however, have twice the memory and the storage capacity that the controller nods have. Object and block storage nodes share the same physical specifications of a single 6-core CPU, (64) MB of RAM, and eight (1) TB hard disks. Nevertheless, the object storage node uses a *RAID*-10 level for better data transfer rates, while the object storage node uses a *RAID*-1 level.

We can replace the *GCCP*'s cloud orchestrator with the open-source cloud orchestrator, OpenStack, and deploy it on the controller nodes. OpenStack [14, 15, 148] is a fully-modular, open-source software architecture helps to provide on-demand processing, storage, memory, and network bandwidth resources [148] over cloud clusters. OpenStack can dynamically scale up and down to meet service requirements, while accordingly adapting to intensive workload situations [6, 7]. OpenStack consists of several individuals, yet integrated, and distributed software projects (e.g., Horizon, Keystone, Nova, Quantum, Cinder, Glance, and Swift [148]).

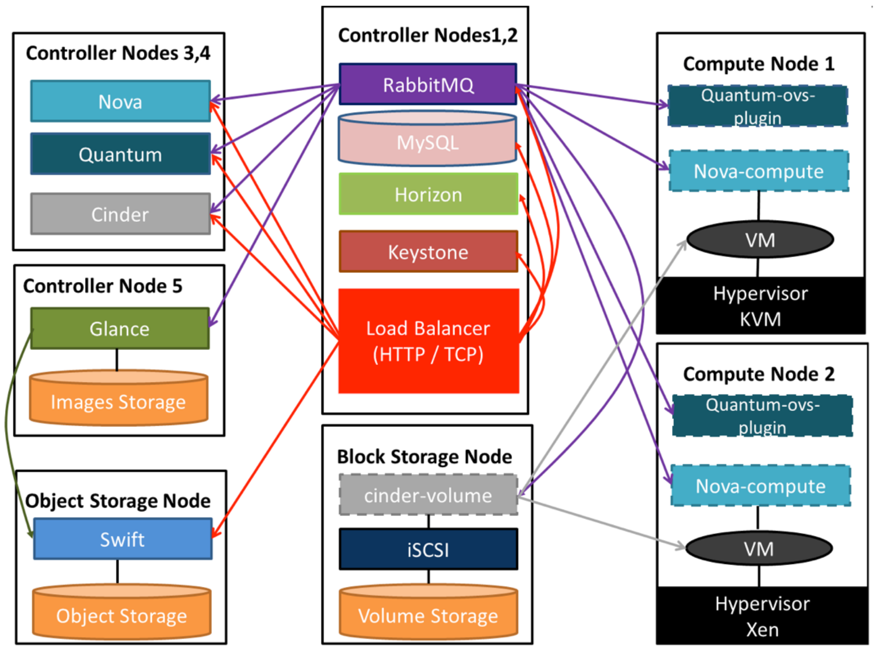


Figure 3.4: GCCP's high availability deployment model. Deploying a cloud orchestrator (e.g., OpenStack) using load balancing and controller redundancy as recommended by Mirantis [6] and Rackspace [7].

- *OpenStack dashboard (Horizon)*

  Horizon is the OpenStack dashboard which provides a web user interface to manage OpenStack services.

- *OpenStack Identity (Keystone)*

  Keystone is the identity and authentication project that provides identity, token, and catalog for OpenStack services.

- *OpenStack compute (Nova)*

  Nova is the OpenStack compute project. Nova provides Infrastructure as a Service (*IaaS*) by provisioning VMs. Nova also manages OpenStack VMs using Nova worker called (Nova-compute) via hypervisor's API's, such as XenAPI for XenServer [145], libvirt for *KVM* or QEMU [151], and VMwareAPI for VMware [152]. Nova consists of Nova RESTful API (nova-api), Nova-database (nova-db), message queue (RabbitMQ), Nova scheduler (nova-scheduler), Nova conductor (nova-conductor), and Nova compute (nova-compute) processes.

- *OpenStack networking (Quantum)*

  Quantum manages OpenStack network topology. It provides Network as a Service (*NaaS*) for OpenStack services and for the VMs operated by Nova. Quantum consists of Quantum server (quantum-server), Quantum plugins and agents (quantum-dhcp-agent and quantum-openvswitch-agent), and quantum database (quantum-db) processes.

- *OpenStack block storage (Cinder)*

  Cinder manages the volumes attached to the VMs in OpenStack. Cinder consists of Cinder API (cinder-api), Cinder scheduler (cinder-scheduler), Cinder database (cinder-db), Cinder volume (cinder-volume), and message queue (RabbitMQ) worker services.

- *OpenStack image service (Glance)*

  Glance is an image repository that discovers, registers, stores, deletes and retrieves all virtual images in OpenStack. Glance contains Glance API (glance-api), Glance database (glance-db), and Glance registry (glance-registry) processes.

- *OpenStack object storage (Swift)*

  Swift is a redundant object storage tool developed by Rackspace [7]. It offers a methodology for storing and retrieving large scale of data objects through API web services.

The two compute nodes in *GCCP* can run two different hypervisors: *KVM* [151] and Xen [145] respectively. Both compute nodes host the VMs that are being launched by OpenStack compute services. OpenStack modules and *KVM* hypervisor run on top of Ubuntu Linux Precise release with kernel version 3.12-4. Object storage services, such as Swift and Ceph, can run on the object storage node, while block storage services, such as Cinder, Ceph or iSCSI, can run on the block storage node. Fig 3.4 shows our high availability cloud deployment model of the OpenStack using load balancing and controllers redundancy based on industrial standards [6, 7, 148].

Next, we review several open-source cloud benchmarks that are readily deployable over *GCCP*.

3.4 List of cloud and virtualization benchmarks that are readily deployable over GCCP

The following open-source cloud and virtualization benchmarks and tools can be easily modified and deployed over *GCCP* to conduct experiments under different orchestrated scenarios.

- *BenchVM:*

  BenchVM [153] is an open-source virtualization benchmark suite. It was originally developed to help perform automated testing and comparison between *KVM* and Xen Hypervisors in terms of the overall performance, performance isolation and scalability [144].

- *Virtbench:*

  Virtbench is a set of smaller benchmarks [154]. It is designed to help developers and engineers optimize hypervisors. Virtbench launches four VMs, install the Virtbench client on each one, then runs various tests and collects the results.

- *vConsolidate:*

  The vConsolidate benchmark was developed mainly by Intel. It measures the performance of aggregated servers in different consolidation scenarios. It spawns up four parallel VMs with separate workloads running a Java server, a Web server, a mail server, and a database server. Those VMs along with their workloads are called Consolidation Stack Unit (*CSU*), and they are used to model the performance of each application in different server consolidation scenarios.

- *SPEC Benchmark Suit:*

  The Standard Performance Evaluation Corporation (*SPEC*) is a non-profit organization that concerns with developing High-Performance Computing benchmarks [155]. For example, *SPECpower_ssj*2008 is the first industry-standard power benchmark. It evaluates the power parameters in a single server or over a cluster of multiple servers. Other benchmarks that were developed by *SPEC* for power-performance analysis are *SPECvirt_sc*2013, *SPECvirt_sc*2010, *SPECweb*2009 and *SPEC OMP* 2012.

- *Rally:*

  Rally is a member of OpenStack projects family [156]. Rally or Benchmark as a Service combines multiple components that cooperate to perform automated and reproducible tests over different deployment scenarios.

There are a large number of other testing and monitoring tools and related projects that can perform the benchmarking task with our *GCCP*. For example, Autotest is a fully automated framework for testing the Linux kernel that can be used to perform several virtualization testing tasks [157]. Sensu [158] and Najios [159] likewise are open source monitoring frameworks that can evaluate the performance of any virtualized system.

3.5 Summary

In this chapter, we introduce the Green Cloud Computing Prototype *GCCP* which is equipped with workload generator tools based on open source cloud software (e.g., [148]), and closely mirrors industry-compatible cloud platforms. We also show how to extend the functionality of *GCCP* to actual cloud clusters. Finally, we list many general and cloud-specific benchmarks that *GCCP* can utilize.

In the next chapter, we discuss our contribution for power-aware cloud workload consolidation with *QoS* guarantees and describe how we employed *GCCP* to validate our proposed algorithms using in-house synthetic and open-source cloud-specific workloads.

CHAPTER 4

WORKLOAD CONSOLIDATION FOR CLOUD DATA CENTERS WITH GUARANTEED QUALITY OF SERVICE USING REQUEST RENEGING

Cloud data centers are widely employed to offer reliable cloud services. However, low resource utilization and high power consumption have been considerable challenges for cloud providers. The accelerated rise in need for affordable cloud services magnifies the obstacles for proficient resource management policies. In this chapter, we investigate how to improve resource utilization and power consumption in cloud data centers when delivering services with statistically guaranteed Quality of Service (*QoS*). We assume that the cloud service provider allocates different types of services, each of which has request classes with different *QoS* requirements. Different from the traditional approaches that distribute workloads with different *QoS* levels on different Virtual Machines (VMs), we introduce a method to pack requests of the same service type, even with different *QoS* requirements, into the same VM, and to remove potential failure requests in time to improve resource usage and energy cost. We formally prove that our algorithm can statistically guarantee *QoS* conditions in terms of deadline miss ratios. We develop a cloud prototype to validate our proposed methods and algorithm empirically. Our intensive experimental results confirm that our approach can significantly outperform other traditional approaches in terms of *QoS* guarantees, power consumption, resource demand, and electricity cost.

4.1 Introduction to the research problem

Cloud computing [15] has recently become the dominant trend for the continuous delivery of online services over the internet using large-scale data centers. In the meantime, the relentless increase in demand for different services [20, 21], in both personal and professional life sectors with high Service Level Agreements (*SLAs*), has posed critical challenges on cloud service providers. Maintaining excessive computing resources won't effectively address this problem, as it can lead to tremendously high power consumption rates and energy costs.

Exorbitant power consumption rates and energy costs are among the main concerns in cloud infrastructure facilities. Cloud service providers strive to enrich competing markets with more reliable, yet less costly, services to a modern world handles Everything as a Service (*EaaS*) [15]. Whereas the price for online services decreases and the performance of computing systems increases at almost the same rate as Moore predicted five decades ago, the performance-per-watt of computing components increases at a much slower pace than what Moore has anticipated [45]. As an example, in 2013, the annual electricity consumption of data centers only in the United States was close to 91 billion KiloWatt Hour (KWH), which is larger than the annual amount of electrical power required by most countries [46]. Thereupon, service providers are taxed by intimidating energy bills as they try to provide adequate Quality-of-Service (*QoS*) guarantees. Such a consumption rate of electricity is not only a cost-and-profit problem but also a severe threat to the environment as a result of the massive amounts of carbon dioxide emissions during powering and cooling those data centers [46]. As a result, proficient power-aware resource management policies become a necessity and a critical infrastructure component for any agile, consolidated and dynamically scalable cloud's data center that provides affordable and reliable high-quality services.

On the other hand, power-saving techniques tend to, if not always, cause a degraded computing performance. The *QoS* is the key to clients' satisfaction, and service providers normally provide multiple (*SLAs*) regarding different *QoS* kinds. For example, a database may be queried internally by a company's employees or externally by a company's customers, who may have a higher or a lower priority than that of the employees [82]. Cloud service providers need to provide a competing and guaranteed *QoS*, but with fewer energy costs. Whereas over-provisioning is a common and simple solution to avoid *SLA* violations, resource over-provisioning is an expensive method by which resources are drastically underutilized, particularly under the unpredictably fluctuating cloud workloads [84].

Low resource utilization [50] is a prevailing problem in virtualized data centers, and is a major leading factor to their high power consumption and increased operational costs [45]. For example, while Google service provider makes its data centers greener by benefiting from wind and solar energy sources, and operating recycling cooling systems, the utilization of Google's servers is less than 50% on average [21]. Maximizing resource utilization becomes more crucial when performance must meet a defined satisfactory level of service given by *QoS* conditions. The challenge is then how to allocate the cloud's workloads in a way that maximizes resource usage and guarantees the *QoS* requirements. In this chapter, we propose and investigate new power-aware cloud workload allocation policies to minimize the processing power demand of cloud's services in cloud's data centers, and to reduce energy consumption, with statistically guaranteed *QoS* for users. We assume that clustered data centers can accommodate different types of services, each of which can have request classes with different *QoS* requirements. Our main contributions in this research are (1) Different from previous studies that employ separate Virtual Machines (VMs) for requests with different *QoS* requirements, we develop a workload multiplexing method that enables requests of the same service type, but with various *QoS* constraints, to share the same VM. To our best knowledge, this is the first approach by which different requests with different *QoS* guarantees are assigned to a single node to increase resource utilization. (2) We also devise a novel methodology that correctly discards potential failure requests as soon as possible to minimize processing rate demands, and to reduce total power consumption with statistically guaranteed *QoS*. We introduce a packing and consolidation algorithm, called *Green Workload Packing and Consolidation* *algorithm* (*GWPC*) that statistically ensures the *QoS* requirements of service requests in terms of deadline miss ratios. (3) In addition to the analytical validation of our proposed methods, we experimentally verify them, under general and cloud-specific workloads in one of our designed cloud platforms (e.g., Green Cloud Computing Prototype (*GCCP*)), described in Chapter 3. For example, we used the Data Caching Benchmark that emulates the behavior of a Twitter caching server and assumes the strict quality of service guarantees, such as 95% of the request must finish within 200 ms. Extensive experimentation results show that *GWPC* widely outperforms existing approaches in terms of *QoS* satisfaction, power consumption efficacy, resource demand minimization, and electricity cost saving. Extensive experimentation results show that *GWPC* outperforms current methods widely in terms of *QoS* satisfaction, power consumption efficacy, resource demand minimization, and electricity cost saving. We use the words VM, node, and server interchangeably throughout the chapter.

Numerous efforts have been made to reduce power and energy consumption in service-oriented computing systems. We can categorize those researches into different abstraction levels and/or according to different criteria. For example, according to the scale/type of the computing systems, Cai et al. in [50] categorized the energy-aware techniques applicable for servers [68,113], clusters [57,59,114], data centers [45,115] and the cloud [116]. Power/energy aware approaches can also be classified according to the different resource types, such as CPUs [45, 117], memory [84], storage devices [118], and/or network [30]. However, since CPUs usually acquire the highest power consumption among all resource types [115], we focus on improving power/energy efficiency of CPUs in cloud data centers using techniques such as virtualization, workload consolidation, and scheduling [119].

Dynamic Voltage Scaling and Dynamic Frequency Scaling (DVS/DFS) have been powerful conventional methods for adaptive performance and power dissipation adjustment to achieve power efficiency [57, 114]. Hwang et al. showed that the maximum energy savings in virtualized multi-core servers could be achieved when combining the *DVS*/*DFS* methods and the consolidation algorithms [68]. Beloglazov et al. introduced in [116] a global-and-local layer approach to make virtualized servers more power-efficient by adjusting the frequency and voltage of processors according to VMs' utilization. Likewise, Kim et al. [114] proposed *DVFS*-enabled, with both time-shared and space-shared, cluster scheduling policies for a bag of tasks to reduce power consumption and to meet end-users' deadline requirements. Although many researchers and engineers acknowledge that *DVFS* scheduling algorithms are powerful energy-saving solutions on the server's level, there are many challenges when they are applied in the current virtualized data centers; for instance, they are architecture dependent, hence they may not achieve their best power/energy-saving when used to the current heterogeneous cloud data centers.

As virtualization technology evolved as a norm in today's data centers to amplify resource usage through running multiple VMs on a single server, VM migration has been widely employed to optimize server utilization and to reduce power consumption [29]. In [120], Mastroianni et al. statistically modeled and analyzed the effects of VMs allocation and migration on minimizing the number of powered-on servers, and on reducing power consumption in data centers. Zhen et al. [29] introduced the concept of skewness to measure the unevenness in the servers' multidimensional-resource utilization. However, VM live migration requires a delay that can degrade the overall system performance and availability and consequently leads to *SLA* violations [121].

In conjunction with VM migration, server consolidation is of particular interest among efficient resource allocation policies [85]. Server consolidation, comparatively to *DVFS*, improves resource utilization without demanding excessive hardware resources, and it is easy to implement and to deploy [120]. Now that, server power consumption is not exactly proportional to its utilization, and a server may waste a non-trivial amount of power even if it is shut down [120], server consolidation methods pack running VMs on a smaller number of physical servers and/or turn off the rest, to minimize the total power consumed by those servers [58, 113, 120]. In [113], Verma et al. presented a two-dimensional, i.e., memory-based and CPU-based, consolidation strategy in which decisions are based on the correlation among different workloads. In [113], Pinheiro presented an algorithm to dynamically turn servers on and off according to the imposed load in computing clusters. Chase et al. [122] reduced the energy consumption of server clusters by degrading services according to their *SLAs*, when power consumption or thermal dissipation exceeds certain limits.

Whereas saving power/energy is important, service providers must also ensure that their services can satisfy users' *QoS* requirements, such as response time and/or deadline miss ratios. For instance, a recent report has found that a 100ms extra delay costs Amazon 1% of sales revenue [123]. The problem becomes more challenging with interactive workload types [61], such as online gaming [21] and multimedia streaming services [62], whose response times are crucial. These online interactive services are defined by soft-timely constraints [61], in that service providers must guarantee that a predefined percentage of them meet their deadlines. Otherwise, service providers fail to keep up with their *SLAs* [20].

VM placement methods with performance-interference awareness were introduced in [124] to improve the performance of VMs and the utilization of physical machines. Resource overbooking, i.e., allocating more resources than the actual available capacity to raise service provider profit, with different real-time constraints is presented in [61]. Energy-aware resource allocations with response time and end-to-end time guarantees were introduced in [58] and [59], respectively. Greenberg et al. [30] studied the costs of cloud data centers and recommended to developing new management systems within and across geographically distributed data centers with the focus on network agility to improve their efficiency and end-to-end performance.

SLA-aware workload consolidation had been proposed to achieve higher dynamic power efficiency and to overcome the under-utilization problems resulting from applying the over-provisioning policies [86]. Lee et al. [125] developed a pricing model, based on the queue model M/M/1/PS, and used it to produce a profit-driven scheduling algorithm with *SLA* for the cloud's dependent services.

To guarantee service requests with different classes of *QoS* requirements, it has been a common approach (e.g., [59]) to serve requests with the same *QoS* requirements on the same VM. Now that, all requests on the same machine have the same *QoS* requirements, different types of *QoS* can be captured by a single variable, such as provisioned resources (e.g., [113] [120] [116]), required processing speed (e.g., [57]) or latency (e.g., [58] [59]). Although this approach simplifies the resource management problem to guarantee one specified *QoS* criteria, it excludes requests that can share resources, and the overall resource usage can be rather inefficient, as illustrated later in this research. Additionally, almost all previous works implicitly assumed that all accepted requests must be served, even if they do not meet their *QoS* conditions. We show in this chapter that if we can judiciously discard the requests that are likely to miss their deadlines, we can significantly improve resource usages without compromising *QoS* conditions.

4.2 System model

In this section, we describe our proposed system model and formulate the problem.

4.2.1 Service model

We assume that a cloud data center consists of several cloud computer clusters, each of which consists of two or more physical machines, and has its own service manager that is analogous to the cluster schedulers in Google's clustered data centers [82]. Each cloud's cluster has a cloud orchestrator (such as, OpenStack) and a virtualization hypervisor (such as, Xen [145]) that work together to create VMs with different types and capacities for hosted services, and to make them available online for customers who submitted requests with different *QoS* requirements. We assume that a service provider provides n different types of services based on their application purpose $S = \{S_1, ..., S_n\}$. Each type of service (e.g., S_i) can accommodate different classes of service requests $\Gamma_i = \{\tau_{i,j}, j = 1, \ldots, r_i\}$, i.e., requests under different *SLAs*. Each class of requests (e.g., $\tau_{i,j}$) has its own *QoS* requirements (e.g., $Q_{i,j}$). We assume that different types of services must be hosted on different VMs, but different classes of requests of the same type can be potentially hosted in the same VM. We assume that there are n types of VMs $\{VM_1, ..., VM_n\}$ with capacities of $\{C_1, ..., C_n\}$ supporting n different types of services $\{S_1, ..., S_n\}$, respectively. VMs with the same service type S_i are logically grouped together into a single server pool SP_i. A server pool SP_i may contain up to m_i VMs. Each VM $\{VM_{i,k}, k = 1, \ldots, m_i\}$ within a server pool SP_i can require a different processing rate $\{U_{i,k}, k = 1, \ldots, m_i\}$. Our service model is illustrated in Fig. 4.1. We can see that different request classes of the same service type can share the same VM (e.g., τ_{11} and τ_{14} share VM_{12}, and τ_{12} and τ_{13} share VM_{11}). Contrarily, $\{\tau_{21}, \tau_{2m}\}$ are hosted separately on $\{VM_{21}, ..., VM_{2m}\}$, respectively.

As long as both waiting and average response times distributions of industrial workloads' requests have variances with small coefficients, we assume that request arrival patterns follow the Poisson distribution, and their response times follow the exponential distribution [160], as they approximate the actual corresponding distributions with acceptable precision [161]. Different request classes of the same service type may have different arrival times, deadlines, and completion ratios. Specifically, a request is modeled with a 3-tuple, i.e., $\tau_{ij} = \{\lambda_{i,j}, D_{i,j}, R_{i,j}\}$; where $\lambda_{i,j}$ is the arrival rate of j-class requests in service S_i, $D_{i,j}$ is the deadline of j-class requests in S_i, and $R_{i,j}$ is the required completion ratio of j-class requests in S_i. The QoS requirement $Q_{i,j}$ of a request $\tau_{i,j}$ is defined by $\{D_{i,j}, R_{i,j}\}$, meaning that at least $R_{i,j}$ percent of $\tau_{i,j}$ requests have to be served no later than $D_{i,j}$ as in [69]; for example, CloudSuite [162], a benchmark suite for cloud services, describes the QoS constraints of web search requests by a latency of $D = 500$ ms and completion ratio of $R = 90\%$ [163].

4.2.2 Power model

Considering that the allocation of processing units in cloud data centers generally occurs at levels of whole core(s) [69, 164], we assume that each VM is allocated to an individual processing core on a physical server, and, thus, we adopt a power model similar to that in [120] to model the power consumption of a VM, as shown in (4.1):

$$P = P^d \varphi + P^s \tag{4.1}$$

where; P^s is the static power, P^d is the dynamic power, and φ is the utilization of a processing core. φ is defined as $\varphi = \frac{U}{C}$; where U is the processing rate for a VM, and C is the capacity limit of the core allocated to it (i.e., the maximum processing rate available). We calculate P^d and P^s empirically, as explained later on in section 3.2.2, and we assume that they are constant and the same for each set of m_i cores hosting the same service type S_i. This model can be easily extended to the scenarios wherein a VM is mapped to multiple cores.

4.2.3 Problem definition

With the system model defined above, the problem we are to address can be formulated as follows.

Problem 4.1. *Given service requests* $\Gamma = \{\tau_{i,j} : j = 1, ..., r_i;\ i = 1, \ldots, n\}$*, determine the server pools* $SP = \{SP_i : i = 1, \ldots, n\}$*; where* $SP_i = \{VM_{i,k} : k = 1, ..., m_i\}$*, the corresponding processing rate* $\{U_{i,j} : i = 1, ..., n; j = 1, ..., m_i\}$ *for each VM (e.g.,* $VM_{i,j}$*), and the allocation of* Γ *to the VMs within each server pool in SP, such that the* QoS *requirements of the requests* $\{Q_{i,j}, i = 1, ..., n; j = 1, ..., r_i\}$ *are guaranteed, and the power consumption of each server pool is minimized.*

This problem involves two intertwined problems: (a) how to judiciously pack service requests to a VM, (b) and how to determine the proper service rate to minimize the power consumption while guaranteeing the *QoS* for all classes of request types. Next, we discuss our analytical results for this problem, and then present our algorithm in details.

4.3 Preliminaries

This section presents several key analysis results with regard to *QoS* guarantees, requests multiplexing, and requests packing. These results form the basis of our approach.

4.3.1 Processing rate minimization for QoS guarantee using request reneging

Traditionally, M/M/1 queue [160] has commonly been adopted to represent the request processing procedure [20], as shown in Fig. 4.2a. Service requests arrive with a rate λ, wait in a queue with an infinite size, and are processed with a rate μ. Accordingly, the

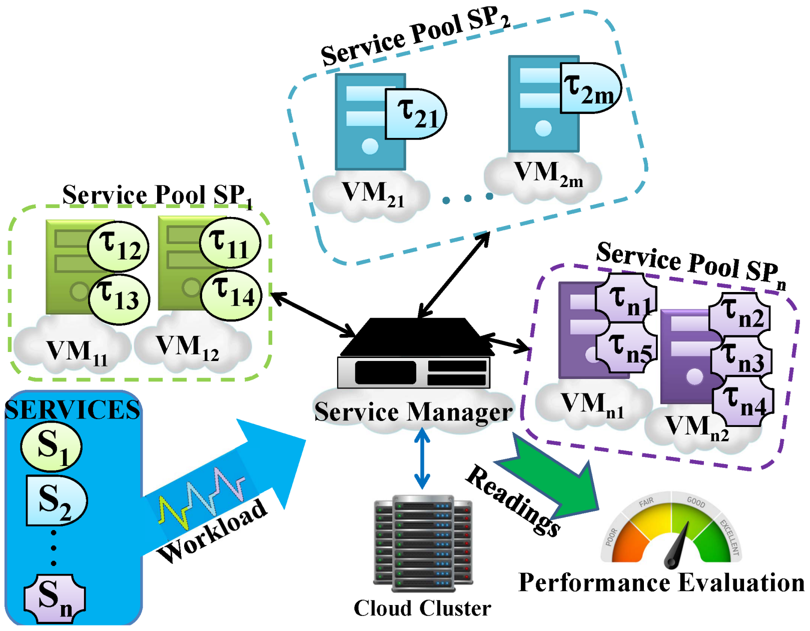


Figure 4.1: Service model

Probability Density Function (*PDF*), and the Cumulative Distribution Function (*CDF*) of the response time can be formulated as:

$$f(t) = (\mu - \lambda)e^{-(\mu-\lambda)t} \tag{4.2}$$

$$F(t) = 1 - e^{-t\mu(1-\frac{\lambda}{\mu})} \tag{4.3}$$

with a mean response time:

$$E[t] = \frac{1}{\mu - \lambda} \tag{4.4}$$

The q-percentile of the response time t_q (i.e., t_q is larger than $q\%$ of all response times) has the following relationship:

$$1 - e^{-t_q\mu(1-\frac{\lambda}{\mu})} = \frac{q}{100} \tag{4.5}$$

To this end, given a request $\tau_{i,j}$'s arrival rate $\lambda_{i,j}$, deadline $D_{i,j}$, and completion ratio requirements $R_{i,j}$. In order to guarantee $Q_{i,j}$, the required service rate $\mu_{i,j}$ (when $\tau_{i,j}$ is hosted alone) is:

$$\mu_{i,j} = \frac{ln[\frac{1}{1-R_{i,j}}]}{D_{i,j}} + \lambda_{i,j} \tag{4.6}$$

The $\mu_{i,j}$ defined above can guarantee $Q_{i,j}$, i.e., no more than $(1-R_{i,j})\%$ of the requests can miss their deadlines.

It is rational to drop a request if it is a potential failure in terms of missing its deadline so that we can save the precious resource for requests that are more likely to successfully complete in time. The problem nevertheless is how to discard these requests without compromising the *QoS*. To this end, we employ the $M/M/1$ queue with the reneging model [165], as illustrated in Fig. 4.2b.

As shown in Fig. 4.2b, according to the reneging model, each request is associated with a deadline. If a request is not fully served by its deadline, it is removed from the system. According to this model, there exists a provocative relationship among the request's deadline miss probability P_{miss}, arrival rate λ, processing rate μ, and deadline D, which can be formulated as [165]:

$$P_{miss} = \frac{(1-\rho)e^{\mu D(\rho-1)}}{1-\rho e^{\mu D(\rho-1)}} \tag{4.7}$$

where $\rho = \frac{\lambda}{\mu}$. Accordingly, for a given $\lambda_{i,j}$, $R_{i,j}$, and $D_{i,j}$ of request $\tau_{i,j}$, we can derive $\mu^*_{i,j}$ that guarantees $Q_{i,j}$:

$$1-R_{i,j} \leq \frac{(1-\frac{\lambda_{i,j}}{\mu^*_{i,j}})e^{\mu^*_{i,j}D_{i,j}(\frac{\lambda_{i,j}}{\mu^*_{i,j}}-1)}}{1-\frac{\lambda_{i,j}}{\mu^*_{i,j}}e^{\mu^*_{i,j}D_{i,j}(\frac{\lambda_{i,j}}{\mu^*_{i,j}}-1)}} \tag{4.8}$$

By judiciously removing the requests from the queue, we can guarantee the same *QoS* with lower processing rates. This conclusion is formally formulated in Theorem 4.1.

Theorem 4.1. *A service request $\tau = \{\lambda, D, R\}$; where λ, D, and R refer to its arrival rate, deadline, and completion ratio requirement, respectively. Let μ^* and μ be the processing rates to satisfy R based on the M/M/1 queue model with and without request reneging, respectively. Then $\mu \geq \mu^*$.*

Proof. From (4.6), we have

$$\mu = \frac{ln[\frac{1}{1-R}]}{D} + \lambda \tag{4.9}$$

If we apply the same processing rate for requests of the same service type with reneging, and let the result completion ratio be R^*, then based on Equation 4.8 we have

$$\begin{aligned} R^* &= 1 - P_{miss} \\ &= \frac{1 - e^{(\lambda D - \mu D)}}{1 - \frac{\lambda}{\mu} e^{(\lambda D - \mu D)}} \end{aligned} \tag{4.10}$$

Then with (4.9), we have

$$\begin{aligned} R^* &= \frac{1 - e^{[\lambda D - (\frac{ln[\frac{1}{1-R}]}{D} + \lambda) D]}}{1 - \frac{\lambda}{\mu} e^{[\lambda D - (\frac{ln[\frac{1}{1-R}]}{D} + \lambda) D]}} \\ &= \frac{R}{1 - \frac{\lambda}{\frac{ln[\frac{1}{1-R}]}{D} + \lambda}(1-R)} \end{aligned} \tag{4.11}$$

Since $(1 - \frac{\lambda}{\frac{ln[\frac{1}{1-R}]}{D} + \lambda}(1-R)) < 1$, we have:

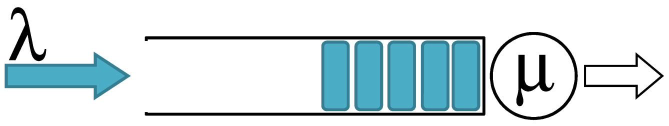
$$R^* = \frac{R}{1 - \frac{\lambda}{\frac{ln[\frac{1}{1-R}]}{D} + \lambda}(1-R)} > R \tag{4.12}$$

Equation 4.12 indicates that the processing rate μ based on the M/M/1 queue can lead to a larger completion ratio R^*, if request reneging is allowed. Therefore, to obtain the same completion ratio (e.g., R), we have $\mu^* \leq \mu$. □

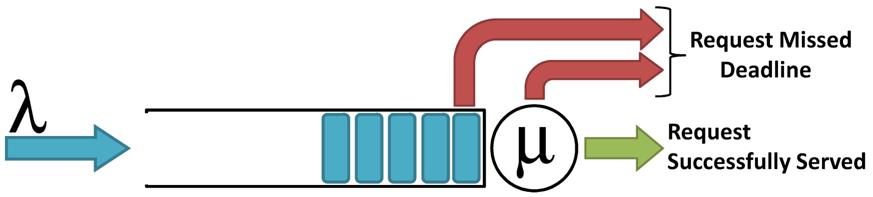
4.3.2 Request multiplexing

When a service S_i has multiple request classes $\{\tau_{i,j}, j = 1, ..., r_i\}$, a common approach is to host each request class on a single VM $\{VM_{i,j}, j = 1, ..., r_i\}$, respectively. The $R_{i,j}$-th percentile response time $t_{R_{i,j}}$ of $\tau_{i,j}$ can be formulated as:

$$t_{R_{i,j}} = \frac{1}{\mu_{i,j} - \lambda_{i,j}} ln[\frac{1}{1 - R_{i,j}}]. \tag{4.13}$$



(a) M/M/1 queue model



(b) M/M/1 model with request reneging

Figure 4.2: Processing models

When hosting each request class $\tau_{i,j}$ individually on $VM_{i,j}$, let the server pool SP_i $= \{VM_{i,j}, ..., VM_{i,r_i}\}$ has the processing rates $\{U_{i,1} = \mu_{i,1}, ..., U_{i,r_i} = \mu_{i,r_i}\}$, respectively. Then to satisfy each $Q_{i,j}$, the processing rates can be calculated according to Equation 4.6:

$$\mu_{i,j} = \frac{ln[\frac{1}{1-R_{i,j}}]}{D_{i,j}} + \lambda_{i,j} \tag{4.14}$$

A better approach, withal, is to host multiple request classes in a single VM with a processing rate U that satisfies the QoS requirements of all hosted classes. We formulated this essential finding in Theorem 4.2.

Theorem 4.2. *For the i-type service requests* $\{\tau_{i,j}, j = 1, ..., r_i\}$ *hosted in a single node* $\hat{VM}$*, let the processing rate of* $\hat{VM}$ *be* $\hat{U}$ *and let*

$$U_{i,j} = \mu_{i,j} + \sum_{q=1, q\neq j}^{r_i} \lambda_{i,q}. \tag{4.15}$$

Then the QoS *requirements for* $\{\tau_{i,j}, j = 1, \dots, r_i\}$ *can be satisfied if* $\hat{U} \geq \max_{j=1}^{r_i} U_{i,j}$.

Proof. When all classes of requests are hosted together in a single node (e.g., $\hat{VM} = VM_{i,1}$), the processing rate of $VM_{i,1}$, i.e., $U_{i,1}$, has to satisfy the *QoS*

$$U_{i,1} = \mu_{i,1} - \lambda_{i,1} + \sum_{j=1}^{r_i} \lambda_{i,j} \tag{4.16}$$

$$= \mu_{i,1} + \sum_{j=2, j\neq 1}^{r_i} \lambda_{i,j}. \tag{4.17}$$

We can equivalently derive the required processing rates for any $VM_{i,l}$, i.e., $U_{i,l}$; where $i \in [1 - r_i]$, according to the *QoS* requirements of $\tau_{i,l}$ as follows:

$$U_{i,l} = \mu_{i,l} + \sum_{j=1, j\neq l}^{r_i} \lambda_{i,j}, \tag{4.18}$$

Therefore, when $\{\tau_{i,1}, ..., \tau_{i,r_i}\}$ are hosted together in $\hat{VM} = VM_{i,l}$, in order to satisfy the *QoS* requirements for all classes of requests

$$\hat{U} = \max\{U_{i,1}, ..., U_{i,r_i}\}. \tag{4.19}$$

$\square$

From Theorem 4.2, when multiple classes of service requests are multiplexed in a single VM, the processing rate can be easily identified to ensure the *QoS* conditions for these requests. In the meantime, we show that request multiplexing helps improve resource utilization, as implied in the Theorem 4.3. Let us first define the processing rate $\Omega(SP_i)$ of a server pool SP_i.

Definition 4.1. The processing rate of a server pool, *denoted as* $\Omega(SP_i)$*; where* $SP_i = \{VM_{i,1}, ..., VM_{i,r_i}\}$*, is the sum of all VMs' required processing rates* $\{U_{i,1}, ..., U_{r_i}\}$ *in that server pool:*

$$\Omega(SP_i) = \sum_{j=1}^{r_i} \mu_{i,j} = \sum_{j=1}^{r_i} [\frac{ln[\frac{1}{1-R_{i,j}}]}{D_{i,j}} + \lambda_{i,j}] \tag{4.20}$$

Theorem 4.3. *Given the i-type service requests $\{\tau_{i,1}, \ldots, \tau_{i,r_i}\}$, let $SP_i = \{VM_{i,1}, \ldots, VM_{i,r_i}\}$ be all VMs, when each class of requests, $\tau_{i,j}$, is served separately with a dedicated $VM_{i,j}$, and let $\hat{SP}_i = \{\hat{VM}_{i,1}\}$ be a server pool with a single VM that serves all the requests simultaneously. Let $\Omega(SP_i)$ ($\Omega(\hat{SP}_i)$) be the processing rate of the server pool SP_i ($\hat{SP}_i$, resp.) such that the* QoS *requirements for all $\{\tau_{i,j}, j = 1, ..., r_i\}$ are satisfied. Then $\Omega(SP_i) \geq \Omega(\hat{SP}_i)$.*

Proof. When each class of requests is served separately with a dedicated VM, we have from Definition 4.1:

$$\Omega(SP_i) = \sum_{j=1}^{r_i} \mu_{i,j} = \sum_{j=1}^{r_i} [\frac{ln[\frac{1}{1-R_{i,j}}]}{D_{i,j}} + \lambda_{i,j}]. \tag{4.21}$$

When all requests are served by a single VM, we have from Theorem 4.2:

$$\Omega(\hat{SP}_i) = \max_{j=1}^{r_i} \hat{U}_{i,j} \tag{4.22}$$

where $\hat{U}_{i,j} = \mu_{i,j} + \sum_{q=1, q\neq j}^{r_i} \lambda_{i,q}$ is the required processing rate of $VM_{i,j}$ to meet $Q_{i,j}$. Because $\mu_{i,j} \geq \lambda_{i,j}$ for all $j \in [1, r_i]$, we have

$$\frac{\Omega(SP_i)}{\Omega(\hat{SP}_i)} \geq 1, \tag{4.23}$$

or, equivalently, $\Omega(SP_i) \geq \Omega(\hat{SP}_i)$. □

Theorem 4.3 indicates that multiplexing different request classes on a single $VM_{i,j}$ can result in a smaller processing rate for the server pool, provided that the processing rate is feasible on $VM_{i,j}$; i.e., required processing rate must not exceed the VM's maximum capacity $C_{i,j}$. If only one VM cannot accommodate all service requests without compromising their QoS requirements, we will need more than one VM. How can we allocate service requests to VMs while optimizing the processing rate utilization of the hosting servers utilization? We address this question next.

Table 4.1: Processing rate comparison with different requests packing strategies

Packing Strategy 1		**Packing Strategy 2**	
VM₁	***VM₂***	***VM'₁***	***VM'₂***
$\tau_i = \{\tau_1, \tau_2\}$	$\tau_i = \{\tau_3, \tau_4\}$	$\tau_i = \{\tau_1, \tau_4\}$	$\tau_i = \{\tau_2, \tau_3\}$
$U_1 = 160$	$U_2 = 140$	$U_1' = 150$	$U_2' = 130$

4.3.3 Request packing

From the discussions above, clustering multiple classes of requests into the same VM helps improve the resource usage. When more than one VM is needed, the question then becomes how to group different classes of requests into each VM to minimize the server pool processing rate $\Omega = \sum_i \sum_k U_{i,k}$, and to maximize the overall resource usage efficiency. Consider the following example with four request classes of the same type $\{\tau_1, \tau_2, \tau_3, \tau_4\}$, with $\lambda = \{60, 40, 50, 20\}$ request' Instances Per Second (*IPS*), and $\mu = \{120, 80, 70, 90\}$ *IPS*, respectively. μ_i is the minimum processing rate of request τ_i to satisfy its *QoS*, when it is allocated individually to a VM. To guarantee all the *QoS* requirements, two request-grouping strategies are shown in Table 4.1. The server pool's processing rates using both strategies are $\Omega(V_1, V_2) = 300$ and $\Omega(V_1', V_2') = 280$, respectively (derived based on Theorem 4.2). This example clearly shows that different requests' allocation strategies lead to server pools with different processing rates and, thus, utilizations.

We show in Theorem 4.6 that the general packing problem is NP-hard in nature. Subsequently, we focus on the development of an effective and efficient heuristic solution for this problem. Specifically, we have made several crucial observations, based on a service allocation onto two servers, which we formulate in the following theorems.

Theorem 4.4. *Let* $\Gamma_1 = \Gamma_{1,p} \bigcup \Gamma_{1,q}$*; where* $\Gamma_{1,p} = \{\tau_{1,p_1}, \tau_{1,p_2}, ..., \tau_{1,p_s}\}$, $\Gamma_{1,q} = \{\tau_{1,q_1}, \tau_{1,q_2}, ..., \tau_{1,q_s}\}$*, and* $\Gamma_{1,p} \bigcap \Gamma_{1,q} = \emptyset$*. Assume* $\Gamma_{1,p}$ *and* $\Gamma_{1,q}$ *are mapped to two VMs with the same capacity,* $VM_{1,p}$ *and* $VM_{1,q}$*, respectively. For each* $\tau_{i,j}$*, let*

$$\phi_{i,j} = \mu_{i,j} - \lambda_{i,j}. \tag{4.24}$$

Let $U_{1,p}$ ($U_{1,q}$, resp.) denote the minimum processing rate for $VM_{1,p}$ ($VM_{1,q}$, resp.) that guarantees the QoS *requirements of $\Gamma_{1,p}$ ($\Gamma_{1,q}$, resp.). Then the processing rate for the server pool $SP = \{VM_{1,p}, VM_{1,q}\}$, i.e., $\Omega(SP) = U_{1,p} + U_{1,q}$, is minimized if the quantity given in Equation 4.25 is minimized:*

$$\Phi = \max_{\tau_{1,p}\in\Gamma_{1,p}} \phi_{1,p} + \max_{\tau_{1,q}\in\Gamma_{1,q}} \phi_{1,q}. \tag{4.25}$$

Proof. From Theorem 4.2, we know that

$$\begin{aligned} u_{1,p} &\geq \max_{\tau_{1,p}\in\Gamma_{1,p}} \{\mu_{1,p} + \sum_{j=p_1, j\neq p}^{p_s} \lambda_{1,j}\} && (4.26)\\ &= \max_{\tau_{1,p}\in\Gamma_{1,p}} \phi_{1,p} + \sum_{j=p_1}^{p_s} \lambda_{1,j} && (4.27)\end{aligned}$$

comparatively,

$$\begin{aligned} u_{1,q} &\geq \max_{\tau_{1,q}\in\Gamma_{1,q}} \{\mu_{1,q} + \sum_{j=q_1, j\neq q}^{q_s} \lambda_{1,j}\} && (4.28)\\ &= \max_{\tau_{1,q}\in\Gamma_{1,q}} \phi_{1,q} + \sum_{j=q_1}^{q_s} \lambda_{1,j} && (4.29)\end{aligned}$$

Thusly, $\Omega(V) = U_{1,p} + U_{1,q}$ is minimized if $\Phi = \max_{\tau_{1,p}\in\Gamma_{1,p}} \phi_{1,p} + \max_{\tau_{1,q}\in\Gamma_{1,q}} \phi_{1,q}$ is minimized. □

Theorem 4.4 means that to minimize the processing rate for a server pool SP_i, we need to minimize the sum of maximum $\phi_{i,j}$ (defined in Equation 4.24) for the services allocated to each node. A simple heuristic is, hence, to sort all requests classes $\{\tau_{i,j}, j = 1, ..., r_i\}$ according to their $\{\phi_{i,j}, j = 1, ..., r_i\}$, and allocate as many high-ranking classes as possible on the same VM. Specifically, we have the following theorem.

Theorem 4.5. *Let $\Gamma_i = \{\tau_{i,1}, \tau_{i,2}, ..., \tau_{i,r_i}\}$ be mapped to two VMs with the same capacity, $VM_{i,p}$ and $VM_{i,q}$. Let $\tau_{i,k} \in \Gamma_i$ and let*

$$\Gamma^h_{i,k} = \{\tau_{i,j} \in \Gamma_i | \phi_{i,j} > \phi_{i,k}\}. \tag{4.30}$$

Then the processing rate for the server pool $SP_i = \{VM_{i,p}, VM_{i,q}\}$, i.e., $\Omega(SP_i) = U_{i,p} + U_{i,q}$, is minimized if $\tau_{i,k}$ is the one with the smallest $\phi_{i,k}$ *such:*

- $\Gamma^h_{i,k}$ *are feasibly allocated to one server (e.g., $VM_{i,p}$);*
- $\Gamma^h_{i,k} + \{\tau_{i,k}\}$ *cannot be feasibly allocated to the same server ($VM_{i,p}$) simultaneously;*
- $\tau_{i,k}$ *is feasibly allocated to another server (e.g., $VM_{i,q}$).*

Proof. From Theorem 4.4, to minimize $\Omega(SP_i)$ we only need to minimize $\Phi = \max_{\tau_{i,p}} \in \Gamma_{i,p}\ \phi_{i,p} + \max_{\tau_{i,q}} \in \Gamma_{i,q} \phi_{i,q}$. Without loss of generality, assume that $\tau_{i,\alpha} \in \Gamma_i$ is the one with the largest value of ϕ, and is allocated to $VM_{i,p}$. Then to optimize $\Omega(SP_i)$ we only need to optimize $\max_{\tau_{i,q} \in \Gamma_{i,q}} \phi_{i,q}$. Note that any $\tau_{i,j} \in \Gamma^h_{i,k}$ allocated to $VM_{i,q}$ will lead to a larger $\max_{\tau_{i,q} \in \Gamma_{i,q}} \phi_{i,q}$ than it is when all $\Gamma^h_{i,k}$ are allocated to $VM_{i,p}$. But, $\Gamma^h_{i,k} + \{\tau_{i,k}\}$ cannot be feasibly allocated to the same node $VM_{i,p}$ simultaneously, and, thus, $\tau_{i,k}$ has to be allocated to $VM_{i,q}$. For the rest of $\tau_{i,j}$, their allocations do not affect the optimality of $\Omega(SP_i)$ as shown in Theorem 4.4. □

Note that Theorem 4.5 helps to identify the optimal service packing solution for two VMs. However, if there are more than two VMs, finding the optimal solution becomes substantially more complicated due to the trade-off between minimizing the maximum value of ϕ for a VM, and the total number of needed VMs. In Theorem 4.6, we show that the service packing problem involving more than two VMs is NP-hard.

Theorem 4.6. *Let $SP_i = \{VM_{i,1}, VM_{i,2}, \ldots, VM_{i,m_i}\}$ be a server pool with $m \geq 3$, hosts a set of requests $\Gamma_i = \{\tau_{i,1}, \tau_{i,2}, ..., \tau_{i,r_i}\}$ from the same service type S_i, but with different* QoS *constraints. Assume that all VMs in SP_i have the same capacity C. Then packing the request set Γ_i in the server pool SP_i such that the server pool processing rate $\Omega_i = \sum_{j=1}^{m_i} U_{i,j}$ is minimized is NP-hard.*

4.4 The Green Workload Packing and Consolidation (GWPC) algorithm with statistical guarantee

We are now ready to discuss our approach for power consumption minimization in cloud data centers with guaranteed *QoS*. Inasmuch as the overall power consumption of a server pool depends on both the processing rates of its VMs, and the static power consumption of the physical servers hosting those VMs (see Equation 4.1), to solve Problem 4.1, we need to minimize the number of VMs, and their processing rates. Thus, we develop an algorithm, called *Green Workload Packing and Consolidation algorithm (*GWPC*)* to allocate VMs and map service requests onto them.

First, *GWPC* intends to consolidate multiple request classes to the same VM. As shown in Theorem 4.2, when various classes are hosted in the same VM, the total processing rate of a server pool reduces, which helps to minimize the dynamic power consumption of the physical cores on which VMs run. Moreover, consolidating multiple request classes in a single VM reduces the total number of needed VMs, which in turn minimizes the total number of hosting physical machines and their static power. Second, *GWPC* adopts the reneging model to judiciously expunge service requests. Specifically, in Fig. 4.3, the required processing rate (e.g., μ_j) for each class of requests (e.g., τ_j) is calculated based on the reneging model (line 1). We then sort all requests based on $\phi_j = \mu_j - \lambda_j$ in a decreasing order (line 2), and pack the requests from the list to VMs with the capacity (i.e., the maximum processing rate) of C (line 3 to 13). Theorem 4.6 clearly demonstrates that Problem 4.1 is NP-hard, so we resort to the traditional first-fit bin-packing algorithm to pack the requests, and minimize the number of VMs. This helps reduce the static power consumption of the server pool. Sorting the requests according to the value of ϕ is a good heuristic with a basis presented in Theorem 4.5. It helps minimize the processing rate of

```
Input: A set of request classes $\Gamma = \{\tau_j, j = 1, \ldots, r\}$ of a service
    $S$, each $\tau_j$ has corresponding $\lambda_j$ and $Q_j$ ($\{D_j, R_j\}$). A server
    pool $SP$ of virtual servers $VM = \{VM_k, k = 1, \ldots, m\}$ with
    the same capacity $C$ serve the request classes from service $S$.
Output: The request classes allocation.
 1: Calculate each $\mu_j$ to satisfy $Q_j$ based on (8);
 2: $dif\_vect \leftarrow$ Sort the $\phi = \mu - \lambda$ in the decreasing order;
 3: for All requests $\tau_j$, $j = 1, 2, \ldots, r$ do
 4:     for All VMs $VM_k$, $k = 1, 2, \ldots, m$ do
 5:         Add $\tau_j$ into node $VM_k$;
 6:         Calculate $VM_k$'s required processing rate $U_k$ based on (2);
 7:         if ($U_k \leq C$) then
 8:             Remove $\tau_j$ from $dif\_vect$;
 9:             Break the loop and pack the next request;
10:         else
11:             Remove $\tau_j$ from the current node $VM_k$;
12:         end if
13:     end for
14:     if (Request $\tau_j$ did not fit in any available VM) then
15:         Allocate a new VM and pack request $\tau_j$;
16:     end if
17: end for
```

Figure 4.3: The Green Workload Packing and Consolidation (GWPC) algorithm

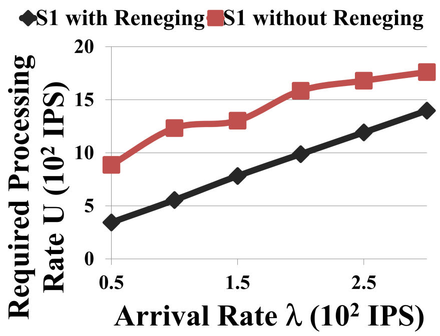
each VM, and, thus, it also help minimize the dynamic power consumption of the server pool.

In what follows, we present the experiments and results we obtained based using the *GCCP* cloud platform, that we described in the previous chapter.

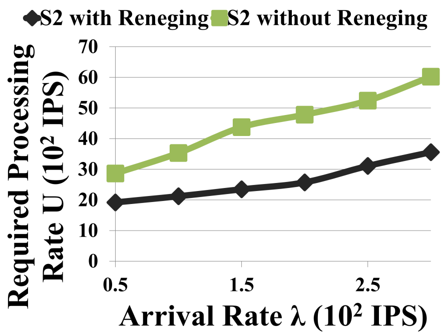
4.5 Experimental validation

In this section, we use experiments to validate our analytical findings and to test the performance of the *GWPC* algorithm using the *GCCP* platform, described in Chapter 3.

To investigate the performance of our approach, we implemented the following workload mapping and scheduling algorithms, which also employ failure reneging: (1) *Split* (denoted as (*SPT*)): the traditional method by which each request class is hosted in a separate VM [113], [166]; (2) *Random* (denoted as (*RND*)): the method that fills mul-

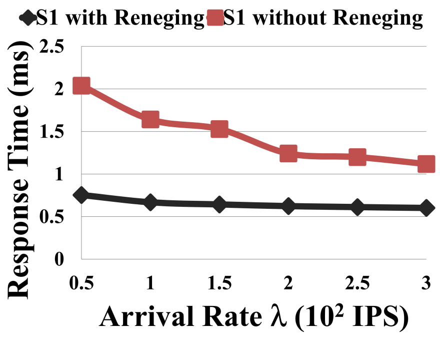


(a) $S_1 : 256 \times 256$ MMUL

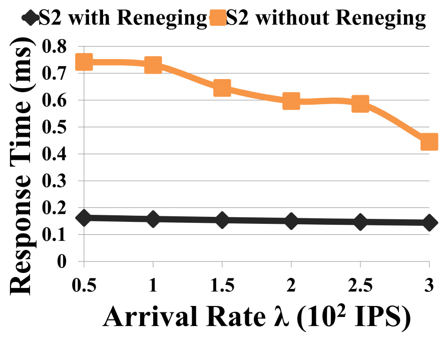


(b) $S_2 : 1 \times 256$ 1-D FFT

Figure 4.4: Minimum required processing rates for guaranteed QoS using M/M/1 queue model with and without reneging for (a) S_1 and (b) S_2 cloud service types



(a) $S_1 : 256 \times 256$ MMUL



(b) $S_2 : 1 \times 256$ 1-D FFT

Figure 4.5: Response time comparison for guaranteed QoS using M/M/1 queue model with and without reneging for (a) S_1 and (b) S_2 cloud service types

tiple request classes randomly into a VM; (3) *First-Fit-Decreasing* (denoted as *FFD*): the bin-packing method [167] with service classes combined based on the decreasing order of their required processing rates; (4) **Green Workload Packing and Consolidation** (denoted as (*GWPC*): our proposed method.

4.5.1 Performance with request reneging

We first empirically study the advantages of request reneging on minimizing the required processing rates and average response times with *QoS* guarantees. We also compare the performance of the system with and without request reneging using two different service types; i.e., memory-intensive (e.g., $S1$) and CPU-intensive (e.g., $S2$) types.

We generated two sets of classes with six different request classes each $\{\tau_{i,j} : i = 1, 2;\ j = 1, ..., 6\}$, with the first set from ($S_1 : 256 \times 256$ MMUL), and the second set from ($S_2 : 1 \times 256$ 1-D FFT). Request classes from both types were set to have average arrival rates of $\{\lambda_{i,1} = 50, \lambda_{i,2} = 100, \ldots, \lambda_{i,6} = 300;\ where\ i = 1, 2\}$, completion ratios of $R_{i,j} = 95\%$, and deadlines randomly generated following a uniform distribution in the ranges $[5-15]\times\ 10^2\mu s$ and $[2-6]\times\ 10^2\mu s$ for S_1 and S_2 classes, respectively. All parameters and their values were arbitrarily chosen.

We generated 10^5 Instances Per Request class (*IPR*), and request instances of each class were executed in a VM with and without reneging. The capacity of the VM was set to 1800 *IPS* and 6000 *IPS* for S_1 and S_2, respectively. In each run, we calibrated the maximum processing rate of each VM by assigning a cap to each VM's Virtual CPU (*VCPU*) that limits the maximum amount of processing rates a VM receives from its allocated physical core. We then applied the traditional binary search method to find the minimum required processing rates that can meet the given *QoS* conditions. We repeated the experiment for each setting 10^4 times, and the average results are shown in Fig. 4.4 and Fig. 4.16.

Fig. 4.4 compares the minimum required processing rates with and without reneging under different arrival rates and *QoS* settings. We see that for both S_1 and S_2 requests, the minimum required processing rates with reneging are much lower than those without reneging. For example, when the arrival rate is 200 *IPS*, the minimum required processing rates with reneging for S_1's and S_2's are 1000 *IPS* and 4800 *IPS*, respectively. If no requests have reneged, the minimum required processing rates become 1600 *IPS* and 5600 *IPS*, respectively. The increase in the processing rate thus is 60% and 16.7% over its counterparts. The minimum required processing rates unsurprisingly increase with the increment of arrival rates. Nonetheless, the minimum required processing rates with reneging increase at a much slower rate, as clearly shown in Fig. 4.4. In average, the minimum required processing rates with reneging for S_1 and S_2 are 62% and 58% lower than those without reneging, respectively. These results evidently conform to the theoretical conclusion formulated in Theorem 4.1 that request reneging helps reduce the processing rates while guaranteeing the same *QoS* requirements.

We can also observe that request reneging helps lower service response times. As shown in Fig. 4.5a and Fig. 4.5b, the average response times of S_1 and S_2 classes without reneging are always longer than those with reneging. On average, the average response times of S_1's and S_2's are 225% and 409% longer than those with reneging. As the arrival rates increase, the minimum required processing rates must increase to ensure the same *QoS* requirement. As a result, response times are reduced. While response times for requests without reneging change dramatically, as arrival rates increase, the response times for requests with reneging do not vary as significantly, which implies that requests with reneging can deliver service more stable in terms of response time variations. When comparing the response time improvement with reneging for the memory-intensive service S_1 and the CPU-intensive service S_2, we can see that S_1 benefits more than S_2 in term of the improvement for the minimum required processing rates and response times, as shown in Fig. 4.4 and Fig. 4.16. We conjecture that this is because S_1 requests require longer I/O-related operations, and cannot be easily terminated for request reneging, which negatively affects their response times when compared with the CPU-intensive requests.

Request reneging to a great extent not only reduces processing rate requirements to guarantee the same *QoS* requirements like the one without reneging, but also results in lower and more stable average response times, and is, therefore, a promising approach to achieve reliable and predictable performance.

4.5.2 Multiplexed vs. split request processing

Having validated the performance improvement of request reneging, we now compare the completion ratios and average response times when serving requests in a multiplexed manner (e.g., *GWPC*) and a split manner (e.g., *SPT*).

We generated two types of services, S_1 (128×128 MMUL) and S_2(1×64 1-D FFT). We randomly generated six testing groups of request classes from each service type, with the number of classes in each group varying from $r = 5$ to $r = 10$, i.e., $\{\tau_{i,j};\ i = 1,2;\ j = 1, \ldots, r\}$. The arrival rates and deadlines in each request class were randomly generated with the average following a uniform distribution in the ranges $[20 - 120]$ *IPS* and $[5 - 6]\ \times 10^2 \mu s$, respectively. We set 90% and 95% completion ratios for S_1 and S_2, respectively. In each test, we generated 10^5 *IPR* for each of S_1 and S_2 classes, separately applied the *GWPC* and *SPT* methods on those instances using VMs with a capacity set to 10^4 *IPS*, and calculated the achieved completion ratios and average response times in each run for each request class. All these parameters and their values were arbitrarily chosen. We repeated the experiment for each setting 10^4 times, and the average results are shown in Fig. 4.6 and Fig. 4.7.

Fig. 4.6 compares the completion ratios achieved by the multiplexed and split approaches under different experiment settings. As shown, both methods successfully guarantee the required completion ratios for S_1 and S_2. Nonetheless, *GWPC* can achieve a much higher average completion ratios than *SPT*. Note that, for the test cases when there are eight different request classes (i.e., $r = 8$), *GWPC* achieves a completion ratio of 94% for S_1 and 97.5% for S_2, in comparison with 90.2% for S_1 and 95.8% for S_2 achieved by *SPT*. Those results comply with the conclusion in Theorem 4.3 that request multiplexing helps reduce required processing rates without compromising *QoS* requirements. When all VMs have the same capacity, *GWPC* can unsurprisingly lead to better completion ratios in all test cases. *GWPC* can on average achieve an average completion ratios of

93.87% for for S_1 and 97.2% for S_2, whereas *SPT* can only achieve 90.18% and 95.92% for S_1 and S_2, respectively.

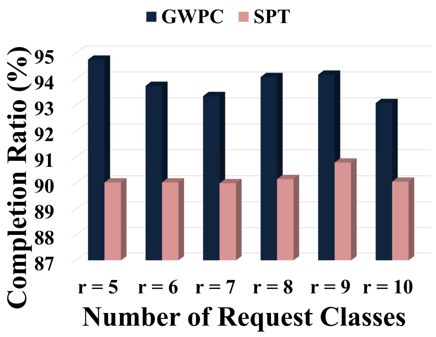
Request multiplexing also results in less average response times than those in *SPT*, as shown in Fig. 4.7. The average response times in the multiplexed approach outperform those in *SPT* in all test cases. The reason for such improvement is that *GWPC* can efficiently utilize computing resources among different request classes, as a result of request reneging and multiplexing. Moreover, allocating a smaller number of VMs reduces the overhead on the VM Manager (*VMM*). For example, the hypervisor, especially with memory-intensive workloads, which demand more privileged operations, such as memory accesses, context switches, system calls and interrupts [168]. For instance, in Fig. 4.7, when the number of classes is $r = 10$, *GWPC* results in an average response time that is 2.26μs less than those in *SPT* for S_1 classes, but for S_2 and with the same number of request classes $r = 10$, *GWPC* shows only 0.06μs better average response time than that of the *SPT*.

Overall, our experiments show that request multiplexing not only guarantees *QoS* requirements, but can also achieve higher completion ratios than required. It can better utilize computing resources than *SPT* does, especially with the memory-intensive service types.

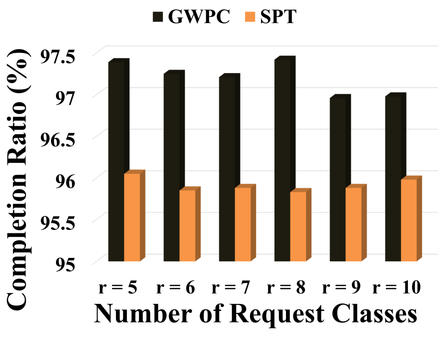
4.5.3 Performance under different service utilizations

In this section, we analyze the performance of *GWPC* in terms of power consumption and processing rate demand, compared with *SPT*, *FFD*, and *RND*.

We generated five groups of test cases, each of which has a different number of request classes $\{\tau_{i,j} : i = 1,2; j = 1,...,r_i\}$; where $r_i = 10,20,...,50$, from two service types, i.e., (S_1 : 128 × 128 MMUL) and (S_2 : 1 × 64 1-D FFT). The arrival rates and completion ratios

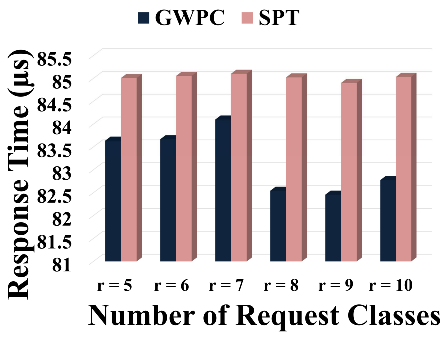


(a) S_1 : 128 × 128 MMUL

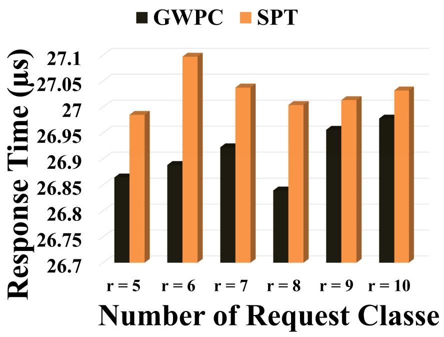


(b) S_2 : 1 × 64 1-D FFT

Figure 4.6: Performance of completion ratios between request multiplexing and splitting using requests of (a) S_1 and (b) S_2 cloud service types



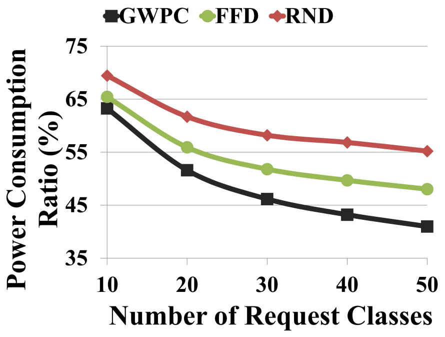
(a) S_1 : 128 × 128 MMUL



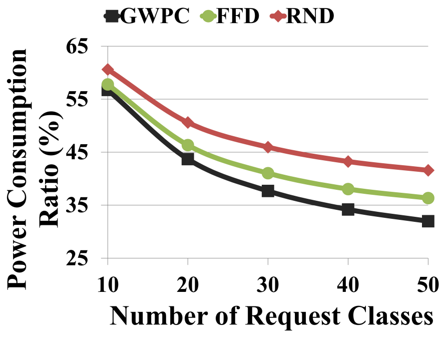
(b) S_2 : 1 × 64 1-D FFT

Figure 4.7: Performance of average response times between request multiplexing and splitting using requests of (a) S_1 and (b) S_2 cloud service types

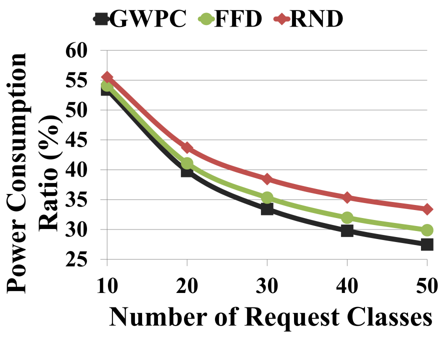
were randomly generated with the average following uniform distributions in the ranges $[20 - 500]$ *IPS* and $[90\% - 95\%]$, respectively. We recall that, from Equation (4.8), when the arrival rate is a constant value, the smaller the deadline, the higher the required processing rate is. Thence, as the deadline reduces, the processing rate $\mu_{i,j}$ increases, and then the service utilization increases. We accordingly varied request utilization by changing the intervals from which we randomly picked the deadlines. For each set of request classes, we varied the deadline range among four intervals, starting from 500 μs and 600 μs with interval length of 50μs and 100μs for S_1 and S_2, respectively.



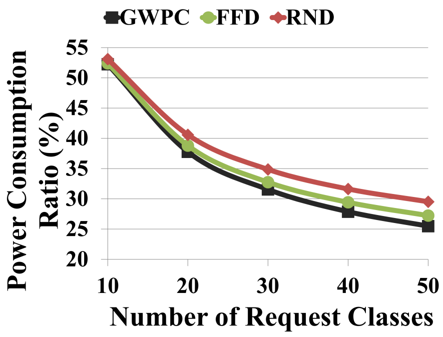
(a) $D_{1,j} \in [500-550]\mu s$



(b) $D_{1,j} \in [550-600]\mu s$



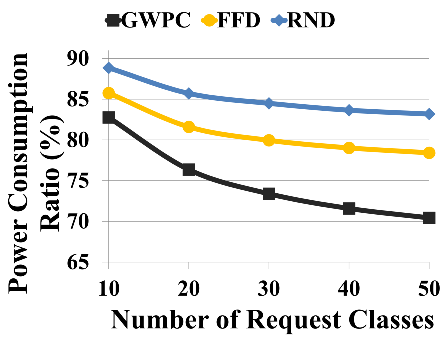
(c) $D_{1,j} \in [600-650]\mu s$



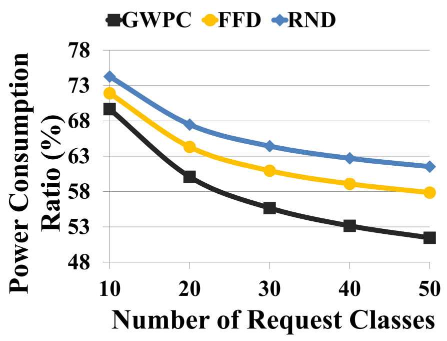
(d) $D_{1,j} \in [650-700]\mu s$

Figure 4.8: Power-saving performance normalized to that of *SPT* for $S_1: 128 \times 128$ MMUL service type with different deadline ranges

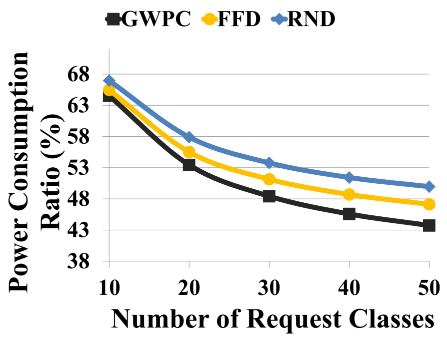
GWPC, *SPT*, *FFD*, and *RND* were tested using the same test cases. In each run, we generated 10^5 *IPR* with reneging on the VMs $VM_{i,j}; i = 1,2; j = 1,2,\ldots,m_i$. We repeated each run 10^4 times, calculated the average results, normalized them to the results by *SPT*, and presented them in Fig. 4.8 to Fig.4.11. Specifically, Fig. 4.8 and Fig. 4.9 show the power consumption of different approaches, and Fig 4.10 and Fig. 4.11 compare the total minimum processing rates required by each server pool (i.e., $\Omega(SP_i)$) to satisfy the given *QoS* constraints. From Fig. 4.8 and Fig. 4.9, we can immediately observe that *GWPC* outperforms the other three approaches under the different testing condition. For example, for S_1's request in Fig. 4.8, and when the deadlines are within the range of [50 - 550] μs, and the number of classes is 30, the power consumption by *GWPC* is about 46% of that by *SPT*. Whereas it is about 52% and 58% for *FFD* and *RND* of that by *SPT*, respec-



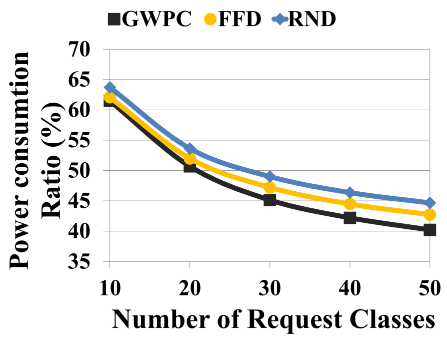
(a) $D_{2,j} \in [6-7] \times 10^2 \mu s$



(b) $D_{2,j} \in [700-800] \mu s$



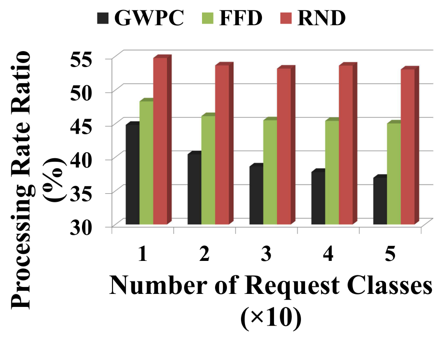
(c) $D_{2,j} \in [800-900] \mu s$



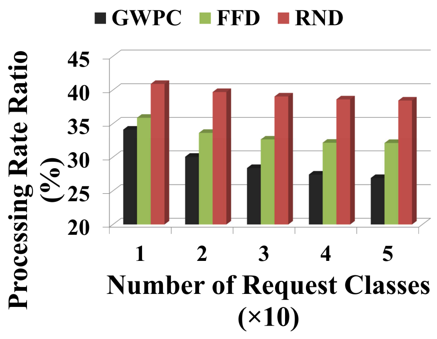
(d) $D_{2,j} \in [900-1000] \mu s$

Figure 4.9: Power-saving performance normalized to that of *SPT* for $S_2 : 1 \times 64$ 1-D FFT service type with different deadline ranges

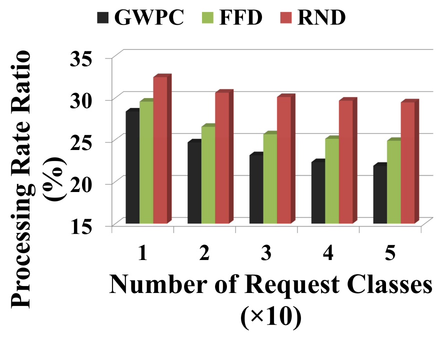
tively. We can also see that *GWPC*'s power-saving performance increases with increasing the number of classes, as well as with increasing the tightness of their deadline ranges. As the number of classes increases and the solution space to map requests to different VMs increases, *GWPC* can henceforth take advantage of the bigger solution space, and achieve better power-saving performance. Correspondingly, when the deadlines are long, or the workload intensity is light, the differences among packing approaches become smaller, and hence the power consumption patterns become similar. As the deadlines become larger and larger, the workload becomes heavier and heavier, and like this *GWPC* can greatly benefit from its effective request reneging and packing methods, and can consequently achieve higher and higher power-saving performance. For example, from Fig. 4.9(d) and Fig. 4.9(a), the power consumption ratios achieved by *GWPC* increase from



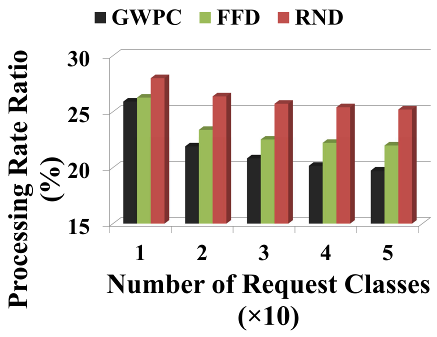
(a) $D_{1,j} \in [500-550]\mu s$



(b) $D_{1,j} \in [550-600]\mu s$



(c) $D_{1,j} \in [600-650]\mu s$

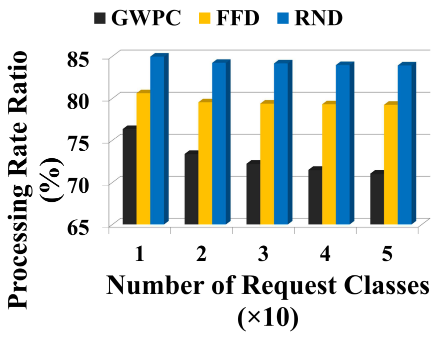


(d) $D_{1,j} \in [650-700]\mu s$

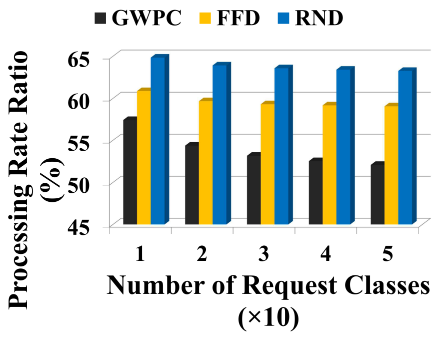
Figure 4.10: Processing rate performance normalized to that of *SPT* for S_1 : 128×128 MMUL service type with different deadline ranges

61% to 82.8% as request utilizations decrease (e.g., the deadline range changed from $[650-700]\mu s$ to $[500-550]\ \times 10^2\mu s$).

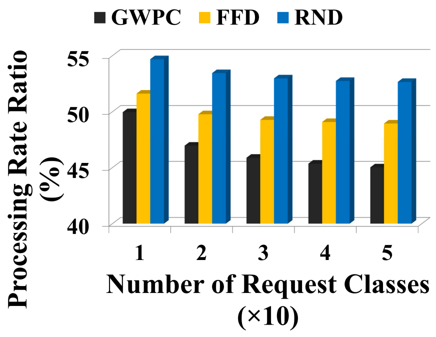
The behaviors of the minimum required processing rates and power consumption rates are closely related. Thus, it is not unforeseen to observe the similar phenomena, when studying our experimental results in terms of the minimum processing rates of the server pools. The total processing rates required by *GWPC* are lower than those required by the other three approaches for both types of services and under different testing cases, as shown in Fig. 4.10 and Fig. 4.11. We can also notice, from the same figures, that the improvement of *GWPC* over the other three approaches increases, when the class number for each service type increases, as well as when the tightness of the deadline ranges increase. Recall that, in Theorem 4.6, we prove that request packing is an NP-hard



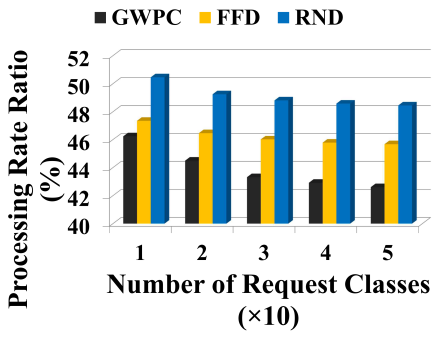
(a) $D_{2,j} \in [600 - 700]\mu s$



(b) $D_{2,j} \in [700 - 800]\mu s$



(c) $D_{2,j} \in [800 - 900]\mu s$



(d) $D_{2,j} \in [900 - 1000]\mu s$

Figure 4.11: Processing rate performance normalized to that of *SPT* for S_2 : 1×64 1-D FFT service type with different deadline ranges

problem [169], when a server pool SP_i needs more than two VMs. Yet, our experimental results clearly show that the heuristic proposed in *GWPC*, i.e., packing requests ordered by $\phi_{i,j}$ (see Equation 4.24) is much more effective than the one (i.e., *FFD*) that packs requests ordered by their individual processing rates (e.g., $\mu_{i,j}$) only. For example, in Fig. 4.11 for S_2 classes, when the class number is 30, the average total processing rate by *GWPC* is about 85% of that by *FFD*.

4.5.4 Performance under different server capacities

Different server capacities affect how many requests can be accommodated in a single server, and how many servers are needed to ensure the *QoS* requirements for a given set of service requests. In this section, we investigate how the performance of different allocation and packing approaches varies with different server's capacities, (e.g. $C_{i,j}$).

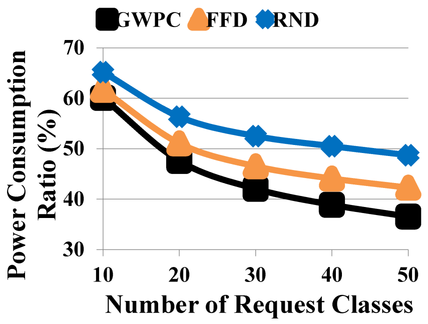
We generated two types of services, S_1 (128×128 MMUL) and S_2 (1×64 1-D FFT), with five testing groups of requests from each service type, and with each group has a different number of classes $\{\tau_{i,j}; i = 1,2; j = 1,...,r; r = 10,20,...,50\}$. The average arrival rates were randomly varied with the average following a uniform distribution in the range of $[20-500]$ *IPS* and $[500-1500]$ *IPS* for S_1's requests and S_2's requests, respectively. The deadlines of the requests were chosen randomly from the interval $[500 - 600]\mu s$. We varied the server's capacities from 6000 *IPS* to 6750 *IPS*, and from 6000 *IPS* to 12000 *IPS* for S_1 and S_2, respectively. We configured each VM with 2 GB memory, 20 GB disk storage, and a dedicated physical core with an adjustable maximum processing rate, i.e., capacity, according to a given value (e.g., $C_{i,j}$) using Xen's credit scheduler [145]. For example, to pin a VCPU to a single physical core, i.e., CPU, we can use *xl vcpupin VM_Name VCPU_ID CPU_ID*. Then to adjust the processing rate of a VCPU in a VM according to a given capacity $C_{i,j}$ in terms of RPS, we can use Xen Credit scheduler that assigns a Cap to the VCPU of a VM, which limits the maximum amount of the physical CPU that VCPU can use. For example, a VM with a 100 Cap means that the VM can consume up to a 100% of its CPU's maximum processing rate: *xlsched -credit -d VM_Name -c Cap_Value*. The arbitrarily chosen parameters and their values are summarized in Table 4.2(a), Table 4.2(b), and Table 4.2(c). For each experimental setting, we repeated each run 10^4 times, collected the total power consumption and the total minimum processing rates, and presented the average results normalized to those by *SPT* in Fig. 4.12 to Fig. 4.15.

Fig. 4.12 and Fig. 4.13 show how power consumption of different approaches vary with different capacities, and Fig. 4.14 and Fig. 4.15 show how total required processing rates change with different capacities. From the results, we can see that *GWPC* outperforms others in terms of both power consumption rates and total processing rate demands under different server capacities. As shown in the figures, when the VMs' capacities increase from 6500 *IPS* (Fig. 4.12(a)) and from 6500 *IPS* (Fig.4.13(a)) to 6750 *IPS* (Fig.4.12(d)) and to 12000 IPS (Fig.4.13(d)), the improvement of *GWPC* over *RND* and *FFD* diminishes for S_1 (S_2) service types. This is because when the VMs' capacities increase, more request classes can be hosted together in the same VM, and, thus, all multiplexing approaches show similar performance. Nevertheless, we can see that the performance improvement of power-saving and the processing rate demands by the multiplexing approaches over *SPT* approach continue to improve as the server's capacities grow larger. For example, in Fig. 4.12(a), when server capacity is 6000 IPS and the number of classes is 30, the power consumption by *GWPC* is about 53% of that by *SPT*. In Fig.4.12(d), when the server capacity is 6750 IPS and the number of classes is 30, the power consumption by *GWPC* becomes about 24% of that by *SPT*. This again conforms to theoretical conclusion in Theorem 4.3.

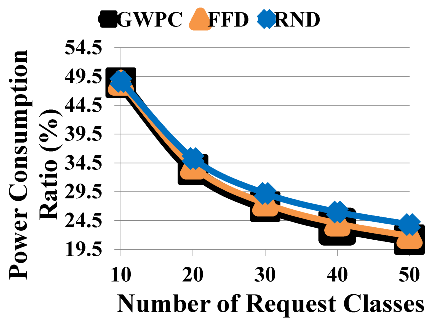
4.5.5 Validation using cloud benchmarks

We further evaluate our methods using the Data Caching Benchmark, i.e., a benchmark that emulates the behavior of a Twitter caching server, from the benchmark suite of cloud services, CloudSuite [162]. The benchmark assumes the strict quality of service guarantees such as 95% of the request must finish within 200 ms.

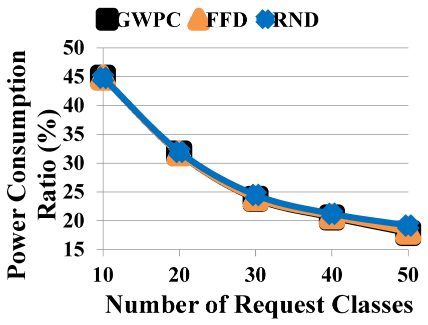
We exploited the *GCCP* platform, described in Chapter 3, to bootstrapped two VMs with 10 GB memory capacity, and a single VCPU pinned to a single physical core in each VM. We then implemented a Memcached server—a distributed memory object caching system that speeds up dynamic web applications by alleviating a database's delay—on the first VM with a single worker (e.g., a single execution queue) to process the data



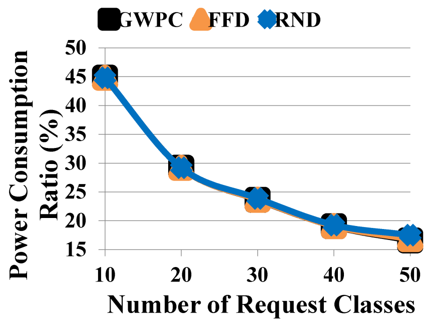
(a) $C_1 = 6000 IPS$



(b) $C_1 = 6250 IPS$



(c) $C_1 = 6500 IPS$

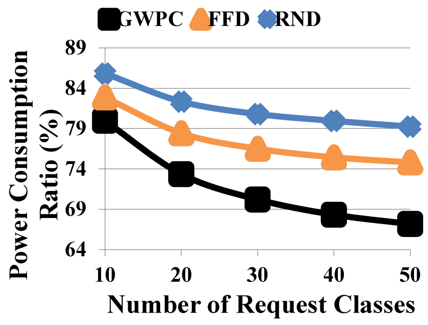


(d) $C_1 = 6750 IPS$

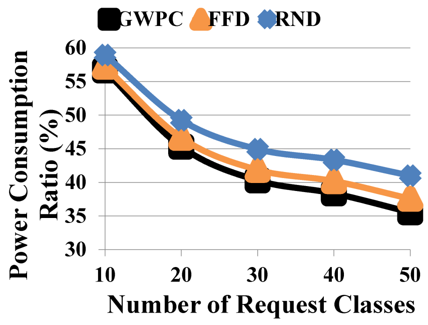
Figure 4.12: Power-saving performance normalized to that of *SPT* for S_1 : 128×128 MMUL service type with different server capacities

caching requests, i.e., *Get* or *Set*. The caching requests are generated by a Memcached client implemented within the workload engine UWME, and the Memcached server processes those requests. The UWME replicates those requests, and forward them to the second Memcached server running on the second VM with reneging. The Performance Monitoring Module collects the results on both servers.

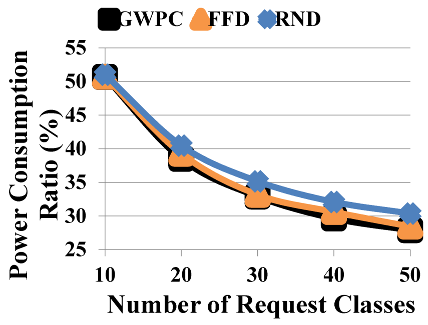
We considered two main service types generated by the UWME; i.e., *Get* (e.g., S_1), and *Set* (e.g., S_2). We generated one set of request with nine classes each from each service type, $\{\tau_{i,j} : i = 1,2; j = 1,...,9\}$, with arrival rates following the exponential distribution and averages of $\{1000, 2000, ..., 9000\}$ RPS. The completion ratios were set to be $R_{i,j} = 95\%$. The deadlines in both sets were set such that the maximum achieved



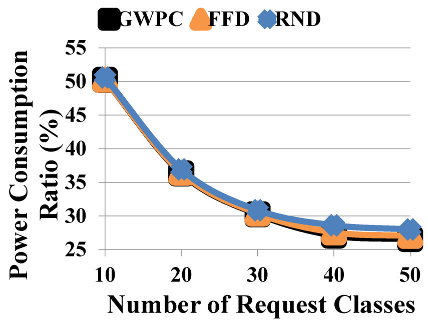
(a) $C_2 = 6000IPS$



(b) $C_2 = 8000IPS$



(c) $C_2 = 10000IPS$



(d) $C_2 = 12000IPS$

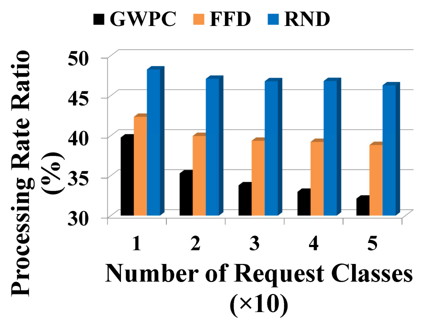
Figure 4.13: Power-saving performance normalized to that of SPT for $S_1 : 1 \times 64$ 1-D FFT service type with different server capacities.

throughput does not violate the target QoS requirements when the requests are served with the Memcached server without reneging.

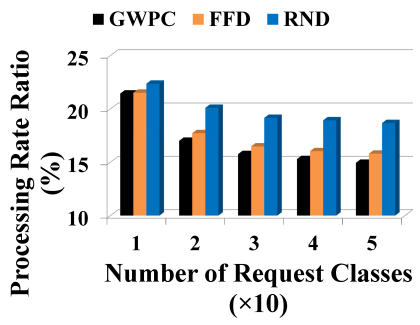
Data caching workload with request reneging

We first compare the average response times under QoS guarantees with and without request reneging. We generated 10^7 IPR, and processed these instances on both VM_1 and VM_2. The average results are shown in Figs. 4.16.

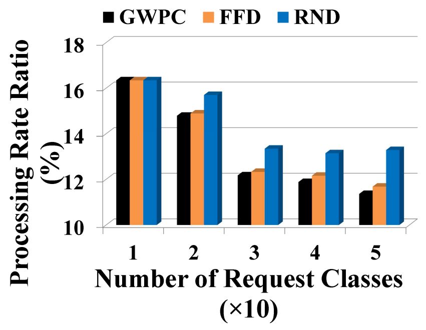
Our experimental results clearly show that request reneging can significantly reduce service response times. As shown in Fig. 4.16(a), the average response times of *Get* and *Set* classes without reneging are always longer than those with reneging. On average, the average response times of *Get* requests are 8%, 14%, and 17%, longer than those



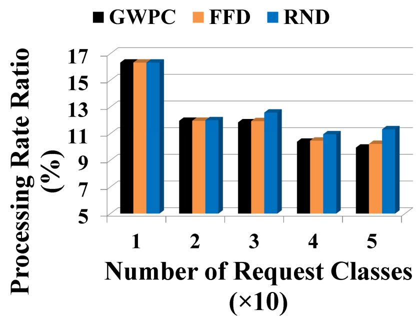
(a) $C_1 = 6000 IPS$



(b) $C_1 = 6250 IPS$



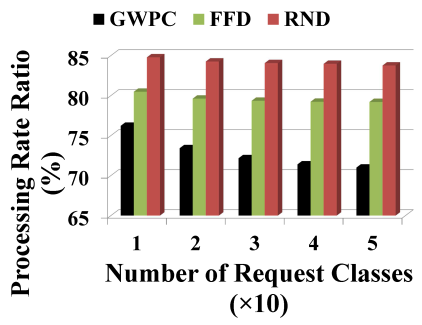
(c) $C_1 = 6500 IPS$



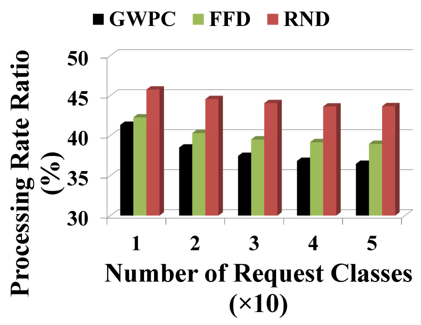
(d) $C_1 = 6750 IPS$

Figure 4.14: Processing rate performance normalized to that of *SPT* for $S_1 : 128 \times 128$ MMUL service type with different server capacities

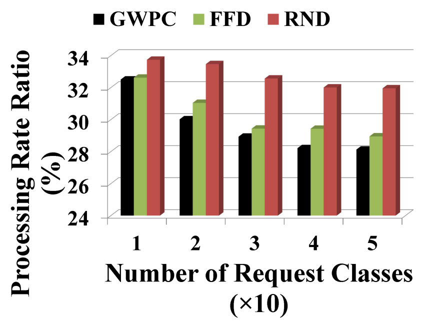
with reneging for Get requests classes with the arrival rates $\{\lambda_{1,1} = 1000, \lambda_{1,5} = 5000, \lambda_{1,9} = 9000\}$ RPS, respectively. We also observe a similar trend in Fig. 4.16(b), where the average response time of Set requests is on average 14% longer than those with reneging. When the arrival rates increase for S_1's and S_2's requests, the minimum required processing rates must increase to guarantee the same QoS requirement, as the waiting times of requests increased. From Fig. 4.16(a), and Fig. 4.16(b), we can also observe that the response time improvement by the reneging over non-reneging server increases with the arrival rate of the requests. For example, in Fig. 4.16(b), and when the arrival rate of Set requests is 5000 RPS, the response time by the reneging server is 14% shorter than the non-reneging server, and it becomes 19% when the arrival rate increases to 9000 RPS.



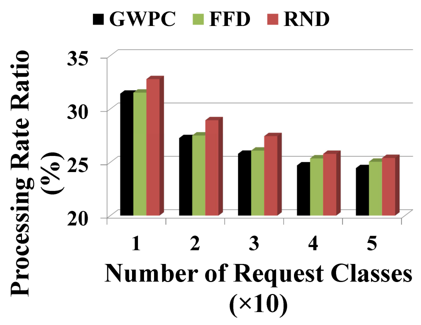
(a) $C_2 = 6000 IPS$



(b) $C_2 = 8000 IPS$



(c) $C_2 = 10000 IPS$



(d) $C_2 = 12000 IPS$

Figure 4.15: Processing rate performance normalized to that of *SPT* for $S_1 : 1 \times 64$ 1-D FFT service type with different server capacities.

Energy-saving performance of data caching workload with request reneging

To study the potential electricity cost savings of our approach, we tested *GWPC*, *SPT*, *FFD*, and *RND*, using the same test cases defended above, with different request classes each time ranging from 10 to 50. We also varied the completion ratio for each class randomly from the interval of [90%,99%]. We assume that each HP workstation with a total memory capacity of 32GB can hold up to 3 Memcached servers. We generated 10^5 *IPR* for each request class and repeated each run 10^4 times with and without reneging. The results are taken among different completion ratios, and service utilization averaged and normalized to the results by *SPT* and presented them in Fig. 4.17. More details about power measurement and electricity cost calculation can be found in Section 3.2.2.

Table 4.2: Power-saving performance of GWPC with different server capacities

(a) Services and Nodes Specifications

S_i	SP_i 's Size (m_i Nodes)	C_i textbf(IPS)
$S_1 : 128 \times 128$ MMUL	$m_1 = \{10, 20, 30, 40, 50\}$	$C_1 = (6, 6.25, 6.50, 6.75) \times 10^3$
$S_2 : 1 \times 64$ 1-D FFT	$m_2 = \{10, 20, 30, 40, 50\}$	$C_2 = (6, 8, 10, 12) \times 10^3$

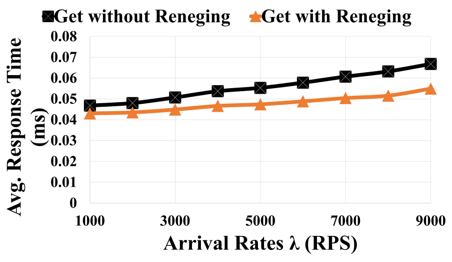
(b) Requests Specifications for S_1 and S_2

$\tau_{i,j}$	$\lambda_{i,j}$ **(IPS)**	$V_{i,j}$	$C_{i,j}$ **(IPS))**	**Run ID**
$(\tau_{1,1} \ldots \tau_{1,r1})$	$[20-500]$	$V_{1,1} \ldots V_{1,m1}$	6×10^3	(0 to 4)
$(\tau_{1,1} \ldots \tau_{1,r1})$	$[20-500]$	$V_{1,1} \ldots V_{1,m1}$	8×10^3	(5 to 9)
$(\tau_{1,1} \ldots \tau_{1,r1})$	$[20-500]$	$V_{1,1} \ldots V_{1,m1}$	10×10^3	(10 to 14)
$(\tau_{1,1} \ldots \tau_{1,r1})$	$[20-500]$	$V_{1,1} \ldots V_{1,m1}$	12×10^3	(15 to 19)
$(\tau_{2,1} \ldots \tau_{2,r2})$	$[500-1500]$	$V_{2,2} \ldots V_{2,m2}$	6×10^3	(20 to 24)
$(\tau_{2,1} \ldots \tau_{2,r2})$	$[500-1500]$	$V_{2,2} \ldots V_{2,m2}$	8×10^3	(25 to 29)
$(\tau_{2,1} \ldots \tau_{2,r2})$	$[500-1500]$	$V_{2,2} \ldots V_{2,m2}$	10×10^3	(30 to 34)
$(\tau_{2,1} \ldots \tau_{2,r2})$	$[500-1500]$	$V_{2,2} \ldots V_{2,m2}$	12×10^3	(35 to 39)

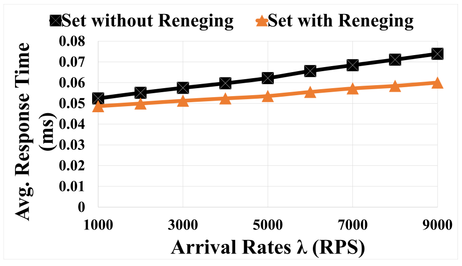
(c) Shared Parameters among Requests

$R_{i,j}$ **(%)**	$IPR_{i,j}$	**Run Repetition**	r_i	$D_{i,j}(10^2 \mu s)$
$[90-95]$	10^5	10^4	$(10, 20, 30, 40, 50)$	$[5-6]$

The annual electricity cost-saving performance of *GWPC* follows a similar pattern of the power-saving performance, and total processing rate as our experimental results showed before. First, compared with *SPT*, all three request multiplexing approach (i.e., *GWPC*, *SPT*, and *RND*), by sharing servers and reneging requests, can improve the processing efficiency and reduce the electricity cost significantly. In addition, as the number of classes increases, the improvement becomes more significant. For example, when the number of classes increases from 10 to 50, the annual electricity cost ratios over *SPT* by *GWPC*, *FFD* and *RND* decrease from 38 (35, resp.), 39 (36, resp.), and 40 (36, resp.) to 19 (16, resp.), 21 (17, resp.), and 25 (20, resp.) for S_1 (and S_2, resp.), respectively. Moreover, we can see that *GWPC* outperforms *FFD* and *RND*. According to Fig. 4.17(a)



(a) Get's Request Classes



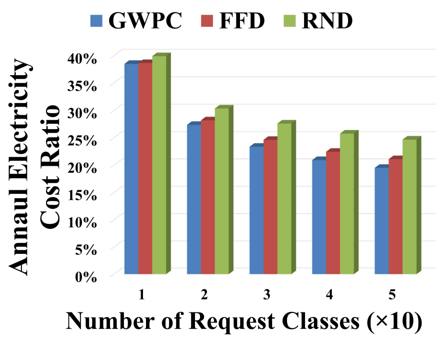
(b) Set's Request Classes

Figure 4.16: Get's (a) and Set's (b) average response times under different arrival rates with and without request reneging

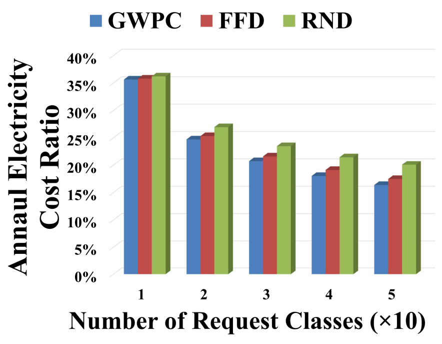
and Fig. 4.17(b), *GWPC* on average saves around 4% and 3% more than *FFD* and 12% and 10% more than *RND* for S_1's and S_2's requests, respectively. Furthermore, such an improvement increases while the number of classes increases. For example, when the number of classes is 10, the relative improvement of *GWPC* over *FFD* and *RND* is 2.6% (2.8%, resp.) and 5.3% (2.8%, resp.) for S_1 (S_2, resp.) requests, respectively. When the number of classes is 50, the relative improvement of *GWPC* over *FFD* and *RND* becomes 10.5% (6.25%, resp.) and 31.6% (25.0%, resp.) for S_1 (S_2, resp.) requests, respectively. This is because that when the number of classes increases, the solution space to map requests to different VMs increases, and *GWPC* consequently can take advantage of the bigger solution space, and achieve better power-saving performance. Overall, our experimental results have clearly demonstrated that request reneging with data caching workload results in lower average response times, and less energy costs comparing to its counterpart. *GWPC* is therefore a promising approach that should be studied and applied to more complicated cloud workloads.

4.6 Summary

In this chapter, we discuss our novel approach that can be applied in virtualized data centers to optimize resource usage and to minimize power consumption when delivering



(a) Get's Request Classes



(b) Set's Request Classes

Figure 4.17: Energy-saving performance of GWPC normalized to that of *SPT* using Get's and Set's request classes under various QoS conditions

cloud services with statistically guaranteed *QoS*. The effectiveness and efficiency of our novel methodology are rooted in two facts. First, our approach can effectively remove potential failure requests as soon as possible to improve resources usage. Second, our approach allows requests with different *QoS* requirements to be served on the same VM. To the best of our knowledge, this is the first approach by which different requests with different *QoS* guarantees can be hosted on a single node to increase resource utilization further and to reduce power consumption. We present several interesting characteristics of our proposed approach with formal proofs. We also present the *GWPC* algorithm that allocates services on the same VMs and reneges potential failure request while statistically guarantees *QoS* constraints in terms of deadline miss ratios. We also design the *GCCP* cloud prototype to validated the *GWPC* algorithm, and our experimental results confirm that our approach can significantly outperform other traditional approaches in terms of guaranteed *QoS* levels, power consumption, resource demands, as well as electricity costs. For the future work, we will extend this work to platforms with heterogeneous servers having different power consumption and processing rates characteristics. We also plan to study the effects of interferences among different service types on the performance of cloud data centers modeled by exploiting different queue models with request reneging.

Next, we study how to securely collocate critical VMs and non-critical VMs onto cloud clusters.

CHAPTER 5

SECURE ALLOCATIONS OF CRITICAL VMS IN CLOUD DATA CENTERS WITH RELAXED AND CONSTRAINED RESOURCE USAGE AND POWER CONSUMPTION

Virtual Machine (VM) multiplexing in the cloud optimizes resource usage and power consumption. However, consolidating VMs of different security requirements on a single server results in cybersecurity threats, such as the VM-to-VM Interdependent Cybersecurity (*IC*) risks. For example, the odds of successfully compromising a secure Critical VM (CVM) are high when an attacker, i.e., a hacker, compromises the hosting hypervisor after a successful direct attack on one of its less secured, Non-critical VMs ($NVMs$).

In this chapter, we study how to securely and efficiently allocate Critical and Non-critical VM types (e.g., $CVMs$ and $NVMs$) onto a cloud cluster. Specifically, we formulate the allocation problem using non-cooperative zero-sum and non-zero-sum game models, between a cloud service provider and an attacker, under relaxed and constrained resource usage and power consumption constraints. Our analysis completely characterizes the existence of all the static and dynamic *Equilibrium* strategies of the provider. We mean by an –*Equilibrium*– strategy that neither the provider nor the attacker can gain more if the provider deviates from that *Equilibrium* strategy.

5.1 Introduction to the research problem

Cloud service providers usually allow several users and applications to share the resources of the cloud infrastructure to maximize resource utilization and minimize energy costs. Packing several VMs on a single server reduces the total number of allocated servers which, in turn, minimizes the power consumption and operating expense (e.g., software licenses, cybersecurity investment, etc.) of cloud clusters [126], [170]. Nevertheless, consolidating VMs with different security requirements (e.g., *CVMs* and *NVMs*) impose security risks on both types of VMs. We refer to those risks as VM-to-VM Interdependent Cybersecurity (*IC*) risks [33], and we call the attack in which an attacker can compromise the hypervisor, after a successful direct attack on one of its hosted VMs, and eventually can compromise all its hosted *CVMs* and *NVMs* as an *IC* attack.

Compromising critical applications could result in a complete functional failure of a whole system (e.g., mission-critical applications, like space navigation systems), could lead to a financial crisis (e.g., business-critical applications, like banking systems), could have a catastrophic impact on the environment or human lives (e.g., safety-critical, like nuclear reactor safety system), or could result in a leakage of classified data (e.g., security-critical, like hospital storage systems). VMs that host critical applications, in consequence, are Critical VMs (*CVMs*) and must be highly secured and must run in safe environments that reduce technology cost.

Cloud data centers are primarily built with global scalability and are profoundly resilient to optimizing the costs of cloud service providers. According to an article published in The Business of Data Centers [171], many agencies, which operate critical workloads, are yet reluctant to make the transition to the cloud due to cybersecurity concerns. According to the Datacenter Dynamic [40], the U.S. federal CIO started the Government-wide Data Center Consolidation and IT Modernization Program in 2010. This program shut down thousands of national, private data centers and enforced the adoption of efficient resource management policies pushing most agencies to outsource their IT infrastructure to the cloud. Any general online search quickly reveals numerous examples of cloud providers hosting critical workloads (e.g., Microsoft Azure government cloud computing [172] and the Hybrid Critical Cloud by Virtustream and VMware [173]).

Cloud providers permit their clients to have full control over the security settings and policies of their critical workloads, as reported by several cloud providers (e.g., the

Mission Critical Cloud [174]). VM multiplexing methods are extensively used in cloud data centers to maximize server sharing among VMs, applications, and users. Resource sharing, unfortunately, exposes $CVMs$ to IC risks resulted from the coexistence with other vulnerable or less secured VMs, such as $NVMs$ [33, 52].

On the contrary, the Split allocation policy, which allocates $CVMs$ and $NVMs$ to separate servers, incurs lower cybersecurity risks. It, however, exacerbates resource usage and energy costs. The challenging question therefrom is – How can cloud service providers optimize the tradeoff between minimizing the IC risks and optimizing energy costs and operating expense of cloud clusters?– Our problem is, thus, a Multi-Objective Optimization (MOO) problem with conflicting goals and non-cooperative decision-makers.

Although MOO methods (e.g., Mathematical Programming (MP) and Evolutionary Multi-objective Optimization (EMO)) are useful methods in solving problems with multiple conflicting objectives [175, 176], we choose the Game theory to formulate our VM allocation problem. Our rationale is that the Game theory offers "*mathematical models of conflict and cooperation between cooperative and non-cooperative intelligent, rational decision-makers*" [135]. Furthermore, non-cooperative games model all the procedural details of the game, unlike MOO methods which only describe the structure, strategies, and payoffs of coalitions. For example, non-cooperative games can provide a socially optimal way to deal with the different behaviors of cybersecurity attackers [177].

In this chapter, we study how to allocate two types of VMs (e.g., $CVMs$ and $NVMs$) onto a cloud cluster so that the provider's Worst-case Potential Cybersecurity (WPC) loss is minimized, while the overall energy costs and operating expense of the cloud cluster are optimized. First, we formulate the VM allocation problem with cybersecurity awareness using a non-cooperative, zero-sum game model between an attacker and a service cloud provider. Second, we incorporate the processing rate utilization, power consumption, energy costs, and operating expense of the cluster into a non-zero-sum game model. Our analysis completely characterizes the existence of all the static and dynamic *Equilibrium* strategy profiles with relaxed and constrained resource usage and cost. We mean by an –*Equilibrium*– profile that a pair of an attacker strategy and a provider strategy in which neither the provider nor the attacker can gain more from unitarily deviating from his/her strategy. We also derive the lower-bound and upper-bound of the *IC* risks.

5.2 Related work

We can divide the most related literature works to our problem into three major categories, i.e., cloud resource allocation without considering cybersecurity risks, cloud security without focusing on resource usage optimization, and security-aware cloud resource allocation. Each of those categories can further be divided into approaches that use game-theoretic based models or use other types of models.

Cloud online services entered in each sector of our lives. Accordingly, optimizing resource utilization and power consumption became a necessity for providers who are always eager to attract more customers and survive in the competing cloud-based service market. Numerous heuristics and meta-heuristics (e.g., genetic algorithm, fuzzy logic, etc.) have been used in the literature to model cloud resources and IoT services over the cloud [126, 170, 175, 176]. Makhlouf et al. [175] suggested using a linear integer program to optimize the partitioning of the cloud workload among the federation members, while dynamically optimizing the provider's prices to achieve the highest revenue. Gai et al. [176] proposed a Cost-Aware Heterogeneous Cloud Memory Model (*CAHCM*) to provision a high-performance cloud-based memory service offering. Their model depends on a Dynamic Data Allocation Advance (2*DA*) Algorithm that employs genetic programming to allocate data to cloud-based memories efficiently. As heuristic and meta-heuristic methods tend to include external resources of uncertainties into the *MOO* model, they increase the complexity of the problem [178]. In addition to the complexity, all previously mentioned approaches analyze the tradeoffs among local objectives in isolation of the dynamic of the different running scenarios of the optimization problem. Even if several cases are taken into consideration, those approaches cannot simultaneously optimize multiple objectives from several actors. Such interaction, fortunately, can be captured using the Game theory, which is crucial in predicting the attacker's behavior when modeling multi-objective security models.

Game theory-based approaches applied in cloud resource management were proposed in [127, 129, 138]. Wei et al. [127] suggested improving the efficiency of cloud resources by proposing two, non-multiplexing and multiplexing, resource assignments using an evolutionary game model. Jalaparti et al. [138] introduced multiple heuristic algorithms with near-optimal allocation and pricing policies, modeling the client-provider and client-client interactions, compared to the fixed-pricing methodologies used today by cloud providers, such as in Amazon EC2. While all such methodologies effectively optimize resource utilization and/or cost savings, none of them consider the cybersecurity risks implications of their allocation algorithms.

Server sharing results in challenging security threats, such as the side-channel attacks [51, 52], and the *IC* risks [33, 55]. In the side-channel attacks, an attacker can build different kinds of side channels to extract private information about the victim's VM (e.g., the victim's workload, traffic rate [52], cryptographic keys [51], etc.,) that eventually enables him/her to launch a successful attack.

On the other hand, the authors in [33, 55] used a non-zero-sum game model to study how to minimize the security expense of the cloud users. They assumed that users could either allocate their VMs to a secure server, i.e., users invest in cybersecurity countermeasures, or to an insecure server to save in the cybersecurity-related expense. They focused on minimizing the cybersecurity risks for a targeted user while reducing his/her cybersecurity related-expense. Whereas it is unrealistic to assume that a service cloud provider would operate a cloud cluster with servers openly vulnerable to cyber attacks, we presume, in this chapter, that a provider always secures all servers and VMs but with different security levels. To the best of our knowledge, our work is the first to model the *IC* risks, processing rate utilization, energy costs, and operating expense in cloud data centers using the Game theory. Our goal is to ensure that the provider's resource usage and cybersecurity loss are optimized.

Security-aware resource allocation techniques were introduced in [90,141,179]. Zhang et al. [90] are the first to develop a formal and quantified migration strategy to improve the security against collocation attacks and with acceptable cost. Han et al. [141] used the Game theory to model the cybersecurity between an attacker and a cloud provider. The provider aims to minimize the attacker's possibility of collocating his/her VMs with the target VMs while maintaining a satisfactory workload balance and low power consumption. Using a game-solver software, they concluded that rather than adopting a single allocation policy, a provider should have a pool of VM allocation policies. In [179], the authors considered both direct and side-channel attacks on hypervisors. They developed VM migration-based techniques to reduce the security risks considering the connection cost among VMs while improving the survivability of the system. Rao et al. [143] used the Game theory to study the ability of a cloud provider to guarantee the survivability of predefined resource capacity and with a certain probability after a physical or cybersecurity attack on his/her data center. They proposed the usage of reinforcement strategies to decrease an attacker's utility. To this end, we assume that a provider secures $CVMs$ and $NVMs$ using two different types of cyber protection measures. We aim to model the *IC* risks and resource usage in cloud clusters using zero and non-zero sum game to investigate how to allocate $CVMs$ and $NVMs$ to a two-server cluster so that the *IC* risks are mitigated, and the overall energy costs, operating expense, and the WPC loss are minimized.

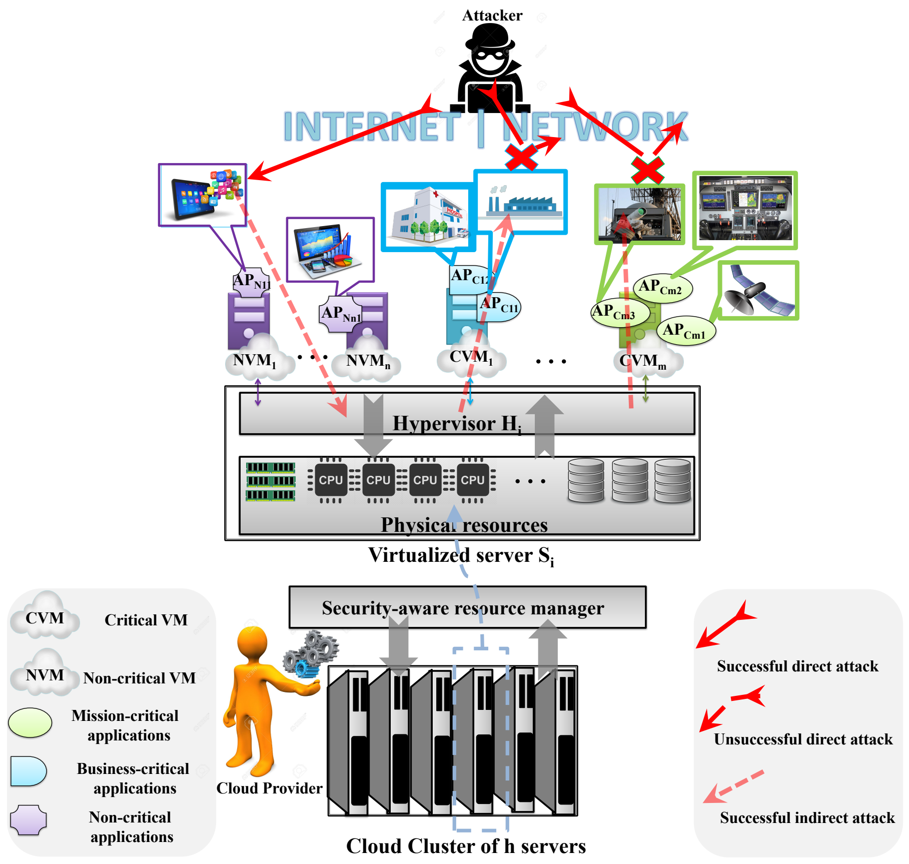


Figure 5.1: A cloud cluster model. The cluster consists of h hypervisors, each of which is hosted on a physical server $S_i; i = 1, 2, ..., h$. A server has a processing rate capacity $C_i = \mathbf{C}$ $MIPS$., utilization U_i, and power consumption P_i. The cluster hosts n Non-critical VMs (e.g., NVM) and m Critical VM (e.g., $CVMs$), that is $\{NVM_1, NVM_2, ..., NVM_n\}$ and $\{CVM_1, CVM_2, ..., CVM_m\}$, respectively. ℓ_T, q_T, and c_T are the maximum security loss after a successful attack on a VM_T, the security level, i.e., the successful attack probability on a VM_T, and the processing rate requirements of VM_T; where $VM_T = NVM$ or $VM_T = CVM$. q_h is the probability of successfully compromising any hypervisor after a successful direct attack on any of its hosted VMs.

5.3 System models

In this section, we introduce our system models.

5.3.1 Cloud cluster model

Cloud data centers usually consist of hundreds or thousands of clusters. Fig. 5.1 illustrates a virtualized cloud cluster with h hypervisors, each of which is hosted on a physical server S_i. Each server has a processing rate capacity $\mathbf{C}$ MIPS, utilization U_i. The cluster hosts n Non-critical VMs and a m $CVMs$, e.g.,$\{NVM_1, NVM_2, ..., NVM_n\}$ and $\{CVM_1, CVM_2, ..., NVM_m\}$, respectively. ℓ_T, q_T, and c_T are the maximum potential security losses after a successful attack on VM_T, the security level of VM_T, and the processing rate requirements of VM_T; where $T = C$ for the $CVMs$ and $T = N$ for the $NVMs$. q_h is the probability of successfully compromising any hypervisor after a successful direct attack on any of its hosted VMs first.

5.3.2 Cloud service model

In this section, we model the VM allocation problem with cybersecurity and resource usage awareness using game-theoretic model between an attacker R_A and the cloud service provider R_V. We assume that both players are reasonable, have a full understanding of the cloud platform, and can only take the actions that maximize their potential payoffs.

R_A's goal is to indirectly compromise a hypervisor and all its hosted VMs after successfully and directly attacking one of its hosted VMs. We assume that R_A has just enough time to compromise one server. We accept the fact that R_A is capable of indirectly compromising a hypervisor via one of its directly compromised VMs, as proven experimentally in [52]. Once R_A has full control over the hypervisor, he/she can easily compromise all VMs hosted on that hypervisor [179]. Specifically, R_A's strategies identify the VM on which he/she launches his/her initial direct attack to use it later in compromising its hosting hypervisor and all its coexisting VMs. Therefore, R_A can either directly attack an

NVM, i.e., a strategy we denote as N, or directly attack a CVM, i.e., a strategy we denote as C.

R_V can either allocate the VMs using the Split (e.g., S) allocation, in which different types of VMs are allocated separately to a single server or using the multiplexing (e.g., X) allocation in which both types are hosted on a single server. The goal of a provider is to select the allocation strategy that minimizes his/her potential cybersecurity Loss (e.g., L) assuming R_A is capable of attacking any VM in the cluster that maximizes his/her potential Gain (e.g., G). Therefore, this loss represents the provider's Worst-case Potential Loss (e.g., WPC) loss under a given VM allocation strategy.

Equation 5.1 shows the set of the four possible strategy profiles for R_A and R_V.

$$Possible\ Strategy\ Profiles = \{(N,X),(N,S),(C,X),(C,S)\} \tag{5.1}$$

For example, the strategy profile (C,S) implies that R_A attacks a CVM when R_V chooses the S allocation policy. We assume that the relative importance of a VM is determined by the potential expected loss for R_V if that VM is compromised. Each strategy profile results in a different payoff to R_V and R_A.

We denote R_A's or R_V's cybersecurity gain or loss if R_A successfully compromises an NVM, or a CVM, as ℓ_N and ℓ_C. Further, it is rational to assume that:

$$\ell_N < \ell_C \tag{5.2}$$

This implies that R_V will suffer more loss if R_A successfully compromises a CVM instead of an NVM. Therefore, $CVMs$ require higher security levels, i.e., security measures and cost, than those of the $NVMs$ to be protected against any direct cybersecurity attack.

We can define the security level of a VM as follows:

Definition 5.1. *The Security Level of a VM is the probability of a direct and successful attack on that VM. If the success probability is substantial (e.g., $q \to 1$), the VM is said to* *have a low-security level. If the success probability is tiny (e.g., $q \to 0$), the VM is said to have a high-security level.*

For example, the security levels of an NVM and CVM are denoted as q_N and q_C, respectively. It is consequently rational to assume that:

$$0 < q_C < q_N < 1 \tag{5.3}$$

The probability of a successful attack on any hypervisor after one of its VMs has been compromised by R_A is denoted as:

$$0 < q_h < 1 \tag{5.4}$$

High-security levels, nonetheless, do not protect VMs from other indirect cybersecurity attacks (e.g., an *IC* attack) that can occur when they coexist with vulnerable VMs on the same hypervisor. We can define the indirect cybersecurity attack as follows:

Definition 5.2. *The indirect cybersecurity attack is an attack on a hypervisor via one of its directly compromised VMs, or an attack on another VM using that compromised hypervisor to bypass any security measure applied to them.*

For example, R_A is unable to launch a successful direct attack on any of the $CVMs$ when they have high-security levels, as illustrated in Fig. 5.1. R_A can rather indirectly compromise them by launching an indirect attack involving any of the $NVMs$ and the hypervisor. We call such an indirect attack on the $CVMs$ an *IC* attack and call such cybersecurity risks *IC* risks.

Definition 5.3. *The Interdependent Cybersecurity (e.g., IC) attack is an indirect cybersecurity attack on targeted VMs, usually with high-security levels, in which R_A involves another VM, usually with a low-security level, and the hosting hypervisor to bypass any sound security levels applied to those targeted VMs.*

Definition 5.4. *The IC risks imposed on a VM_i is R_V's potential security loss due to a successful IC attack on that VM_i, after a successful and direct attack on another coexisting VM_j, with a security level q_j, and followed by an indirect attack on the hypervisor hosting both VMs with a success probability q_h. Consequently, the IC risks imposed on VM_i by VM_j are $q_j q_h \ell_i$.*

When an *IC* attack on any VM under a given allocation strategy is optimal in terms of maximizing R_A's potential cybersecurity gain, we call this attack the Worst-case Potential Cybersecurity WPC attack from R_V's perspective.

Using those previously introduced definitions, we can now define the *Equilibrium* strategy.

Definition 5.5. *The allocation strategy is a static or dynamic* Equilibrium *strategy if it optimally minimizes R_V's WPC loss*

In other words, the static *Equilibrium* allocation strategy corresponds to a pure Nash Equilibrium (NE) strategy profile at which neither R_V nor R_A can improve their payoffs by unitarily deviating to another strategy.

Let $E(X)$ and $E(S)$ be R_V's operating expense under the allocation policies X and S, respectively. Operating expense includes the cost of renting or purchasing servers, security investment, hardware/software upgrade, licenses, etc. The operating cost of X and S allocation strategies, according to Fig. 5.3 are:

$$E(X) = E_X \tag{5.5}$$

$$E(S) = 2 \times E_X \tag{5.6}$$

We see that it costs R_V an extra E_X units when using the S strategy compared to using the X strategy, in which R_V acquires fewer servers.

Let $P(X)$ and $P(S)$ be the cluster's total, static and dynamic, power consumption under the allocation policies X and S, respectively. For instance, when $CVMs$ and $NVMs$ coexist on a single server, i.e., the X allocation strategy, the static power consumption of the cluster is minimized due to reducing the number of utilized servers. Equation 6.7 models the total power consumption consumed by a cloud cluster, as in the previous chapter:

$$P(X \; or \; S) = (DP \cdot Utz) + SP \tag{5.7}$$

DP is the maximum dynamic power consumed by all servers in the cluster; Utz is the utilization of all servers (e.g., $Utz = \sum_{i=1}^{h} U_i$; where $\{i = 1, 2, ..., h\}$). SP is the static power of the non-idle servers in the cluster. The total power consumption for the X and S allocation strategies are given in Equation 5.8 and Equation 5.12, respectively.

$$P(X) = DP \times (Utz_X) + SP \tag{5.8}$$

$$P(S) = 2 \times (DP \times (Utz_S) + SP) \tag{5.9}$$

Let:

$$P_{svg} = P(S) - P(X) \tag{5.10}$$

Where ($P_{svg} > 0$) represents the savings in the cluster's total power consumption when switching from the allocation strategy S to the strategy X. For ease of representation, we will write the power equations as:

$$P(X) = P_X \tag{5.11}$$

$$P(S) = P_X + P_{svg} \tag{5.12}$$

In this chapter, we ignore the VM-to-hypervisor IC risks, i.e., R_A cannot directly attack a hypervisor. Next, we introduce our research problem.

5.4 Problem definition

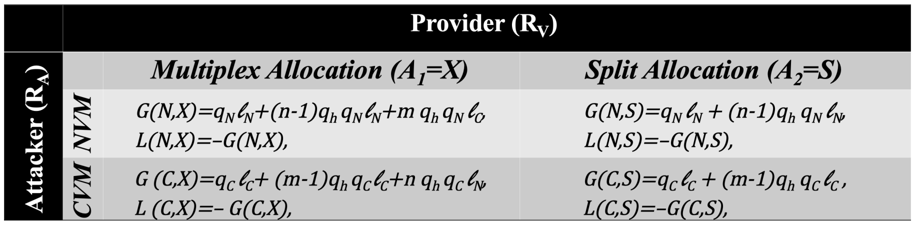
With the system and service models defined above, we can define our allocation problem as follows:

Problem 5.1. *Given the cloud cluster described in Section 5.3 with $h = 2$ servers, $(n \geq 1)$ $NVMs$ and $(m \geq 1)$ $CVMs$. Determine if R_V should multiplex both types of VMs onto a single server (e.g., the X allocation strategy), or if R_V should split them off across two separate servers (e.g., the S allocation strategy) so that R_V's WPC loss is minimized and the overall power consumption and operating expense is optimized.*

5.5 Security-aware allocation with unconstrained resource usage

We first analyze our VM allocation problem assuming no resource usage and power consumption constraints via formulating it into a zero-sum game model. The normal form of the game model is illustrated in Fig. 5.2.

We denote R_A's and R_V's Gain and Loss as G and L, respectively. As illustrated in Fig. 5.2, the strategy profile $G(N,X)$ denotes R_A's gain when directly attacking an NVM under the multiplexing allocation strategy, i.e., (N,X). R_A's gain at this strategy profile is, thus, equal to the probability of a successful attack on an NVM (e.g., q_N) times the maximum security gain from attacking that NVM (e.g., ℓ_N). Because both $NVMs$ and $CVMs$, are packed on the same server (e.g., the X strategy), R_A can indirectly attack the hypervisor and eventually compromise all its VMs. Therefore, the potential security loss for R_V if R_A compromises the rest $(n-1)$ VMs upon compromising the hypervisor is $((n-1) \times q_h \times q_N \times \ell_N)$. Similarly, the potential security loss from compromising all m $CVMs$ via launching an IC attack on them is $(m \times q_h \times q_N \times \ell_C)$.



		Provider (R_V)	
		Multiplex Allocation (A_1=X)	***Split Allocation (A_2=S)***
Attacker (R_A)	***NVM***	$G(N,X)=q_N\ell_N+(n-1)q_h q_N\ell_N+m\, q_h q_N \ell_C$, $L(N,X)=-G(N,X)$,	$G(N,S)=q_N\ell_N + (n-1)q_h q_N \ell_N$, $L(N,S)=-G(N,S)$,
	CVM	$G(C,X)=q_C\ell_C+ (m-1)q_h q_C\ell_C+n\, q_h q_C \ell_N$, $L(C,X)=- G(C,X)$,	$G(C,S)=q_C\ell_C + (m-1)q_h q_C\ell_C$, $L(C,S)=-G(C,S)$,

Figure 5.2: The zero-sum security game in normal form

Similar reasoning is applied to calculate R_A's gain at other strategy profiles. We can see that while R_A is trying to maximize his/her gain, R_V is trying to minimize his/her expected loss. Therefore, R_V's loss is $L = -G$, i.e., a zero-sum game.

R_V wishes to minimize his/her cybersecurity loss assuming that R_A can attack the VM that maximizes his/her potential cybersecurity gain, i.e., R_V wants to minimize his/her WPC loss. Therefore, we need to find the strategy profiles at which no player has an advantage in deviating from his/her current decision after considering all the opponent's possible choices. We call such strategy profile an NE [135].Our allocation problem is for this reason a problem of identifying the NE strategy profile of our game model.

In order to identify the existence of the NE strategy profiles, we first have to analyze R_A's attack preferences under X and S allocation policies using the following lemmas and theorems.

Lemma 5.1. *R_A always and directly attacks a CVM under the allocation X if:*

$$q_h < q_{h_0} \tag{5.13}$$

$$q_C\ell_C > q_N\ell_N \tag{5.14}$$

where q_h is defined in Equation 5.15.

$$q_{h_0} = \left(\frac{((q_N - q_C)(n\ell_N + m\ell_C)}{(q_C\ell_C - q_N\ell_N))}\right)^{-1} \tag{5.15}$$

Proof. R_A prefers the strategy profile (C,X) over (N,X), if:

$$G(C,X) > G(N,X) \tag{5.16}$$

$$q_C\ell_C + (m-1)q_hq_C\ell_C + nq_hq_C\ell_N > q_N\ell_N + (n-1)q_hq_N\ell_N + mq_hq_N\ell_C \tag{5.17}$$

$$q_C\ell_C - q_N\ell_N > (n-1)q_hq_N\ell_N + mq_hq_N\ell_C - (m-1)q_hq_C\ell_C - nq_hq_C\ell_N \tag{5.18}$$

$$q_C\ell_C - q_N\ell_N > nq_hq_N\ell_N - nq_hq_C\ell_N + mq_hq_N\ell_C - mq_hq_C\ell_C + q_hq_C\ell_C - q_hq_N\ell_N \tag{5.19}$$

$$q_C\ell_C - q_N\ell_N > q_h(n\ell_N(q_N - q_C) + m\ell_C(q_N - q_C) + (q_C\ell_C - q_N\ell_N)) \tag{5.20}$$

$$q_C\ell_C - q_N\ell_N > q_h((n\ell_N + m\ell_C)(q_N - q_C) + (q_C\ell_C - q_N\ell_N)) \tag{5.21}$$

Note that:

$$q_h((n\ell_N + m\ell_C)(q_N - q_C) + (q_C\ell_C - q_N\ell_N)) > 0 \tag{5.22}$$

because if

$$q_h((n\ell_N + m\ell_C)(q_N - q_C) + (q_C\ell_C - q_N\ell_N)) < 0 \tag{5.23}$$

then using simple transformation, we get:

$$\frac{q_N}{q_C} < \frac{(n\ell_N + m\ell_C - \ell C)}{(n\ell_N + m\ell_C - \ell_N)} \tag{5.24}$$

but $q_N > q_C$, and $\ell_C > \ell_N$ according to Equation 5.3 and Equation 5.2, respectively. Therefore:

$$\frac{q_N}{q_C} > \frac{(n\ell_N + m\ell_C - \ell_C)}{(n\ell_N + m\ell_C - \ell_N)} \tag{5.25}$$

or

$$((n\ell_N + m\ell_C)(q_N - q_C) + (q_C\ell_C - q_N\ell_N)) > 0 \tag{5.26}$$

then

$$\left(\frac{((q_N - q_C)(n\ell_N + m\ell_C)}{(q_C\ell_C - q_N\ell_N))}\right)^{-1} > q_h \tag{5.27}$$

which is Equation 5.17.

We can rewrite q_{h_0} as:

$$q_{h_0} = q_C\ell_C - q_N\ell_N/(n\ell_N(q_N - q_C) + m\ell_C(q_N - q_C) + (q_C\ell_C - q_N\ell_N)) \tag{5.28}$$

and since $0 < q_h < 1$, and from Equation 5.26, we find that if $q_h < q_{h_0}$ then:

$$q_{h_0} > 0 \tag{5.29}$$

and therefore:

$$q_C \ell_C > q_N \ell_N \tag{5.30}$$

□

Lemma 5.1 implies that securing the hypervisor can significantly reduce the *IC* risks imposed on the *CVMs* under the *X* allocation strategy. When q_h is small enough (e.g., $q_h < q_{h_0}$), the *IC* risks imposed on any of the *CVMs* are small as well, i.e., $q_N q_h \ell_C$. According to Equation 5.14, R_A's gain from attacking a *CVM* is larger than the gain from attacking an *NVM*. R_A consequently prefers to directly attack a *CVM*. Moreover, when q_h is large enough (e.g., $q_h > q_{h_0}$), the *IC* risks imposed on *CVMs* are maximized, and R_A benefits more from launching an *IC* attack on the *CVMs*.

Whereas Lemma 5.1 shows that the number of VMs has no effect on R_A's decision under the *X* allocation strategy, Lemma 5.2 states that R_A's preferences under the *S* allocation strategy not only depend on the value of q_h, but also depend on the number of *CVMs* and *NVMs* per server.

Lemma 5.2. • *R_A always and directly attacks a CVM under the S allocation strategy if all the conditions in any of the following cases hold:*

Case 1-1

$$(q_C \ell_C > q_N \ell_N) \tag{5.31}$$

$$m \geq n;\ \forall n > 0 \tag{5.32}$$

Case 1-2

$$q_C \ell_C > q_N \ell_N \tag{5.33}$$

$$m < n;\ \forall m \geq 1 \tag{5.34}$$

$$\frac{n-1}{m-1} < \frac{q_C \ell_C}{q_N \ell_N} \tag{5.35}$$

Case 1-3

$$q_C \ell_C > q_N \ell_N \tag{5.36}$$

$$\frac{n-1}{m-1} > \frac{q_C \ell_C}{q_N \ell_N};\ \forall\ m \geq 2,\ \forall\ \ n \geq 2 \tag{5.37}$$

Case 1-4

$$q_C \ell_C < q_N \ell_N \tag{5.38}$$

$$\frac{n-1}{m-1} < \frac{q_C \ell_C}{q_N \ell_N}; \forall m \geq 2, \forall n \geq 2 \tag{5.39}$$

$$q_h > q_{h_1} > 0 \tag{5.40}$$

- *R_A always and directly attacks an NVM under the S allocation strategy if all the conditions in any of the following cases hold:*

  Case 2-1

$$q_C \ell_C > q_N \ell_N \tag{5.41}$$

$$\frac{n-1}{m-1} > \frac{q_C \ell_C}{q_N \ell_N}; \forall m \geq 2, \forall n \geq 2 \tag{5.42}$$

$$1 > q_h > q_{h_1} > 0 \tag{5.43}$$

Case 2-2

$$q_N \ell_N > q_C \ell_C \tag{5.44}$$

$$n \geq m; \forall m > 0 \tag{5.45}$$

Case 2-3

$$q_C\ell_C < q_N\ell_N \tag{5.46}$$

$$\frac{n-1}{m-1} > \frac{q_C\ell_C}{q_N\ell_N}; \forall m \geq 2, \forall n \geq 2 \tag{5.47}$$

Case 2-4

$$q_C\ell_C < q_N\ell_N \tag{5.48}$$

$$\frac{n-1}{m-1} < \frac{q_C\ell_C}{q_N\ell_N}; \forall m \geq 2, \forall n \geq 2 \tag{5.49}$$

$$0 < q_h < q_{h_1} < 1 \tag{5.50}$$

where $(n \geq 2)$, $(m \geq 2)$, and q_{h_1} is defined in Equation 5.51.

$$q_{h_1} = \frac{(q_C\ell_C - q_N\ell_N)}{(n-1)q_N\ell_N - (m-1)q_C\ell_C} \tag{5.51}$$

The proof of Lemma 5.2 can be easily verified using Fig. 5.2. Lemma 5.2 lists all the cases when any of the *CVMs* or *NVMs* are the direct target of R_A's attack under the *S* allocation strategy. When there are no *IC* risks, R_A's preferences are determined by the value of q_h, the potential gain of attacking each VM type, and the number of *CVMs* and *NVMs* per server.

After understanding R_A's preferences under different cybersecurity requirements and allocation strategies, we can now identify R_V's *Equilibrium* strategies assuming a worst-case attack scenario, i.e., the VM allocation strategy that corresponds to an *NE* strategy profile.

Theorem 5.1. *If all conditions in any of the following cases hold, R_V's static Equilibrium strategy is the S allocation strategy in which one of the CVMs is a direct target for R_A,* *i.e., the allocation game model admits the strategy profile (C,S) as a pure NE strategy profile.*

Case 1

$$q_C\ell_C > q_N\ell_N \tag{5.52}$$

$$m \geq n;\ \forall\ n \geq 1 \tag{5.53}$$

$$0 < q_h < q_{h_0} \tag{5.54}$$

Case 2

$$q_C\ell_C > q_N\ell_N \tag{5.55}$$

$$m \leq n;\ \forall m \geq 1 \tag{5.56}$$

$$0 < q_h < q_{h_0} \tag{5.57}$$

$$\frac{n-1}{m-1} < \frac{q_C\ell_C}{q_N\ell_N} \tag{5.58}$$

Case 3

$$q_C\ell_C > q_N\ell_N \tag{5.59}$$

$$(n-1)/(m-1) > (q_C\ell_C)/(q_N\ell_N);\ \forall\ m \geq 2,\ \forall\ \ n \geq 2 \tag{5.60}$$

$$\frac{n-1}{m-1} > \frac{q_C\ell_C}{q_N\ell_N};\ \forall\ m \geq 1,\ \forall\ \ n \geq 1 \tag{5.61}$$

$$0 < q_h < q_{h_1} < q_{h_0} \tag{5.62}$$

Case 4

$$q_C\ell_C > q_N\ell_N \tag{5.63}$$

$$\frac{n-1}{m-1} > \frac{q_C\ell_C}{q_N\ell_N};\forall m \geq 2,\forall n \geq 2 \tag{5.64}$$

$$\frac{n}{m} < \frac{q_N\ell_C}{(q_C\ell_N};\ \forall\ m \geq 1,\ \forall\ \ n \geq 1 \tag{5.65}$$

$$0 < q_h < q_{h_0} < q_{h_1} \tag{5.66}$$

Proof. When Case 2-1 from Lemma 5.1 holds, and when Case 1-1, Case 1-2 with $q_{h_1} < q_{h_0}$, or Case 1-3 with $q_{h_1} > q_{h_0}$ from Lemma 5.2 hold, we get the conditions listed in Cases 1 to 4 from Theorem 5.1, respectively.

We use the Minimax method [180] to prove Theorem 5.1. R_A is trying to maximize his/her expected gain (e.g., G) after a successful attack on an NVM or CVM. On the other hand, R_V is trying to minimize his/her potential loss by choosing the best allocation policy (e.g., S or X) to mitigate R_A's expected gain $L = -G$.

R_A chooses to attack the VM that maximizes his/her gain, i.e., R_A seeks to maximize his/her minimum gain causing R_V's a WPC loss.

Let $min(G(N))$, and $min(G(C))$ represent R_A's minimum expected payoffs when successfully attacking an NVM and a CVM under both X and S policies. From Fig. 5.2, we can write:

$$min(G(N,X),G(N,S)) = G(N,S) \tag{5.67}$$

$$min(G(C,X),G(C,S)) = G(C,S) \tag{5.68}$$

R_A, hence, chooses to attack the VM with the maximum minimum payoff, and according to Lemma 5.2:

$$a = max(min(G(N,X),G(N,S)),min(G(C,X),G(C,S))) = G(C,S) \tag{5.69}$$

Therefore, R_A's prefers to attack a CVM under the allocation S, i.e., the strategy profile (C,S).

On the other hand, R_V seeks to minimize his/her worst maximum loss assuming that the attacker is capable of successfully attacking any VM in the cluster (e.g., minimize his/her maximum loss).

According to Lemma 5.1, Case 2-1, we can write:

$$max(G(N,X),G(C,X)) = G(C,X) \tag{5.70}$$

Based on Case 1-1, Case 1-2, Case 1-3 when $q_{h_1} < q_{h_0}$, and Case 1-3 when $q_{h_1} > q_{h_0}$ in Lemma 5.2, we can see that the following is true when Case 1, Case 2, Case 3, and Case 4 in Theorem 5.1 hold:

$$max(G(N,S),G(C,S)) = G(C,S) \tag{5.71}$$

R_V as a result chooses the allocation strategy with the minimum potential loss assuming that the attacker can launch a successful attack on the VM that maximizes his/her gain under the S and X allocation strategies. That is:

$$v = min(max(G(N,X),G(C,X)),max(G(N,S),G(C,S))) = G(C,S) \tag{5.72}$$

From Equation 5.69 and Equation 5.72, we see that $a = v$. Further, from [180], we find that the preferred strategies for both R_A and R_V are that for R_A to attack a CVM when R_V chooses the allocation strategy S if any of the cases in Theorem 5.1 holds. Using the Game theory terminology, we can say that the game admits the pure strategy profile (C,S) as a pure NE strategy profile when the conditions stated in any of the cases of Theorem 5.1 holds because R_V and R_A cannot increase their payoffs by unilateral deviation from the strategy profile (C,S) □

Theorem 5.2. *The CVMs are the targets of IC attacks under the X allocation strategy and the targets of direct attacks under the S allocation strategy. Moreover, the* Equilibrium *strategy of R_V is the S allocation (e.g., the game admits the strategy profile (C,S) as a pure NE strategy profile, If all the conditions in any of the following cases hold.)*

Case 1

$$q_C\ell_C > q_N\ell_N \tag{5.73}$$

$$m \geq n; \forall n \geq 1 \tag{5.74}$$

$$0 < q_{h_0} < q_h \tag{5.75}$$

Case 2

$$q_C\ell_C > q_N\ell_N \tag{5.76}$$

$$m \leq n; \forall m \geq 1 \tag{5.77}$$

$$\frac{n-1}{m-1} < \frac{q_C\ell_C}{q_N\ell_N}; \forall m \geq 2, \forall n \geq 2 \tag{5.78}$$

$$0 < q_{h_0} < q_h \tag{5.79}$$

Case 3

$$q_C\ell_C > q_N\ell_N \tag{5.80}$$

$$\frac{n-1}{m-1} > \frac{q_C\ell_C}{q_N\ell_N}; \forall m \geq 2, \forall n \geq 2 \tag{5.81}$$

$$\frac{n}{m} < \frac{q_N\ell_C}{(q_C\ell_N}; \forall m \geq 1, \forall n \geq 1 \tag{5.82}$$

$$0 < q_{h_0} < q_h << q_{h_1} \tag{5.83}$$

Case 4

$$q_C\ell_C < q_N\ell_N \tag{5.84}$$

$$\frac{n-1}{m-1} < \frac{q_C\ell_C}{q_N\ell_N}; \forall m \geq 2, \forall n \geq 2 \tag{5.85}$$

$$0 < q_h < q_{h_1} \tag{5.86}$$

Proof. When Case 2-1 from Lemma 5.1 and Case 2-1, Case 2-2, or Case 2-3 from Lemma 5.2 hold simultaneously, or when Case 2-2 from Lemma 5.1 and Case 2-4 from Lemma 5.2 hold simultaneously, we get the conditions listed in Cases 1 to 4 in Theorem 5.2, respectively.

To this end, we use the Minimax method [180] to prove Theorem 5.2. R_A is trying to maximize his/her expected gain by attacking a CVM or an NVM. R_V is trying to minimize his/her WPC loss, by choosing the S or X allocation policy.

Using the minima method [180], R_A considers the minimum gains after attacking an NVM and a CVM under the S and X policies. R_A next chooses to attack the VM with the maximum minimum gain. In other words, R_A seeks to maximize his/her minimum gain.

$$min(G(N,X),G(N,S)) = G(N,S) \tag{5.87}$$

$$min(G(C,X),G(C,S)) = G(C,S) \tag{5.88}$$

R_A thus chooses the strategy profile with the maximum gain from the strategies with the minimum payoffs at each row of the payoff matrix in Fig. 5.2, and according to Lemma 5.2:

$$a = max(min(G(N,X),G(N,S)),min(G(C,X),G(C,S))) = G(C,S) \tag{5.89}$$

Therefore, R_A's strategy profile with the maximin gain is (C,S).

On the other hand, R_V seeks to minimize his/her worst maximum loss at each column of the payoff matrix in Fig. 5.2, i.e, the minimax payoff. From Lemma 5.1, Case 2-2, we have:

$$max(G(N,X),G(C,X)) = G(N,X) \tag{5.90}$$

and from Case 1-1, Case 1-2, Case 1-3 and Case 1-4 in Lemma 5.2, we can see that in Case 2-1, Case 2-2, Case 2-3, and Case 2-4 in Theorem 5.2, respectively:

$$max(G(N,S),G(C,S)) = G(C,S) \tag{5.91}$$

R_V hence chooses the strategy profile with the minimum potential payoff out of the strategies with the maximum potential payoff at each column of the payoff matrix in Fig. 5.2. According to Lemma 5.1, we have:

$$G(C,X) < G(N,X) \tag{5.92}$$

but $G(C,X) > G(C,S)$, therefore:

$$v = min(max(G(N,X),G(C,X)),max(G(N,S),G(C,S))) = G(C,S) \tag{5.93}$$

From Equation 5.89 and Equation 5.93, we see that $a = v$ and according to [180], we find that the game admits the pure strategy profile (C,S) as an NE when the conditions stated in any of the cases stated in Theorem 5.1 hold. □

Theorem 5.1 and Theorem 5.2 identify all the conditions under which R_V always prefers the split allocation policy to minimize his/her WPC loss when R_A prefers to directly attack a CVM. Both Theorems also imply that when R_A always takes advantage of the IC risks exists in the X allocation to indirectly compromise $CVMs$, R_V's best response to minimize his/her WPC loss is to choose the allocation strategy S. On the other hand, Theorem 5.3 helps identify R_V's *Equilibrium* strategy when R_A always prefers to use direct attacks under either available allocation strategies.

Theorem 5.3. *If all the conditions in any of the following cases hold, the CVMs are always the primary targets of a direct attack under the X allocation strategy, and the NVMs are the primary target of a direct attack under the S allocation strategy. Moreover, the* Equilibrium *strategy is the S allocation strategy (e.g., the game admits the strategy profile* (NVM, S) *as a pure NE strategy profile.)*

$$q_C \ell_C > q_N \ell_N \tag{5.94}$$

$$\frac{n-1}{m-1} > \frac{q_C \ell_C}{q_N \ell_N}; \forall m \geq 2, \forall n \geq 2 \tag{5.95}$$

$$0 < q_{h_1} < q_h < q_{h_0} \tag{5.96}$$

$$\frac{n}{m} > \frac{q_N \ell_C}{q_C \ell_N}; \forall m \geq 1, \forall n \geq 1 \tag{5.97}$$

Proof. When Case 1-1 from Lemma 5.1 and Case 2-1 from Lemma 5.2 hold simultaneously, we get the conditions listed in Theorem 5.3, respectively.

We use the Minimax method [180] to prove Theorem 5.3. R_A is trying to maximize his/her expected gain by launching an attack on a CVM or on an NVM. On the contrary, R_V is trying to minimize his/her WPC loss, by choosing between the S and X allocation policies to mitigate R_A's expected gain $L = -G$.

R_A considers the possible strategies at each row of the payoff matrix in Fig. 5.2, determines the profiles with the worst payoffs if he/she plays that row, and chooses the profile at which his/her payoff is best, among those with the worst payoffs, i.e., R_A seeks to maximize his/her minimum payoffs.

$$min(G(N,X),G(N,S)) = G(N,S) \tag{5.98}$$

$$min(G(C,X),G(C,S)) = G(C,S) \tag{5.99}$$

R_A thus chooses the strategy profile with the maximum payoff from the strategies with the minimum payoffs at each row of the payoff matrix, illustrated in Fig. 5.2, and from Lemma 5.2, we have:

$$a = max(min(G(N,X),G(N,S)),min(G(C,X),G(C,S))) = G(C,S) \tag{5.100}$$

Therefore, R_A's strategy profile with the maximin payoff is (C,S).

On the other hand, R_V seeks to minimize his/her WPC loss at each column of the payoff matrix in Fig. 5.2, i.e, the minimax payoff.

According to Lemma 5.1, Case 1-1:

$$max(G(N,X),G(C,X)) = G(C,X) \tag{5.101}$$

According to Lemma 5.2, we can see:

$$max(G(N,S),G(C,S)) = G(N,S) \tag{5.102}$$

R_V then chooses the strategy profile with the minimum potential payoff out of the strategies with the maximum potential payoff at each column of the payoff matrix, illustrated in Fig. 5.2. From Case 1-1 in Lemma 5.1, we have:

$$G(C,X) > G(N,X) \tag{5.103}$$

but $G(N,X) > G(N,S)$, therefore:

$$v = min(G(C,X),G(N,S)) = G(N,S) \tag{5.104}$$

From Equation 5.100 and Equation 5.104, we see that $a = v$ and according to [180], we find that the game admits the pure strategy profile (N,S) as an NE when the conditions stated in Theorem 5.3 hold true. □

On the contrary to Theorem 5.2, Theorem 5.3 lists the conditions by which the IC risks, imposed on the $CVMs$, are minimized under the X allocation strategy.

Theorem 5.4. *If all the conditions in any of the following cases hold, then the $NVMs$ are always the primary target of a direct attack under both the X and S policies. Moreover, the* Equilibrium *strategy is the S allocation policy (e.g., the game admits the strategy profile (NVM,S) as a pure NE strategy profile.)*

Case 01:

$$q_C\ell_C > q_N\ell_N \tag{5.105}$$

$$\frac{n-1}{m-1} > \frac{q_C\ell_C}{q_N\ell_N}; \forall m \geq 2, \forall n \geq 2 \tag{5.106}$$

$$\frac{n}{m} > \frac{q_N\ell_C}{q_C\ell_N}; \forall m \geq 1, \forall n \geq 1 \tag{5.107}$$

$$0 < q_h < q_{h_1} < q_{h_0} \tag{5.108}$$

Case 02:

$$q_C\ell_C > q_N\ell_N \tag{5.109}$$

$$\frac{n-1}{m-1} > \frac{q_C\ell_C}{q_N\ell_N}; \forall m \geq 2, \forall n \geq 2 \tag{5.110}$$

$$\frac{n}{m} < \frac{q_N\ell_C}{q_C\ell_N}; \forall m \geq 1, \forall n \geq 1 \tag{5.111}$$

$$0 < q_h < q_{h_0} < q_{h_1} \tag{5.112}$$

Case 03:

$$q_C\ell_C < q_N\ell_N \tag{5.113}$$

$$m \geq n; \forall n \geq 1 \tag{5.114}$$

Case 04:

$$q_C \ell_C < q_N \ell_N \tag{5.115}$$

$$m > n; \forall \ n \geq 1 \tag{5.116}$$

Case 05:

$$\frac{n-1}{m-1} < \frac{q_C \ell_C}{q_N \ell_N}; \forall \ m \geq 2, \forall n \geq 2 \tag{5.117}$$

$$q_h < q_{h_1} \tag{5.118}$$

Proof. When Case 2-1 from Lemma 5.1 and Case 2-1, when $q_{h_1} < q_{h_0}$, or Case 2-2, when $q_{h_1} > q_{h_0}$ from Lemma 5.2 hold, or when Case 2-2 from Lemma 5.1 and Case 2-2, Case 2-3 or Case 2-4 from Lemma 5.2 hold simultaneously, we get the conditions listed in Case1 to Case 5 from Theorem 5.4, respectively.

We use the Minimax method [180] to proof Theorem 5.4. R_A is trying to maximize his/her expected payoff, G, after a successful attack on NVM or CVM, and R_V is trying to minimize his/her potential loss, by choosing between the S and X allocation policies to mitigate R_A's expected gain $L = -G$.

R_A considers the possible strategies at each row of the payoff matrix in Fig. 5.2, determines the profiles with the worst payoffs if he/she plays that row, and chooses the profile at which his/her payoff is best, among those with the worst payoffs, i.e., R_A seeks to maximize his/her minimum payoffs.

$$min(G(N,X), G(N,S)) = G(N,S) \tag{5.119}$$

$$min(G(C,X), G(C,S)) = G(C,S) \tag{5.120}$$

R_A chooses the strategy profile with the maximum payoff from the strategies with the minimum payoffs at each row of the payoff matrix, illustrated in Fig. 5.2, and from Lemma 5.2, we have:

$$a = max(G(N,S), G(C,S)) = G(N,S) \tag{5.121}$$

Therefore, R_A's strategy profile with the maximin gain is (N,S).

On the other hand, R_V seeks to minimize his/her WPC loss at each column of the payoff matrix, illustrated in Fig. 5.2, i.e, the minimax payoff. From Lemma 5.1, Case 2-2, we have:

$$max(G(N,X),G(C,X)) = G(N,X) \tag{5.122}$$

and from Case 1, when $q_{h_1} < q_{h_0}$, or from Case 2, 3, 4, or 5 in Theorem 5.4, we have:

$$max(G(N,S),G(C,S)) = G(N,S) \tag{5.123}$$

R_V selects the strategy profile with the minimum potential payoff out of the strategies with the maximum potential payoff at each column of the payoff matrix in Fig. 5.2:

$$v = min(G(N,X),G(N,S)) = G(N,S) \tag{5.124}$$

From Equation 5.121 and Equation 5.124, we see that $a = v$ and according to [180], we find that the game admits the pure strategy profile (N,S) as a pure NE strategy profile when the conditions stated in any of the cases in Theorem 5.4 hold. □

The reason why the S allocation strategy is always the $Equilibrium$ strategy of R_V is that the zero-sum game does not incorporate resource usage in its model. This results in the fact that the zero-sum game does not have a mixed NE strategy profile. In other words, the $Equilibrium$ allocation strategy of R_V is always the static VM allocation strategy S.

Theorem 5.5. *The zero-sum security game model, described in Fig. 5.2 does not accept a mixed NE strategy profile, i.e., there is no $Equilibrium$ dynamic VM allocation strategy.*

Proof. At the mixed NE strategy profile, both R_A and R_V are indifferent in choosing their strategies. For example, R_A must randomize in such a way that R_V does not care which VM is to be attacked, i.e., $G_X = G(N,X)) = G(S,X)$ and $G_S = G(N,S)) = G(S,S)$. Let

$(0 < \beta < 1)$ be the probability that R_A attacks an NVM, then:

$$\begin{aligned} G_X = \beta(q_N\ell_N + (n-1)q_h q_N \ell_N + m q_h q_N \ell_C) \\ +(1-\beta)(q_C\ell_C + (m-1)q_h q_C \ell_C + n q_h q_C \ell_N) \end{aligned} \tag{5.125}$$

$$G_S = \beta(q_N\ell_N + (n-1)q_h q_N \ell_N) + (1-\beta)(q_C\ell_C + (m-1)q_h q_C \ell_C) \tag{5.126}$$

In the context of attacker randomizing, in order to find β, we must equalize Equation 5.125 and Equation 5.126 and solve them. This eventually gives:

$$\beta = \frac{-nq_C\ell_N}{(mq_N\ell_C) - (nq_C\ell_N)} \tag{5.127}$$

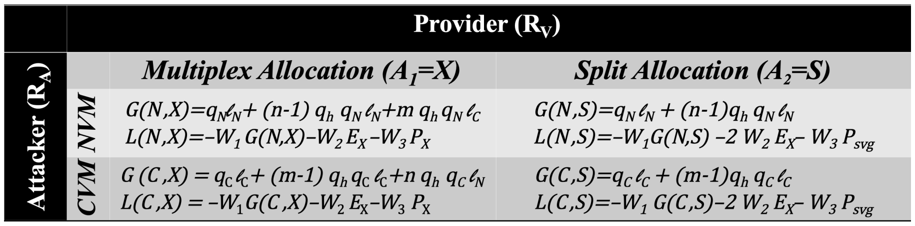
or

$$\beta = \frac{1}{(mq_N\ell_C)/(-nq_C\ell_N) + 1} \tag{5.128}$$

if $|(mq_N\ell_C)/(-nq_C\ell_N)| > 1$ then $\beta < 0$, and if $|(mq_N\ell_C)/(-nq_C\ell_N)| < 1$ then $\beta > 1$. But this contradicts the fact that β is a probability, i.e., $0 < \beta < 1$. Therefore, we conclude that there is no mixed NE strategy profile for the zero-sum security game. $\square$

We can so far draw several conclusions from our analysis to the zero-sum game model under no resource usage and power consumption constraints. First, the *Equilibrium* strategy that solves Problem 5.4 under different cybersecurity requirements, which are detailed in Theorem 1 to Theorem 4, is always the static VM allocation strategy S, under which there are no IC risks imposed on the $CVMs$. Second, the existence of the IC risks under the X allocation strategy primarily depends on the probability of indirectly compromising the hypervisor. Third, there are no *Equilibrium* dynamic allocation strategies. We show in Section 5.6 that the mixed NE strategy profiles are of great importance in optimizing R_V's loss and cost.

Although minimizing the potential cybersecurity loss is important, R_V also aims to optimize cloud resource usage and related expense to strive in the competing cloud market. In the next section, we extend our cybersecurity-aware zero-sum game model to include resource usage and power consumption constraints.



		Provider (R_V)	
		Multiplex Allocation (A_1=X)	***Split Allocation (A_2=S)***
Attacker (R_A)	***NVM***	$G(N,X)=q_N\ell_N+(n-1)\,q_h\,q_N\ell_N+m\,q_h\,q_N\ell_C$ $L(N,X)=-W_1\,G(N,X)-W_2\,E_X-W_3\,P_X$	$G(N,S)=q_N\ell_N+(n-1)q_h\,q_N\ell_N$ $L(N,S)=-W_1G(N,S)-2\,W_2\,E_X-W_3\,P_{svg}$
	CVM	$G(C,X)=q_C\ell_C+(m-1)\,q_h\,q_C\ell_C+n\,q_h\,q_C\ell_N$ $L(C,X)=-W_1G(C,X)-W_2\,E_X-W_3\,P_X$	$G(C,S)=q_C\ell_C+(m-1)q_h\,q_C\ell_C$ $L(C,S)=-W_1\,G(C,S)-2\,W_2\,E_X-W_3\,P_{svg}$

Figure 5.3: The non-zero-sum security game in normal form

5.6 Security-aware allocation with resource usage and power consumption constraints

In this section, we formulate our allocation problem using a non-zero-sum game model taking into consideration energy costs and operating expense. Fig 5.3 shows the game in its normal form. Whereas R_A's payoff matrix is the same as the payoff matrix described earlier in Fig. 5.2, R_V's payoff, in this case, is calculated using the utility function illustrated in Equation 5.129. We can see that Equation 5.129 consists of three sub-utility functions, L, P, and E), that are multiplied by three coefficients, w_1, w_2, and $(1-w_1-w_2)$, respectively. Those coefficients reflect the relative importance of the sub-utility functions to R_V:

$$L(VM,A) = -w_1 \cdot G(VM,A) - w_2 \cdot P(A) - (1-w_1-w_2) \cdot E(A) \tag{5.129}$$

where

$$VM \in \{N,C\} \tag{5.130}$$

$$A \in \{S,X\} \tag{5.131}$$

$$0 < w_1, w_2 < 1 \tag{5.132}$$

$L(VM,A)$ is R_V's loss after a successful attack on an NVM or a CVM and under the allocation policy $A = X$ or $A = S$. $P(T)$ and $E(T)$ are the cluster's total energy and operating cost under the allocation strategy $A = X$ or $A = S$, as described in Section 5.3.2.

R_A aims to attack the VM that maximizes his/her security gain. R_V, on the other hand, wishes to allocate the VMs so that the overall cybersecurity loss and cost is always at its minimum, i.e., multiplexing all VMs on the fewest number of servers while minimizing the WPC loss. However, collocating $CVMs$ with other $NVMs$ on a single server maximizes R_V's WPC loss. Therefore, we need to analyze the tradeoffs between cost savings optimization and cybersecurity loss minimization.

5.6.1 The VM-to-VM Interdependent Cybersecurity (IC) risk's bounds

In the following Theorem, we define the IC risk's bounds.

Theorem 5.6. *R_V's possible cost savings when selecting the S VM allocation strategy is constrained by the following upper and lower IC risk bounds.*

$$mq_hq_N\ell_C > Cost\ Savings > nq_hq_C\ell_N \tag{5.133}$$

Proof. Equation 5.12 and Equation 5.6 show that R_V can minimize his/her energy costs and operating expense by choosing the X strategy. Therefore, if R_A prefers a direct attack on a CVM, i.e., Lemma 5.1, a rational provider should always choose the X allocation strategy in which there are no IC risks are imposed on the $CVMs$. In the game model, this translates to $L(C,X) > L(C,S)$, which can be simplified to:

$$E + P_{svg} > nq_hq_C\ell_N \tag{5.134}$$

Equation 5.134 represents the IC risk's Lower Bound (LB). This bound quantifies the relationship between R_V's energy costs and operating expense and the IC risks that the $CVMs$ impose on the $NVMs$ under the X strategy. In other words, Equation 5.134 implies that R_V's minimum cost savings must always be larger than any IC risks imposed on the $NVMs$.

Moreover, R_V aims to minimize the IC risks imposed on any CVM by the $NVMs$. R_V, thus, should use extra resources to prevent R_A from indirectly attacking one of the $CVMs$. However, the question is –when is it justified, for R_V, to use extra resources to minimize his/her cybersecurity loss that is resulted from an IC attack, and how many extra resources are needed?– For example, R_V can eliminate the IC risks imposed on the $CVMs$ (e.g., the strategy profile (N,X)) by using the S strategy (e.g., switching to the strategy profile (N,S)), if R_V's overall loss and cost under the S strategy is less than the overall loss and cost under the X allocation strategy. That is, $L(A_C,X) < L(A_C,S)$, or:

$$mq_hq_N\ell_C > E_X + P_{svg} \tag{5.135}$$

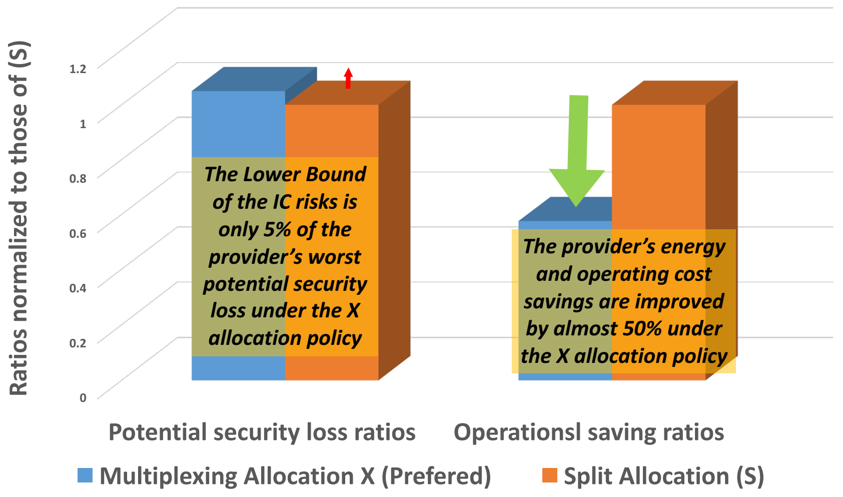
Equation 5.135 represents the Upper Bound (UB) of the IC risks. This bound quantifies the tradeoff between minimizing the IC risks that the $NVMs$ impose on the $CVMs$ under the X strategy and between optimization the cost savings by switching to the S strategy. This IC risks' limit sets the resource usage threshold over which R_V is not allowed to exceed when using the S allocation to minimize the IC risks. It is rational that when the cost of the extra resources used to mitigate the IC risks exceed the cybersecurity loss, R_V chooses the X strategy, (e.g., the strategy profile (C,X)) to minimize his/her overall potential loss and cost.

To this end, we can see that from Equation 5.134 and Equation 5.135 that R_V's feasible cost savings are bounded by the lower and upper bounds of the IC risks. We can merge the previous two inequalities into a single inequality (e.g., Equation 5.136), and call it the Cyber-constrained cost Savings Range (CSR):

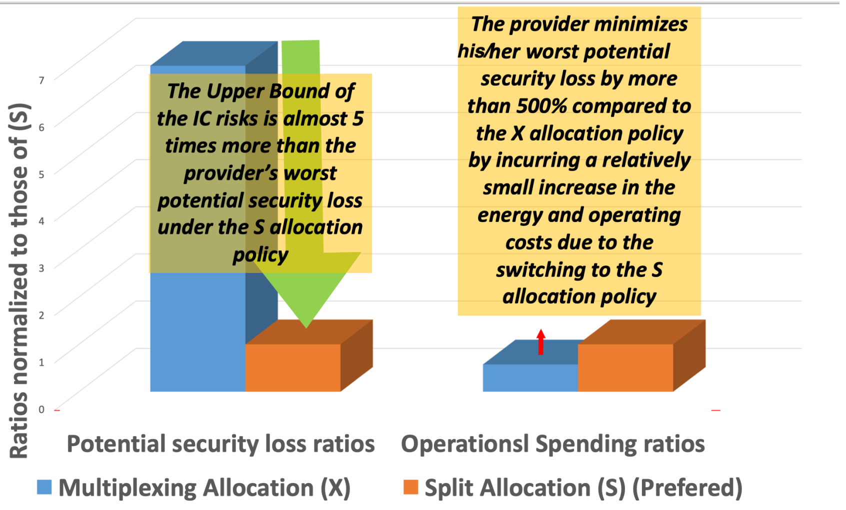
$$mq_hq_N\ell_C > E_X + P_{svg} > nq_hq_C\ell_N \tag{5.136}$$

□

Both IC risk's lower and upper bounds are illustrated in Fig. 5.4(a) and Fig. 5.4(b), respectively. We assume a single CMV and a single NVM (Other experiment setup and



(a) The Lower Bound (LB) of the IC risks



(b) The Upper Bound (UB) of the IC risks

Figure 5.4: Trade-off optimization between cybersecurity risks and cost savings.

values are detailed in Section 5.7). Fig. 5.4(a) shows that R_V can improve his/her cost savings by almost 50% while incurring less than a 5% increase in the WPC loss, due to the IC risks imposed on the $NVMs$, by multiplexing both types of VMs on the same server. On the other hand, when one of the $CVMs$ is the target of an IC attack, R_V can minimize his/her WPC loss by more than five times when allocating the $CVMs$ and $NVMs$ to separate servers, as illustrated in Fig. 5.4(b).

Next, we will identify R_V's *Equilibrium* strategies that solve Problem 5.4 under different attack scenarios.

Theorem 5.7. *If any of the cases in Theorem 5.1 hold, and if the LB inequality (e.g., Equation 5.134) holds, the X strategy is R_V's static $Equilibrium$ strategy (e.g., the non-zero-sum game admits the strategy profile (C,X) as a pure NE strategy profile). However, if the LB inequality does not hold, i.e.*

$$E_X + P_{svg} < q_h q_C \ell_N \tag{5.137}$$

then the S strategy is R_V's static $Equilibrium$ strategy (e.g., non-zero-sum game admits the strategy profile (C,S) as a pure NE strategy profile).

Proof. If any of the cases in Theorem 5.1 hold, then according to Lemmas (5.1) and (5.2) R_A prefers to directly attack a CVM when R_V chooses the allocation policy X or S.

On the other hand, if the lower bound inequality (e.g., Equation 5.134) holds, then $L(C,X) > L(C,S)$, and R_V consequently prefers the X strategy (e.g., the strategy profile (C,X)), the non-zero-sum game admits the pure strategy profile (C,X) as an NE. However, if Equation 5.137 holds, i.e., $L(C,X) < L(C,S)$, and the non-zero-sum game admits the pure strategy profile (C,S) as an NE because no player can increase his/her payoff by unilateral deviation from the strategy profile (C,S). $\square$

Theorem 5.7 reflects the important role of the LB inequality in optimizing R_V's cost savings performance. R_V can multiplex both types of VMs to save in his/her cost, while imposing no IC risks on the $CVMs$. Nonetheless, R_V allows $CVMs$ to impose minimal IC risks on $NVMs$. However, if the potential cyber security loss, assuming that the shared server is compromised, exceeds the total cost savings, R_V should switch to the S allocation. Although, the second scenario is almost impossible, as ℓ_N is negligible in front of ℓ_C, and q_C is very small, i.e., Equation 5.134. Next, we will use the UB to help identify R_V's $Equilibrium$ strategies using Theorem 5.8.

Theorem 5.8. *If any of the cases in Theorem 5.4 hold, and if Equation 5.135 holds, the S allocation strategy is R_V's static Equilibrium strategy in which the NVMs are the direct target of R_A (e.g., the non-zero-sum game admits the strategy profile (N,S) as a pure NE strategy profile). However, if any of the cases in Theorem 5.4 hold true, and if Equation 5.135 does not hold, i.e.*

$$E + P_{svg} > q_h q_N \ell_C \tag{5.138}$$

then the X allocation strategy is R_V's static Equilibrium strategy in which the CVMs are the direct target of R_A, (e.g., the non-zero-sum game admits the strategy profile (N,X) as a pure NE strategy profile).

Proof. If any of the cases in Theorem 5.4 hold, then according to Lemma 5.1 and Lemma 5.2, R_A always prefers to attack an NVM (e.g., both players prefer the profile strategies (N,X), and (N,S)).

On the other hand, if Equation 5.135 holds, i.e., $L(N,S) > L(N,X)$, R_V consequently prefers the strategy profile (N,S), and the non-zero-sum game admits the pure strategy profile (N,S) as an NE. However, if Equation 5.138 holds, i.e., $L(N,S) < L(N,X)$, the non-zero-sum game admits the pure strategy profile (N,X) as an NE because no player can increase his/her payoff by unilateral deviation from the strategy profile (N,X). □

Theorem 5.8 implies that R_V can judiciously acquire more resources to minimize the IC risks imposed on the $CVMs$ under the X strategy (e.g., the strategy profile (N,S) is a pure NE strategy profile). However, if the cost of the newly allocated resources exceeds the WPC loss under the X allocation, R_V should switch from S to X (e.g., the strategy profile (N,X) is a pure NE strategy profile).

So far, all R_V's *Equilibrium* allocation strategies that we can identify are static. In other words, all the provider's optimal allocation strategies correspond to pure NE strategy profiles. The following two theorems discuss the cases, according to which, the provider optimal allocation strategy is dynamic (e.g., the non-zero-sum game accepts mixed NE strategy profiles).

Theorem 5.9. *If all of the conditions stated in Theorem 5.2 hold and if Equation 5.136 holds, R_V's* Equilibrium *strategy is a dynamic allocation that involves both X and S strategies (e.g., the non-zero-sum game admits a mixed NE strategy profile). However, if Equation 5.136 does not hold and Equation 5.137 holds, the S strategy is R_V's static* Equilibrium *strategy in which the $CVMs$ are the target of a direct attack (e.g., non-zero-sum game admits the strategy profile (C,S) as a pure NE strategy profile). Moreover, if Equation 5.136 does not hold and if Equation 5.138 holds, the X strategy is R_V's static* Equilibrium *strategy in which the $NVMs$ are the target of a direct attack (e.g., the non-zero-sum game then admits the strategy profile (N,X) as a pure NE strategy profile).*

Proof. If all the conditions stated in Theorem 5.2 hold, according to Lemma 5.1 and Lemma 5.2, R_A and R_V prefers the strategy profiles (N,X) and (C,S).

On the other hand, if Equation 5.136 holds, R_V prefers the strategies (X), and (S) when R_A prefers to directly attack an NVM and a CVM, respectively. Thus, when R_V chooses the X strategy, R_A directly strikes a VM of type NVM (e.g., the strategy profile (N,X)). R_V then switches to the X allocation strategy to minimize the IC risks imposed on the $CVMs$ by the $NVMs$ (e.g., the strategy profile (N,S)). However, according to Theorem 5.3, R_A can now gain more by directly attacking a CVM (e.g., the strategy profile (C,S)). Further, R_V can save in resource usage by switching to the X strategy (e.g., the strategy profile (C,X)), and the game continues similarly in an infinite loop. In other words, there is no pure NE strategy profile for the game described in Fig. 5.3, because there is no strategy at which both players cannot make unilateral changes to improve their payoffs.

However, if Equation 5.134 and/or Equation 5.135 do not hold, and if Equation 5.134 and/or Equation 5.135 hold, R_V prefers the strategy profiles (C,X) and/or (N,S) because $L(C,X) > L(C,S)$ and/or $L(N,S) > L(N,X)$ when R_A chooses to directly attack a VM of types CVM and/or NVM, respectively. Consequently, the game admits the strategy profiles (N,X) and/or (C,S) as pure NE strategy profiles, respectively. $\square$

In the mixed NE strategy profile, described in Theorem 5.9, when R_V selects the S allocation, R_A gains more by directly attacking a CVM (e.g., the strategy profile (C,S)). However, Theorem 5.9 shows that R_V can optimize his/her payoff by switching to the X allocation (e.g., the strategy profile (C,X).) This decision is justified and quantified based on Equation 5.134. However, R_A can now utilize the IC risks that the $NVMs$ impose on the $CVMs$ by switching to the X strategy (e.g., the strategy profile (N,X)). Again, R_V can minimize the IC risks by spending extra resources and switching to the strategy S (e.g., the strategy profile (N,S)). This decision is also justified and quantified according to Equation 5.135. This scenario continues as players continue switching from one strategy to another.

Theorem 5.10. *If any of the cases stated in Theorem 5.3 holds and if Equation 5.137, and/or if Equation 5.138 hold, R_V's* Equilibrium *strategy is a dynamic allocation strategy that involves both the X and S strategies (e.g., the non-zero-sum game admits a mixed NE strategy profile). However, if Equation 5.136 holds, both the X and S are R_V's static* Equilibrium *strategies (e.g., the non-zero-sum game admits the strategy profiles (C,X) and (N,S) as pure NEs strategy profiles).*

Proof. If any of the cases stated in Theorem 5.3 hold, according to Lemma 5.1 and Lemma 5.2, R_A prefers to attack a CVM and an NVM (e.g., the strategy profiles (C,X) and (N,S)) when R_V chooses the X and S strategy, respectively.

On the other hand, if Equation 5.136 holds, according to Theorem 5.9, R_V prefers the strategies X and S (e.g, the strategy profiles (C,X) and (N,S)) when R_A prefers to directly attack an NVM and CVM, respectively. The non-zero-sum game, described in Fig. 5.3, thus admits the strategy profiles (C,X) and (N,S) as pure strategy NE strategy profiles.

Nonetheless, if the lower and upper bounds given in Equation 5.134 and Equation 5.135 do not hold, then R_V prefers the strategies S and X (e.g., strategy profiles (C,S) and (N,X)) since $L(C,X) < L(C,S)$ and $L(N,S) < L(N,X)$ when R_A chooses to directly attack $CVMs$ and $NVMs$, respectively. The game hence admits (N,X) as a mixed NE strategy profile and/or (C,S) as a pure NE strategy profile, respectively. $\square$

In the mixed NE strategy profile, discussed in Theorem 5.10, when R_V selects the S allocation, R_A gains more by directly attacking the $NVMs$ (e.g., the strategy profile (N,S)). This case is possible when the number of $NVMs$ is substantial, or the probability of directly attacking an NVM or the hypervisor is very high. Theorem 5.10 shows that R_V can optimize his/her payoff by switching to the X allocation (e.g., the strategy profile (N,X)). This decision is justified and quantified based on Equation 5.137. However, R_A can now exploit the IC risks that the $CVMs$ imposed on the $NVMs$ by switching to the strategy profile (C,X). Again, R_V minimizes the IC risks imposed on the $NVMs$ by switching to the profile (C,S), i.e., Equation 5.138. The scenario continues as players continue switching from one strategy to another.

When R_V and R_A randomize in such a way that the other player does not care which strategy the first player selects, i.e., $G(V_N) = G(V_C)$ and $L(N) = L(C)$, the game reaches a dynamic *Equilibrium* state, i.e., a mixed NE strategy profile. In Theorem 5.11, we show how to calculate the probabilities by which every player randomly selects many strategies resulting in uncertainty and unpredictability to the other player.

Whereas Theorem 5.10 and Theorem 5.9 identify the conditions at which R_V can use VM migrations to optimally minimize the overall WPC loss and costs, Theorem 5.11 calculates the percentage according to which R_V should wait at each allocation strategy before migrating the VMs and switching to another allocation strategy.

Theorem 5.11. *If all of the conditions stated in Theorem 5.2 hold, if Equation 5.136 holds, if R_V assigns a probability weight α to choosing the X strategy and a probability weight $(1-\alpha)$ to choosing the S strategy, and if R_A assigns a probability weight β to directly attacking an NVM and a probability weight $(1-\beta)$ to directly attacking a CVM, then:*

1. *The game admits the optimal dynamic* Equilibrium *state (e.g., a mixed NE strategy profile) in which a player's response to the other player is his/her best response.*
2. *R_V ensures that the IC risks imposed on the $CVMs$ by the $NVMs$ under the X strategy are minimized by $(1-\beta\%)$; where:*

$$\alpha = \frac{G(C,S)-G(N,S)}{G(N,S)+G(C,S)-G(N,S)-G(C,X)} \tag{5.139}$$

$$\beta = \frac{L(C,S)-L(C,X)}{L(N,X)+L(C,S)-L(N,S)-L(C,X)} \tag{5.140}$$

Proof. First, according to Nash, An *Equilibrium* point is an n-tuple so that a player's mixed strategy maximizes his/her payoff if the strategies of the others are held fixed. Thus, each player's strategy is optimal against those of the others, which induces that R_A is indifferent in choosing the target which in turn significantly affects the cluster security.

Suppose that R_V assigns probability weight α to choosing the X strategy and a probability weight $(1-\alpha)$ to the S strategy. R_A's expected gain, then, from attacking an NVM is:

$$G(N) = \alpha G(N,X) + (1-\alpha)G(N,S) \tag{5.141}$$

and R_A's expected gain from attacking a CVM:

$$G(C) = \alpha G(C,X) + (1-\alpha)G(C,S) \tag{5.142}$$

R_V wants to randomize so that R_A has no preference in which VM types to attack, i.e., $G(N) = G(C)$:

$$\alpha G(N,X) + (1-\alpha)G(N,S) = \alpha G(C,X) + (1-\alpha)G(C,S) \tag{5.143}$$

which can be simplified to Equation 5.139:

$$\alpha = \frac{G(C,S) - G(N,S)}{G(N,S) + G(C,S) - G(N,S) - G(C,X)} \tag{5.144}$$

Now suppose that R_A assigns a probability weight β to attack an NVM, and a probability weight $(1-\beta)$ to attack a CVM. U

sing a similar argument, we find that R_A must randomize in such a way that R_V does not prefer an allocation policy over the other, $L(X) = L(S)$:

$$\beta L(N,X) + (1-\beta)L(C,X) = \beta L(N,S) + (1-\beta)L(C,S) \tag{5.145}$$

which can be simplified to Equation 5.140:

$$\beta = L(C,S) - L(C,X)/L(N,X) + L(C,S) - L(N,S) - L(C,X) \tag{5.146}$$

We can easily observe from equations 5.139 and 5.140 that:

$$0 \leq \alpha, \beta \leq 1 \tag{5.147}$$

and according to [181] and [182] the game admits an optimal NE strategy profile.

Second, R_V's overall cybersecurity loss and cost expected from choosing the X strategy, assuming that R_A attacks an NVM and a CVM with probabilities β and $1-\beta$ is:

$$L(X) = \beta L(N,X) + (1-\beta)L(C,X) \tag{5.148}$$

We can see from Equation 5.148 that the IC risks imposed on the $CVMs$ by the $NVMs$ under the optimal mixed NE strategy profile are minimized by $1-\beta\%$, i.e., $\beta \times m \times q_h \times q_N \times \ell_C$ $\square$

Next, we discuss our experimental setup and present our numerical results.

5.7 Numerical results

We used Matlab to numerically analyze the existence of NE strategy profiles of the non-zero-sum game using different values of q_C, q_N, and q_h.

In each simulation, we varied a specific parameter and studied the effect of that parameter on R_V's WPC loss under the pure and mixed NE strategy profiles. Without loss of generality, we assumed the following values to illustrate some of the non-intuitive implications of findings: $\ell_C = 20$, $\ell_N = 5$, $q_C = 0.1$, $q_N = 0.6$, $q_h = 0.3$, $n = 3, m = 1$, $E_X = 0.5$, and $E_S = 1$. We used the Green Cloud Computing Prototype (GCCP) described in [170] to measure the processing rate utilization and power consumption under every allocation strategies, and we found that $P_S = 1$, and $P_X = 0.6514$ when the power consumption is normalized to the S.

The provider's payoff performance with the change in q_C

We varied q_C from 0.001 to $q_N - 0.001$ while measuring R_V's payoff, as shown in Fig. 5.5(a). We observe that R_V's payoff is maximized when q_C is very small. For instance, according to Theorem 5.3 and Case 03 of Lemma 5.2, when q_C is small (e.g., $q_C < 0.24$), we can see that $q_C\ell_C < q_N\ell_N$ and accordingly the game admits the strategy profile (N, S) as a pure NE strategy profile.

When $(0.24 < q_C < 0.3070)$ and according to Lemma 5.2, Case 03, R_A's payoff at the strategy profile (CVM, S) becomes larger than his/her payoff at the profile (NVM, S). But because the LB inequality holds true (Equation 5.134), R_V changes from the S to the X strategy. On the other hand, $q_{h_0} < q_h$ and R_A, thus, can gain more by directly attacking a CVM (e.g., the strategy profile (N, X)) according to Lemma 5.1. However, $0 < q_{h_0} < q_h < q_{h1} < 1$ and $n/m < (q_N\ell_C / q_C\ell_N)$. Therefore, according to Theorem 5.9,

R_V switch back again to the S allocation policy, and the players continue like this as the game admits a mixed NE strategy profile.

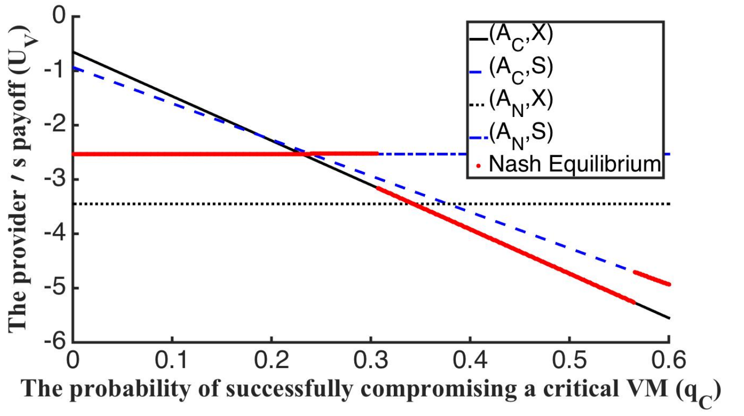
When $(0.3070 < q_C < 0.5660)$, the LB inequality (e.g., Equation 5.134) holds, and according to Theorem 5.8, the game admits the strategy profile (C,X) as a pure NE strategy profile since no player can unilaterally improve his/her payoffs by changing to another profile.

On the other hand, when q_C is relatively very large,$(0.5660 < q_C)$ and almost equal to q_N, the IC risks the CVM imposes on the $NVMs$ become larger than the cost savings achieved from choosing the X allocation strategy (e.g., the profile (C,X)), and R_V switches to the S strategy (e.g., the strategy profile (CVM,S)) i.e., the LB inequality does not hold anymore. Moreover, $G(C,S) = 4.8 \ < \ G(N,S) = 11.32$, and according to Theorem 5.11, the game admits the profile (C,S) as a pure NE strategy profile.

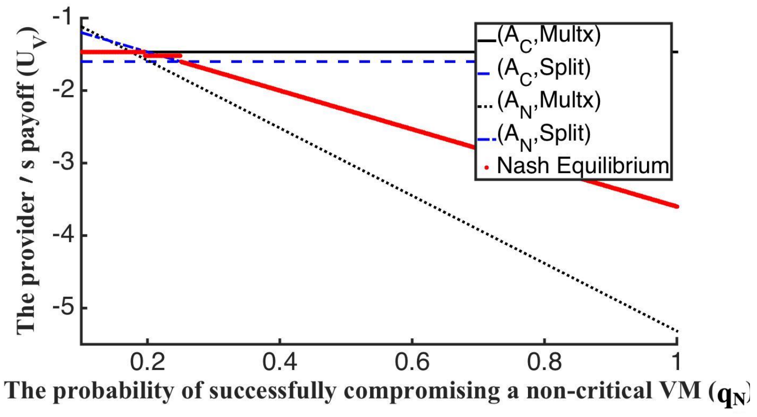
The provider's payoff performance with the change in q_N

We varied q_N from q_C to 1 and measured R_V's payoff accordingly, as shown in Fig. 5.5(b). We notice that when q_N is very small $(e.g., q_N < 0.1960)$, R_A prefers to attack a CVM under either allocation policy, i.e., Theorem 5.1. Furthermore, since the LB holds (e.g., Equation 5.134), the game admits the strategy profile (C,X) as a pure NE strategy profile, i.e., Theorem 5.9.

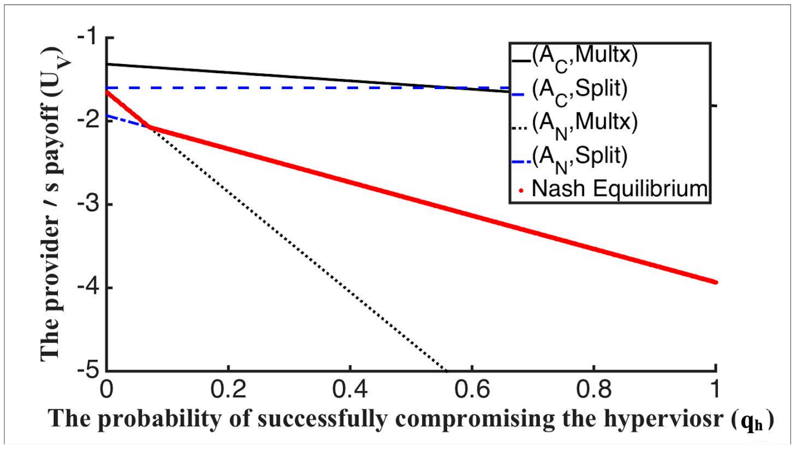
When $(0.1960 < q_N < 0.2510)$, we get $q_{h_0} = 0.2982 < q_h < q_{h1} = 0.5204$ and n/m $< (q_N\, \ell_C/q_C\, \ell_N))$, therefore, the CSR inequality holds (e.g., Equation 5.136,) and according to Theorem 5.9, the game admits a mixed NE strategy profile. As q_N increases (e.g., $q_N > 0.2510$), R_A prefers to directly attack an NVM under both strategies, i.e., Theorem 5.4. However, $q_h > q_{h1} = 0.2968$ and $n/m < (q_N\ell_C/q_C\ell_N) = 251$, and because the UB inequality holds, the game admits the strategy profile (C,S) as a pure NE strategy profile, i.e., Theorem 5.9.



(a) The provider's payoff performance with the change in q_C



(b) The provider's payoff performance with the change in q_N



(c) The provider's payoff performance with the change in q_h

Figure 5.5: The provider's payoff performance

The provider's performance with the change in q_h

We varied q_h from 0.001 to 1 and measured R_V's payoff accordingly, as shown in Fig. 5.5(c). When $0 < q_h < 0.0710$, the IC risks that the $NVMs$ impose on the CVM are minimized. Therefore, R_A always launches an IC attack on the CVM under either allocation policies, i.e., Theorem 5.4. On the other hand, the UB inequality does not hold, (e.g., Equation 5.138), as q_h is very small, and R_V consequently can minimize his/her energy costs and operating expense by choosing the X allocation strategy, and the game admits the strategy profile (C,X) as a pure NE strategy profile, i.e., Theorem 5.10.

As q_h increases, $0.0710 < q_h < 1$, the IC risks imposed on the CVM also increase, and the UB inequality holds. On the other hand, R_A's payoff from attacking the CVM under the S allocation strategy (e.g., $q_C \ell_C = 2$) is smaller than his/her payoff from attacking an NVM at the S strateg, i.e., $(q_N \ell_N + 2q_h q_N \ell_N) = 3.4260$, thus and according to Theorem (5.10), the game admits the strategy profile (C,S) as a pure NE strategy profile.

5.8 Summary

In this chapter, we analyze the VM-to-VM Interdependent Cybersecurity IC risks imposed on VMs of different criticality when they coexist together on the same hypervisor. We also investigate how to optimize the efficiency of resource usage and energy costs while minimizing the provider's WPC loss. Specifically, we study how to securely and efficiently allocate critical and Non-critical VM types (e.g., $CVMs$ and $NVMs$) onto a cloud cluster so that the provider's WPC loss is minimized. We formulate the allocation problem using non-cooperative zero-sum and non-zero-sum game models, between a cloud service provider and an attacker, under relaxed and constrained resource usage and power consumption. Our analysis completely characterizes the existence of all the static and dynamic *Equilibrium* strategies.

Next, we extend our cybersecurity models, presented in this chapter, to include VMs with any levels of cybersecurity and resource requirements.

CHAPTER 6
GAME THEORETIC-BASED APPROACHES FOR CYBERSECURITY-AWARE VIRTUAL MACHINE PLACEMENT IN CLOUD CLUSTERS

Allocating several Virtual Machines (VMs) onto a single server helps to increase cloud resource utilization and to reduce its operating expense. However, multiplexing VMs with different security levels onto a single server gives rise to significant VM-to-VM cybersecurity interdependent risks. In this chapter, we address the problem of the static VM allocation with cybersecurity loss awareness by modeling it as a two-player zero-sum game between an attacker and a cloud provider. We first obtain the optimal solutions by employing the mathematical programming approach. We then seek to find the optimal solutions by quickly identifying the equilibrium allocation strategies in our formulated zero-sum game. We mean by –equilibrium– that none of the provider nor the attacker has any incentive to deviate from one's chosen strategy. Specifically, we study the characteristics of the game model, based on which, to develop effective and efficient allocation algorithms. Simulation results show that our proposed cybersecurity-aware consolidation algorithms can significantly outperform the commonly used multi-dimensional bin packing approaches for large-scale cloud data centers.

6.1 Introduction to the research problem

Private data center managers should always configure servers to operate at high utilization rates. However, unwise workload allocation practices in private data centers result in low server utilization (e.g., 12% to 18%) and high power consumption per customer compared to (65%) cloud's server utilization and (84%) power consumption per cloud's consumer, as reported by the Data Center Efficiency Assessment from the Natural Resources Defense Council in 2014. Traditional data centers are duplicative, costly and complex [40] causing the annual electricity cost to increase by 2.5% each year over the last 20 years [41]. The scalable and elastic public cloud infrastructure maximizes resource usage and server utilization by adopting efficient workload multiplexing policies, such as VM consolidation. Therefore, the U.S. Office of Management and Budget (*OMB*) has issued guidance to reduce the number of federal, private data centers starting in 2010 as reported in [39]. The goal is to shift most of the critical and non-critical IT workload to public cloud data centers to save energy and operating costs.

The commonly shared infrastructure of public clouds among VMs and users, unfortunately, exposes cloud workload to several cybersecurity risks, such as side-channel attacks [51–53] and VM-to-VM Interdependent Cybersecurity (*IC*) risks. Packing several VMs on a single server minimizes the total number of allocated servers, which in turn minimizes power consumption and operating expense. For example, cloud clients require (77%) fewer servers than what they need on-premises [31], assuming (15%) on-premises utilization. However, when the consolidated VMs are of different security levels, the less secure VMs impose security risks on the more secure VMs. An attacker can compromise the hypervisor after a successful attack on one of its vulnerable VMs, and consequently, he/she could compromise all other coexisted secure VMs.

We intend to investigate the static allocation problem of VMs with different cybersecurity and resource requirements in cloud clusters. The dynamic VM allocation is flexible, adaptive and potentially more effective in a highly dynamic environment. On the other hand, static VM allocation has many unique advantages (e.g., low overhead, reliable predictability, performance guarantee) and thus the study of static VM allocation not only has solid theoretical values but is also useful and applicable in many practical scenarios. For instance, static allocation is ideal for batch workload allocation during each epoch, as they are usually well documented and characterized by predictable performance and intensive computation time.

We use the Game theory to formulate the static VM allocation problem into a non-cooperative two-player game [135]. Game theory is an appealing approach that offers mathematical models of conflict between non-cooperative intelligent and rational decision-makers, such as cloud providers and cyber attackers [135]. The game-based allocation model can help recognize the provider's static VM allocation strategies which optimally minimize his/her worst potential cybersecurity loss when an attacker can successfully compromise any VM in the cluster. We show later that the static VM allocation strategy is optimal if a cloud provider and an attacker reach an *Equilibrium* state [183]. We mean by –Equilibrium– that none of the provider nor the attacker has any incentive to deviate from one's chosen strategy.

We have made the following contributions in this chapter:

(1) A constrained-allocation problem is a typical NP-hard problem, and thus we formulate the security-constrained and resource-constrained VM allocation problems using the Mathematical Programming (*MP*) approach to obtain the optimal solutions, which will be used as a comparison baseline against other proposed methods in this chapter when the problem size is small;

(2) We formally formulate the resource and cybersecurity constrained VM allocation problems as non-cooperative two-player zero-sum game models. We conduct a thorough analysis of the characteristics of the pure Nash Equilibrium (*NE*) in our game model, which is formulated as a series of lemmas and theorems;

(3) Based on our analysis insights, we have developed several effective and computationally efficient algorithms to allocate VMs, of different resource and security requirements, with resource usage and cybersecurity loss optimized;

(4) We have implemented our algorithms and studied their effectiveness and efficiency. Our extensive simulation results show that our novel approaches are good trade-offs when compared with the computational-intensive approaches, such as *MP* based-

approaches and existing *NE* search methods, or with the computational-efficient multi-dimensional bin-packing methods.

6.2 Related work

As cloud online services entered in each sector of our personal and professional lives, maximizing resource usage and minimizing power consumption in cloud data centers using efficient allocation policies have become a necessity. Many allocation approaches, based either on traditional optimization methods (e.g., *MP* [126], evolutionary programming method [127], fuzzy control [128]) or on a variety of different heuristics (e.g., [129]), have been proposed for applications and VMs with different characteristics, requirements, and optimization goals in cloud platforms.

In this section, we review the research about VM allocation in cloud platforms. We can classify those related works on cloud computing into four significant categories directly related to our research focus in this dissertation, i.e., Vector Bin Packing (*VBP*) based cloud resource allocation, game theory-based cloud resource management, cloud cybersecurity measures using the Game theory, and security-aware cloud resource allocation approaches.

The VM allocation problem is a well-known NP-hard problem, as we show later on, and hence heuristics, such as Vector Bin Packing (*VBP*), have been adopted heavily to solve similar problems with multiple optimization goals [130]. In a *VBP* heuristic, there is a weight function applied to the items so that each item is assigned a single scalar, based on which, the standard bin packing can be used to sort those items. Wood et al. [92] introduced a multi-dimensional First Fit Decreasing (FFD) approach to lively migrate VMs out of overloaded servers. Their approach considers multiple CPU, memory, and network resource demands of each VM. Panigrahy et al. [131] proposed several *VBP*-based methods for VM allocation with different resource demands and optimization goals. Beloglazov et al. [126] suggested several VM migration algorithms to improve CPU utilization and demonstrated that increasing the lower-utilization threshold increases the Service Level Agreement (*SLA*) violations while reduces the server's energy consumption. In [128], the authors used combinatorial and multi-objective optimizations to optimize resource usage in a two-level control system that allocates workloads from virtual to physical resources. Using a local and a global fuzzy controller, they tried to minimize power consumption, thermal dissipation, and a peak temperature of the system. In [132], the authors introduced an application placement controller that consolidates applications according to the ratio of their CPU to memory demands. Microsoft's Virtual Machine Manager used in Azure applies the Dot-Product and Norm-based Greedy heuristics [133]. The authors in [133] proposed new geometric heuristics that run nearly as fast as *FFD*. We are, however, not aware of any prior work that employs *VBP* to deal with security requirements during a VM allocation process.

Several works employed the Game theory to model the resource optimization problem in cloud platforms [127, 129, 138]. Wei et al. [127] adopted the Game theory to solve a *QoS* constrained resource allocation problem across a cloud-based network. Kunsemoller et al. [129] elaborated on cloud economics benefits for businesses using a game-based cloud model of an *IaaS* economy including the dynamics of pricing and usage. Pillai et al. [139] proposed a VM allocation policy onto cloud platforms, based on the principles of coalition formation and the uncertainty principle of the Game theory. They illustrated that the coalition-formation of the VMs leads to higher resource utilization and higher request satisfaction. Teng et al. [140] suggested a new Bayesian pure *NE*-based resource management policy assumes heterogeneous and distributed resources, cognitive behaviors of cloud consumers, non-perfect information, and dynamic successive allocation. They proved that the resource price would converge to the optimal rate at the end of the gambling sequence. Jalaparti et al. [138] similarly employed the Game theory to optimize resource efficiency, and pricing policies by modeling the client-provider and client-client interactions, respectively. They introduced multiple heuristic algorithms with near-optimal allocation and pricing policies compared to the fixed-pricing strategies used today by cloud providers, such as in Amazon EC2. However, all the above ignored the cybersecurity effects of their allocation methods.

As more and more enterprises, companies, and private users move their computing facility to public cloud data centers, there have been increasing interests and concerns in the security problem in the cloud (e.g., [90, 141]). Several works focused only on studying the types of cybersecurity attacks result from the commonly shared infrastructure of public clouds among several applications and users.

Side-channel attacks are one of the most popular types of cybersecurity attacks on cloud infrastructure. Several countermeasures were proposed in the literature to mitigate or prevent side-channel attacks [51–53]. The proposed methods include modifying or tuning up the infrastructure to prevent hackers from extracting information about private keys (e.g., secret key extraction attack, [51]), preventing attackers from verifying co-residence with the victim's VM [52], or introducing a new infrastructure design (e.g., mitigate the threat of timing channels by eliminating high-resolution clocks [142], or adding latency to potentially malicious operations [52, 142]).

Kamhoua et al. [33] used a non-zero-sum game framework to model the VM-to-VM interdependent cybersecurity risks between an attacker and two users in the cloud. They showed that the existence of *NE* strategy profiles depends on the probability that the hypervisor is compromised given a successful attack on one of the users and the total cost of the user's security investments. They also irrationally concluded that there is no *NE* strategy profile in which all the users in the public cloud fully invest in cybersecurity measures. Et al. Kwiat [55] applied the same cybersecurity model introduced by Kamhoua to a different allocation problem. They considered a game between an attacker and three users. The first user never invests in security and is always allocated to the first insecure hypervisor. The second user invests in security countermeasures and his VM is always assigned to the second secure hypervisor.Unlike [33, 55], wherein the defenders' strategy is to invest or not invest, we assume that all VMs have different resource requirements and cybersecurity counter-measurements. The provider's strategy is to choose the allocation policy that minimizes his/her loss under a worst-case cybersecurity attack on a cloud cluster with a limited set of resources.

Secure-aware resource allocation methods were introduced in [90, 141, 143]. Rao et al. [143] used the Game theory to search the ability of a cloud computing provider to guarantee a given capacity C with a particular probability P given a physical or cybersecurity attack on his/her data center. They proposed the use of reinforcement strategies to decrease the attacker's utility. While many works focused on VM migration to maximize resource utilization and minimize power consumption, et al. Zhang [90] is the first to develop a formal and quantified migration strategy of clouds to improve security against collocation attack and with accepted costs.

On the other hand, [141] used the Game theory to model the collocation attack between attackers and a provider. The attackers try to collocate their VMs with target VMs on the same physical server and exploit side-channel attacks to extract private information from the victims' VMs. The provider aims to minimize the attackers' possibility of co-locating their VMs with the target VMs while maintaining a satisfactory workload balance and low power consumption for the system. Specifically, the provider strategies are to choose among a pool of allocation strategies.

In this chapter, we are more interested in studying the problem on how to optimize the provider's potential security loss or the provider's resource usage when allocating several VMs with different resource usage requirements and security vulnerabilities on a resource-constrained or security-constrained cloud cluster, respectively. In what follows, we first introduce our game model and formally define our problem. We then present our VM allocation approach in more detail.

6.3 Cloud system model

In this section, We formulate our cybersecurity-aware VM consolidation problems as non-cooperative zero-sum game models between an attacker and a provider [135]. A zero-sum game model is a game in which the players share no common interests [135]. We assume a game with perfect information. That is both players are rational, have full knowledge about the strategies of each other, and have the means to optimize their payoffs.

According to Nash [183], there is at least one best way, i.e., that corresponds to an *NE* strategy profile, by which both players simultaneously choose the best strategy for any limited number of strategies. The best policy for the provider, nonetheless, does not allow the provider to win but allows him/her to minimize his/her potential loss. However, there may not exist a pure *NE* strategy profile, and it is not always possible for players to find it in a polynomial time due to the large sizes of the pure strategy profiles of the provider and attacker.

A public cloud data center usually consists of hundreds or thousands of clusters, and we are interested in the static VM allocation policy for each cluster. We assume that a cloud cluster has n hypervisors that are hosted on the n physical servers $Sr = \{S_i; i = 1, 2, ..., n\}$. Each server has a maximum processing rate capacity $C_i = \mathbf{C}$ in Million Instructions Per Second (*MIPS*), a processing rate utilization U_i, power consumption P_i, and operating expense E_i (e.g., server costs, security investment, maintenance, etc.).

The cluster hosts m VMs (e.g., $M = \{V_j; j = 1, 2, ..., m\}$). A VM (e.g., V_j) is characterized by three parameters, i.e. ($V_j = \{c_j, q_j, \ell_j\}$); where ($c_j \leq \mathbf{C}$) is the processing rate

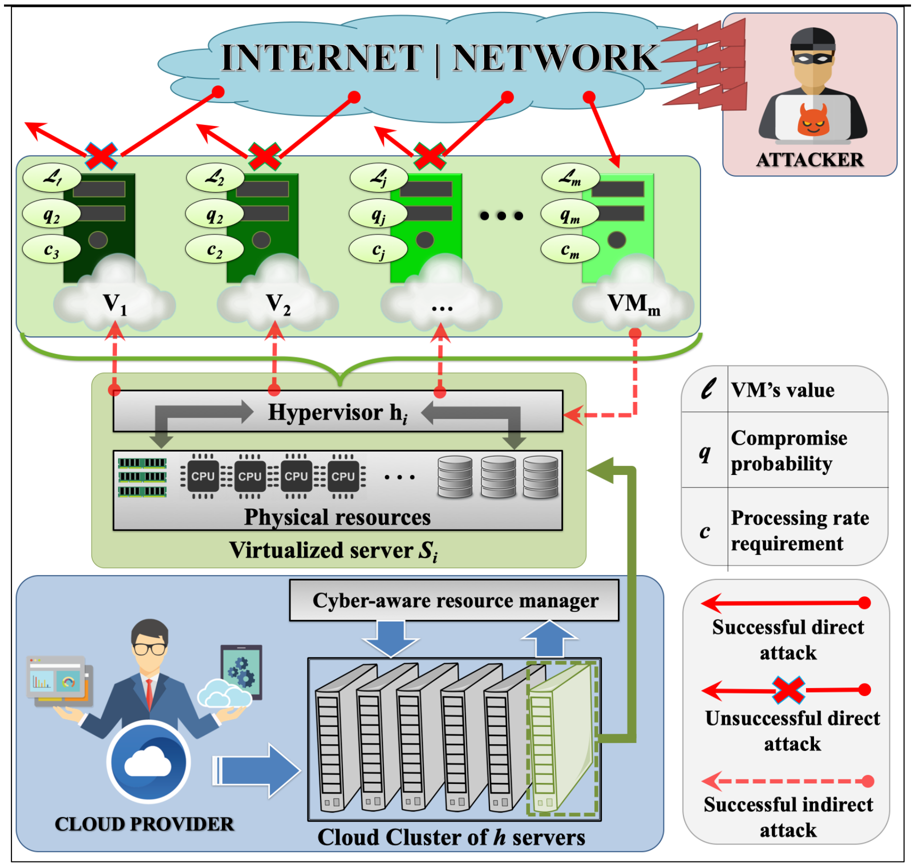


Figure 6.1: A cloud cluster. The cluster has n hypervisors that are hosted on the n physical servers $Sr = \{S_i; i = 1, 2, ..., n\}$. Each server has a maximum processing rate capacity $C_i = \mathbf{C}$ in Million Instructions Per Second ($MIPS$), a processing rate utilization U_i, power consumption P_i, and operating expense E_i (e.g., server costs, security investment, maintenance, etc.). The cluster hosts m VMs (e.g., $M = \{V_j; j = 1, 2, ..., m\}$). A VM (e.g., V_j) is characterized by three parameters, i.e. ($V_j = \{c_j, q_j, \ell_j\}$); where ($c_j \leq \mathbf{C}$) is the processing rate required by V_j; q_j is the probability that V_j could be directly attacked and compromised; ℓ_j is the provider's maximum security loss if V_j is compromised.

required by V_j; q_j is the probability that V_j could be directly attacked and compromised; ℓ_j is the provider's maximum security loss if V_j is compromised (e.g., Fig. 6.1) .

We further assume that $(\ell_j \geq \ell_k)$ if $(q_j \leq q_k)$ implying that the VM with the higher criticality level (e.g., V_j) has a smaller probability to be compromised. Moreover, we accept the fact that an attacker is capable of indirectly compromising a hypervisor, via one of its compromised VM (e.g., VM_m in Fig 6.1). Once an attacker has full control over the hypervisor, he/she can easily compromise all its VMs (e.g., $V_i; i = \{1, 2, ...m-1\}$ in Fig 6.1).

We denote the probability of a successful attack on a hypervisor after one of its VMs has been compromised as q_h. Furthermore, we assume that the possibility that an attacker successfully and directly attacks multiple VMs or a hypervisor is close to zero, and can thus be ignored.

Commonly sharing the infrastructure of a cloud data center, among applications, VMs, and users results in several cybersecurity risks, such as *IC* risks. For instance, an attacker can compromise the hypervisor after a successful attack on one of its vulnerable VMs. An attacker consequently can compromise all other coexisting secure VMs on that hypervisor. In this chapter, we are interested in identifying the VM static allocation strategy that can minimize the provider's Worst-case Potential Cybersecurity (WPC) loss, resulted from *IC* risks among coexisting VMs on a single server. The provider's potential loss under the z^{th} allocation strategy is the cybersecurity loss that the provider suffers if an attacker successfully and indirectly attacks a hypervisor, via one of its hosted VM that has been directly compromised earlier by the attacker, and thereafter compromises all other hosted VMs onto that hypervisor. If the attacker directly compromises the VM that maximizes his/her gain, after indirectly compromising its hypervisor and all its other hosted VMs, the provider's potential loss is then called the (WPC) loss under the z^{th} allocation strategy.

We can formally formulate the provider's WPC loss by first modeling the cybersecurity-aware VM allocation problem as a non-cooperative two-player zero-sum game defined by the 3-tuple (decision-makers, strategy profiles, payoffs) [135].

The decision-makers are a cloud service provider, R_V, and an attacker, R_A, who share no shared goals, i.e., a non-cooperative game model. R_A tries to maximize his/her gain by directly attacking one of the m VMs and potentially all other collocated VMs on the same server by attacking the underlying hypervisor [52,179]. R_V defends by trying to minimize his/her security loss by choosing an allocation strategy that reduces the attacker's gain.

R_A has the pure strategy spaces M from which R_A can choose to attack any VM (e.g., V_j) as a strategy to maximize his/her profit. On the other hand, R_V wishes to allocate all VMs onto the available servers to minimize his/her WPC loss. Let $\Gamma = \{A_z; z = 1,2,...,\zeta\}$ be R_V's set of all possible allocations of m VMs to n servers. Specifically, Γ is a $(1 \times \zeta)$ vector where each element, A_z, is an $(n \times m$ allocation matrix; i.e., $\{A_z = \{a^z_{ij}; i = 1,2,...,n; j = 1,2,...,m\}$; where:

$$a^z_{ij} = \begin{cases} 1;\ if\ V_j\ is\ allocated\ to\ S_i\ under\ A_z \\ 0;\ otherwise \end{cases} \tag{6.1}$$

Thus, Γ represents R_V's pure strategy space. In this chapter, we focus on static VM allocation strategies, therefore we assume that R_V and R_A can only choose one allocation strategy from their strategy spaces.

Choosing different strategies leads to different payoffs for R_V and R_A. Assume that R_V chooses the strategy A_z in which V_j is allocated to S_i (e.g., $(a^z_{ij} = 1)$), and assume that R_A chooses to attack V_j directly. We denote the corresponding R_V's potential cybersecurity loss or R_A's potential cybersecurity gain by $(l(V_j, A_z) = l_{jz})$ or $(g(V_j, A_z) = g_{jz})$, respectively; where $(l_{jz} = -g_{jz})$ as this is a zero-sum game. Then we have:

$$g_{jz} = a^z_{ij}(\Sigma^m_{k=1}(a^z_{ik} q_h q_j \ell_k) + q_j(\ell_j - q_h \ell_j)) \tag{6.2}$$

where m is the total number of VMs; $a_{ij}^z = 1$. We can consequently model R_V's and R_A's potential cybersecurity potential losses and gains as $(m \times \zeta)$ payoff matrices L and G, in which each column represents a single R_V's allocation strategy (e.g., A_z) and each row represents a single R_A's attack strategy (e.g., V_j). That is:

$$L = \{l_{jz} : j = \{1, 2, ..., m\}; z = \{1, 2, ..., \zeta\}\};\ L = -G \tag{6.3}$$

Specifically, we denote $L^*{}_z$ as the WPC loss when R_V chooses strategy A_z; where:

$$L_z^* = -G_z^* = -\max_{\forall j \in \{1,2,...,m\}} g_{jz} \tag{6.4}$$

Our zero-sum allocation game thereupon is defined by the 3-tuple $(\{R_A, R_V\}, \{M, \Gamma\}, \{G, L\})$.

Let Eop be a $(1 \times \zeta)$ vector representing R_V's total operating expense under the allocation strategies $A_z; z = 1, 2, ..., \zeta$. Operating expense includes the cost of renting or purchasing servers, security investment, hardware/software upgrades, maintenance, etc. For example, the operating expense of a cloud cluster under allocation A_Z is given by Equation 6.5.

$$Eop_z = \Sigma_{i=1}^n e_i^z \varepsilon_i \tag{6.5}$$

where ε_i is the operating expense of the non-idle server S_i under allocation A_z. That is:

$$e_i^z = \begin{cases} 1;\ if\ \Sigma_{j=1}^m a_{ij}^z \geq 1 \\ 0;\ otherwise \end{cases} \tag{6.6}$$

Let $Eeng$ be a $(1 \times \zeta)$ vector represents the cluster's total energy cost under the allocation strategies $A_z; z = 1, 2, ..., \zeta$ when the cluster consumes a total, static and dynamic, power P_z. Equation 6.7 models the total power WATT consumed by a cloud cluster under allocation A_Z.

$$P_z = \Sigma_{i=1}^n (e_i^z DP_i(\Sigma_{j=1}^m ((a_{ij}^z c_j)/\mathbf{C})) + e_i^z SP_i) \tag{6.7}$$

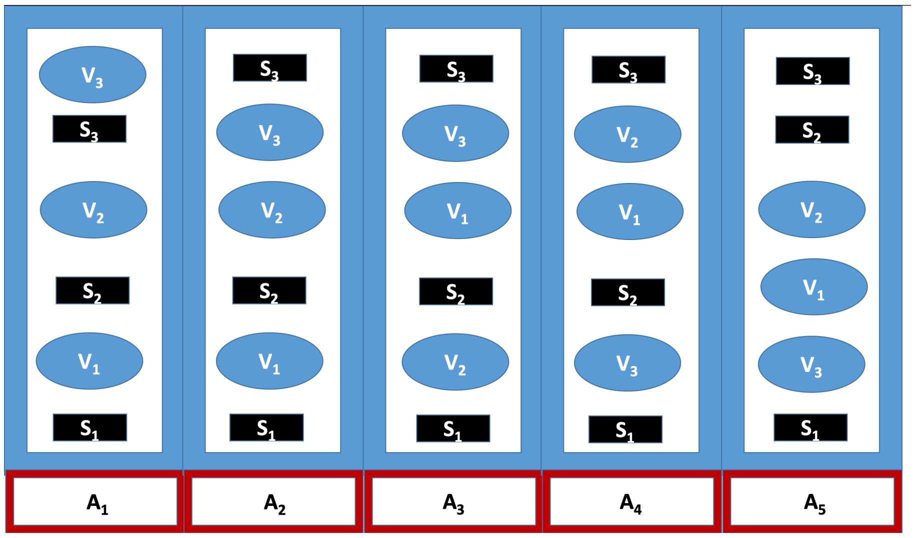


Figure 6.2: Illustration of the different ways of consolidating three VMs onto a three-server cloud cluster

where DP_i is the maximum dynamic power consumed by the server S_i when it is fully utilized; c_j is the processing rate requirements of V_j measured by $MIPS$; $\mathbf{C}$ is the maximum processing rate capacity of server S_i measured by $MIPS$. SP_i is the total static power consumed by server S_i.

Therefore, the $(1 \times \zeta)$ vector TC represents the *Total energy Cost and operating expense (TC)* of R_V under an allocation strategy A_z. That is:

$$TC_z = Eop_z + Eeng_z \tag{6.8}$$

On the other hand, R_V's *Overall WPC Loss and total energy Costs and operating expense (OLC)* under an allocation strategy A_z is represented by the $(1 \times \zeta)$ vector OLC.

R_V's overall loss and costs under an allocation A_z is then modeled using Equation 6.9.

$$OLC_z = TC_z + L_z^* \tag{6.9}$$

Fig. 6.2 shows that when a three-server cloud cluster hosts three VMs (e.g., $\{V_1, V_2, V_3\}$), R_V has five possible static VM consolidation strategies to allocate those VMs to the three servers $\{S_1, S_2, S_3\}$. In the first allocation strategy, R_V allocates one VM per server.

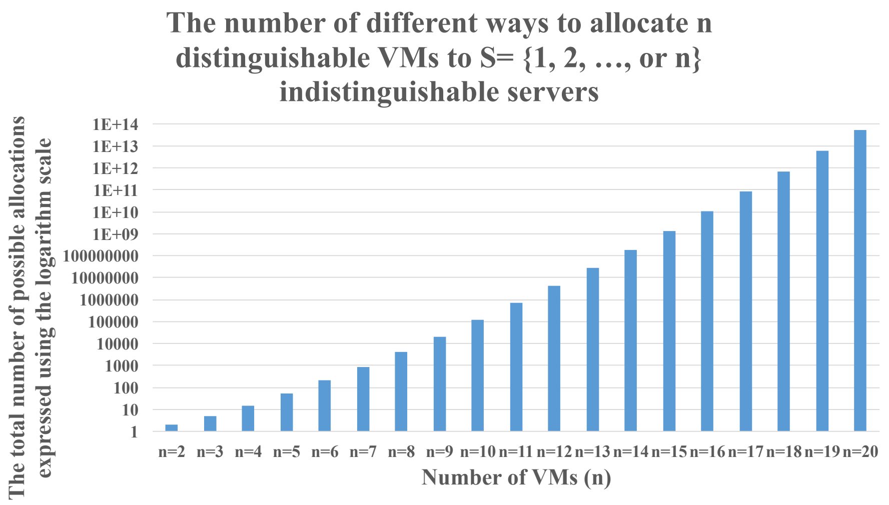


Figure 6.3: The size of the provider's pure strategy support vector with the increase of the numbers of VMs and servers

In the second, third, and forth allocation strategies, R_V consolidates two VMs on a single server and allocates the third VM to a second server. In the fifth allocation strategy, R_V multiplexes all three VMS on a single server S_1.

The first strategy allocates one VM per server which eliminates any IC risks, but it worsens the server's utilization and operating expense as it involves three servers rather than one server as opposed to the fifth strategy. Fig. 6.3 enumerates the different ways of allocating $m \leq 20$ different VMs to $n \leq 20$ different servers.

Whereas the size of R_A's pure strategy space is bounded by the total number of VMs (e.g., m), the size of R_V's strategy space, i.e. the number of columns in the payoff matrices L and G, is a function rapidly increasing with the increase of the numbers of VMs and available servers. To this end, Fig. 6.3 shows how the total number of different ways of allocating m VMs to n servers exponentially increases with the increases of n and m.

Assuming all allocation strategies are feasible, the total number of R_V's possible allocation strategies can be formulated as:

$$\zeta = \Sigma_{i=1}^{n} STN(m,i) \tag{6.10}$$

where $STN(m,i)$ is the Stirling numbers of 2^{nd} kind representing the total number of different ways to allocate m distinguishable VMs to exactly $(i \leq n)$ non-idle servers [184]. That is:

$$STN(m,i) = (\frac{1}{i!})\Sigma_{i_0=0}^{i}(-1)^{i_0}\binom{i_0}{i}(i-i_0)^m \tag{6.11}$$

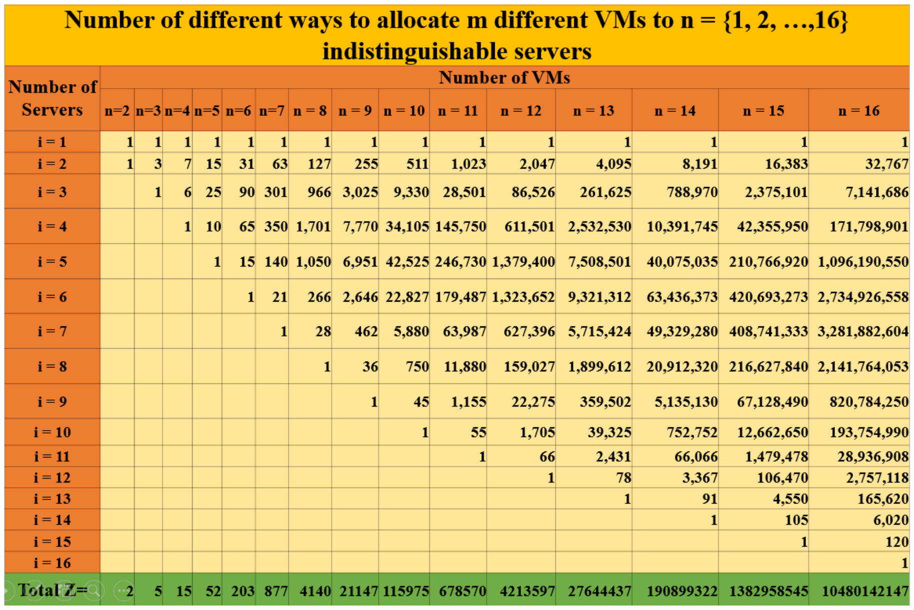
Consequently, to enumerate all possible ways of allocating the m VMs to n servers, we can rewrite Equation 6.11 as the recurrence equality given in Equation 6.12 [184]

$$STN(m,i) = STN(m-1,i-1) + i \times STN(m-1,i) \tag{6.12}$$

For example, $STN(1,1) = 1$ because there is just one way to allocate a single VM to a single server. We suppose that we know the number of ways to allocate $(m-1)$ VMs to $(i \leq n)$ identical servers and that we know the number of ways to allocate $(m-1)$ VMs onto $(i-1)$ identical servers. To this end, if we want to allocate one more VM and use i servers. We can either start with any of the $STN(m-1,i-1)$ combinations and allocate the m^{th} VMs to a new empty server, or we can start with any of the $STN(m-1,i)$ combinations and allocate the m^{th} VM to a non-empty server. In other words, for each of the $STN(m-1,i-1)$ combinations, there is just one way to add the new VM to a new server. However, for each of the $STN(m-1,i)$ combinations, there are i ways to allocate the new VM to a non-empty server. Thus the number of different ways to allocate m VMs to n servers (e.g., $STN(m,i)$) is the sum of $STN(m-1,i-1)$ and $i \times STN(m-1,i)$.

Fig. 6.4 shows the number of ways of allocating $m = \{1,2,...,16\}$ labeled VMs to $\{n = 1,2,...,or16\}$ identical empty or non-empty servers. It is clear that as the number of VMs increases, the total number of R_V's possible allocation strategies exponentially increases. For instance, when the number of VMs is $(m = 16)$, there are more than (10) billion possible allocation strategies when using $n = 16$ servers (e.g., Fig. 6.4).

Based on our system models, we can formulate our research problems as follows.



Number of different ways to allocate m different VMs to n = {1, 2, …,16} indistinguishable servers															
Number of Servers	Number of VMs														
	n=2	n=3	n=4	n=5	n=6	n=7	n = 8	n = 9	n = 10	n = 11	n = 12	n = 13	n = 14	n = 15	n = 16
i = 1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
i = 2	1	3	7	15	31	63	127	255	511	1,023	2,047	4,095	8,191	16,383	32,767
i = 3		1	6	25	90	301	966	3,025	9,330	28,501	86,526	261,625	788,970	2,375,101	7,141,686
i = 4			1	10	65	350	1,701	7,770	34,105	145,750	611,501	2,532,530	10,391,745	42,355,950	171,798,901
i = 5				1	15	140	1,050	6,951	42,525	246,730	1,379,400	7,508,501	40,075,035	210,766,920	1,096,190,550
i = 6					1	21	266	2,646	22,827	179,487	1,323,652	9,321,312	63,436,373	420,693,273	2,734,926,558
i = 7						1	28	462	5,880	63,987	627,396	5,715,424	49,329,280	408,741,333	3,281,882,604
i = 8							1	36	750	11,880	159,027	1,899,612	20,912,320	216,627,840	2,141,764,053
i = 9								1	45	1,155	22,275	359,502	5,135,130	67,128,490	820,784,250
i = 10									1	55	1,705	39,325	752,752	12,662,650	193,754,990
i = 11										1	66	2,431	66,066	1,479,478	28,936,908
i = 12											1	78	3,367	106,470	2,757,118
i = 13												1	91	4,550	165,620
i = 14													1	105	6,020
i = 15														1	120
i = 16															1
Total Z=	2	5	15	52	203	877	4140	21147	115975	678570	4213597	27644437	190899322	1382958545	10480142147

Figure 6.4: Enumerating the different ways of allocating $m = \{1, 2, ..., or 16\}$ different VMs to $n = \{1, 2, ...,$ *or* $16\}$ servers

Problem 6.1. *Given the virtualized cloud cluster with n servers and m VMs that is defined above, determine the static feasible allocation policy so that R_V's WPC loss (e.g., L^*) is minimized;*

Problem 6.2. *Given the virtualized cloud cluster with n servers and m VMs that is defined above, and given R_V's WPC loss threshold (e.g., $Thrsh$), determine the static feasible allocation policy so that R_V's overall loss and total energy cost and operating expense (e.g., OLC) is optimized while R_V's WPC loss is guaranteed to be less than the given threshold, i.e. $L^* \leq Thrsh$.*

In what follows, we first identify the optimal solutions for our VM allocation problems using the *MP* approach for comparison purposes, when R_V's pure strategy space is small. Second, we present our approaches by taking advantage of the game-theoretical techniques to solve problems with much larger sizes as in practical scenarios.

```
   Minimize $G^*$, subject to:
1: $G^* \geq G^*_j; \forall\ j \in \{1,2,...,m\}$
2: $G^*_j \geq g_{ij}; \forall\ i\ \in \{1,2,...,n\}$
3: $g_{ij} \geq d_{ij}((q_h q_j([d_{i1},d_{i2},...,d_{im}][\ell_1,\ell_2,...,\ell_m]^T)) + q_j(\ell_j - q_h \ell_j));$
   $\forall\ j \in \{1,2,...,m\}; \forall\ i \in \{1,2,...,n\}$
4: $\Sigma^n_{i=1} d_{ij} = 1; \forall\ j \in \{1,2,...,m\}$
5: $0 \leq ([d_{i1},d_{i2},...,d_{im}][c_1,c_2,...,c_m]^T) \leq \mathbf{C}; \forall\ i\ \in \{1,2,...,n\}$
```

Figure 6.5: The MPL algorithm

6.4 The mathematical programming formulation of the VM allocation problems

Fig. 6.5 describes the *Mathematical Program for Loss minimization (MPL)* to solve Problem 1 in which the decision variables are defined as follows:

$$d_{ij} = \begin{cases} 1;\ if\ V_j\ is\ allocated\ to\ S_i \\ 0;\ otherwise \end{cases} \tag{6.13}$$

For example, the first allocation strategy in Fig. 6.2 is represented by the decision matrix:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

in which V_1, V_2, and V_3 are allocated to S_1, S_2, and S_3, respectively.

MPL minimizes R_V's worst potential security loss by finding the allocation strategy that minimizes R_A's best potential gain G^* (e.g., Equation 6.4). The attacker's best gain is larger than or equal to the gain of attacking any V_j (e.g., Fig. 6.5, Line (1)) when it is allocated to any server S_i (e.g., Fig. 6.5, Line (2)). The potential security gain of R_A when attacking V_j on S_i is g_{ij} (Line (3)). Further, a V_j can be allocated to no more than a single server (e.g., Fig. 6.5, Line (4)), and the total processing rate requirements of all VMs allocated to a server S_i cannot exceed the server's capacity (Fig. 6.5, Line (5)).

According to the MPL program, the optimal solution to Problem 6.1 which minimizes R_V's worst potential security loss given a worst case cybersecurity attack is the first allocation strategy in Fig. 6.2:

$$A_{sv}(1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For example, let the virtual machines V_1, V_2, V_3, illustrated in Fig. 6.2, have a security-level's vector $\{0.4, 0.4, 0.9\}$, a loss-value's vector $\{-99.63, -66.198, -36.685\} \times \$10^3$, and resource-requirement's vector $\{1, 4, 3\} \times 10^3$ $MIPS$, respectively. If R_V has three servers S_1, S_2, S_3 each of which with the capacity $\mathbf{C} = 8 \times \$10^3 MIPS$, he/she can allocate the VMs using the five allocation strategies given in Fig. 6.2. To this end, in order to understand the solution of the mathematical program illustrated above, let's construct R_V's loss payoff matrix L that represents R_V's potential cybersecurity loss given a successful attack on V_i under the allocation strategy A_z.

$$L = \begin{bmatrix} V_j/A_i & A_1 & A_2 & A_3 & A_5 & A_4 \\ V_1 & 39.852 & 39.852 & 44.254 & 47.795 & 52.198 \\ V_2 & 26.479 & 30.881 & 26.479 & 38.434 & 42.837 \\ V_3 & 33.016 & 50.890 & 59.916 & 33.016 & 77.790 \end{bmatrix} \times \$ - 10^3$$

R_V's worst losses under each allocation A_z are then:

$$L_z^* = \begin{bmatrix} V_j/A_i & A_1 & A_2 & A_3 & A_5 & A_4 \\ V_1 & 39.8520 & \square & \square & 47.7958 & \square \\ V_2 & \square & \square & \square & \square & \square \\ V_3 & \square & 50.8900 & 59.9166 & \square & 77.7901 \end{bmatrix} \times \$ - 10^3$$

Consequently, the optimal WPC is equal to $L_1^* = -39.8520$. In other words, the best option for R_V is to choose the first allocation policy in which he/she would optimally minimize his/her WPC loss.

```
   Minimize $-1 \times OLC^*$, subject to:
1: $OLC^* \geq -1 \times \Sigma_{i=1}^{n}(Ceil(\frac{\Sigma_{j=1}^{m}(d_{ij})}{n}) \times (DP_i(\Sigma_{j=1}^{m}((d_{ij}c_j)/\mathbf{C})) + SP_i) \times \varepsilon_i)$
2: $G_z^* \leq Thrsh$
3: $G_z^* \geq G_j^*; \forall\ j \in \{1,2,...,m\}$
4: $G_j^* \geq g_{ij}; \forall\ i\ \in \{1,2,...,n\}$
5: $g_{ij} \geq d_{ij}((q_h q_j([d_{i1},d_{i2},...,d_{im}][\ell_1,\ell_2,...,\ell_m]^T)) + q_j(\ell_j - q_h\ell_j));$
   $\forall\ j \in \{1,2,...,m\}; \forall\ i \in \{1,2,...,n\}$
6: $\Sigma_{i=1}^{n} d_{ij} = 1; \forall\ j \in \{1,2,...,m\}$
7: $0 \leq ([d_{i1},d_{i2},...,d_{im}][c_1,c_2,...,c_m]^T) \leq \mathbf{C}; \forall\ i\ \in \{1,2,...,n\}$
```

Figure 6.6: The MPR algorithm

Similarly, Fig. 6.6 describes the *Mathematical Program for Resource usage maximization (MPR)* to solve Problem 2. *MPR* minimizes R_V's total energy costs and operating expense (e.g., Fig. 6.6, lines (1 and 2)), while keeping R_V's WPC less than a given loss threshold (e.g., Fig. 6.6, lines (2-7)).

Using the same previous example with the three VMs, we see that the average resource utilization under all different five possible allocation policies illustrated in Fig. 6.2 are: $U = \{\frac{1}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, 1\}$. Moreover, if we set the cybersecurity loss threshold to $Thrsh = 80 \times 10^3$, the optimal solution of Problem 2 is the fifth allocation strategy in Fig. 6.2 with an average resource utilization that is equal to 1.

Although both *MPL* and *MPR* solve Problem 1 and Problem 2, both mathematical programs have their limitations. The computation times of both algorithms are exponential in the increasing number of VMs. For example, in Section 6.8.2, we show that the average computation time of both *MPL* and *MPR* using 25 servers and 50 VMs increases by roughly 8000 times when increasing the number of servers and VMs to 250 servers and 500 VMs. Therefore, in the next section, we explore more computationally efficient methods to solve our allocation problems because such *MP* methods are not applicable in today's cloud data centers in which a single rack can have up to 1000 VMs [185].

6.5 Finding the optimal VM allocation strategy using Nash Equilibrium (NE) general search algorithms

In this chapter, we explain how to use *NE* general search methods to solve our allocation problems, after discussing several limitations of those methods.

Researchers use the Game theory to study and understand the interaction among economic, social, or military entities. We use Game theory in this chapter to analyze and understand the interaction between R_V and R_A. The goal is to find a stable pair of static strategies, i.e., *NE* [183], at which neither R_V nor R_A has any incentive to deviate from their decisions. In other words, at the *NE* strategy profile, neither R_V nor R_A can further minimize or maximize his/her potential cybersecurity loss or gain, respectively.

Theorem 6.1 shows that R_A's allocation strategy that corresponds to a pure *NE* strategy profile is an optimal solution for Problem 1.

Theorem 6.1. If the strategy profile (V^*, A^*) is a pure *NE* strategy profile, R_V's static VM allocation strategy A^* is optimal to minimize his/her *WPC* loss.

Proof. According to the Minimax method [180, 186] in a two-player zero-sum game, the strategy profile (V_{j^*}, A_{z^*}) is a pure *NE* strategy profile if:

$$\max(\min(G^T)) = \min(\max(G)) = G^* \tag{6.14}$$

where G is R_A's $(m \times \zeta)$ payoff matrix.

R_A's maximin gain (e.g., $\max(\min(G^T))$) is the highest potential security gain that he/she can be sure to get without knowing the VM allocation strategy chosen by R_V. On the other hand, R_V's minimax loss (e.g.,$\min(\max(G))$) is the smallest potential cybersecurity loss that R_V can be sure to get without knowing the targeted VM by R_A. In a zero-sum game, if the maximin and minimax payoffs are equal in the absolute value (e.g., Equation 6.14), the strategy profile solution of the maximin and minimax is the same as the pure *NE* strategy profile of the game. Moreover, according to [187], a necessary and sufficient condition for a saddle point, i.e., an *NE* strategy profile, to exist is the presence of a payoff matrix element which is both a minimum of its rows and a maximum of its columns, i.e., G^*. Therefore, the *NE* strategy profile is the optimal solution for both R_V and R_A given that each of them knows the other opponent well. $\square$

Theorem 6.1 implies that we can transform Problem 1, into a pure *NE* strategy profile search problem, and thus existing *NE* searching algorithms can be readily applied to our game model. However, we have two major problems when using those algorithms.

First, there may be no pure *NE* strategy profile in existence at all. For instance, let the $(m = 3 \times \zeta = 2)$ payoff matrix of R_A be defined as follows:

$$G = \begin{bmatrix} V_j/A_i & A_1 & A_2 \\ V_1 & 0.375 & 0.530 \\ V_2 & 0.250 & 0.625 \\ V_3 & 0.156 & 0.756 \end{bmatrix}$$

According to the Minimax method [180, 186], R_A seeks maximizing his/her minimum gain. The minimum potential gain of attacking any VM under all allocation policies is the minimum of each row in the payoff matrix G. That is:

$$G = \begin{bmatrix} V_j/A_i & A_1 & A_2 \\ V_1 & 0.375 & \square \\ V_2 & 0.250 & \square \\ V_3 & 0.156 & \square \end{bmatrix}$$

Consequently, to maximize his/her minimum potential gain, R_A should attack the first VM and gain $G_{11} = \min\{0.375, 0.250, 0.156\} = 0.3750$.

Similarly, R_V wishes to minimize his/her maximum potential loss at each allocation strategy or column of the payoff matrix G. That is:

$$G = \begin{bmatrix} V_j/A_i & A_1 & A_2 \\ V_1 & 0.375 & \square \\ V_2 & \square & \square \\ V_3 & \square & 0.756 \end{bmatrix}$$

As a result, R_V should choose the first allocation strategy such that his/her WPC loss is minimized, i.e. $G = \min\{0.3750, 0.7560\} = 0.3750$. Therefore, the strategy profile (V_1, A_1) is an NE strategy profile because none of R_V nor R_A can improve one's payoff by making a unitarily changing toward another strategy. Therefore, R_V's optimal allocation strategy in terms of minimizing the WPC loss is A_1.

Now consider this example in which there is no pure NE strategy profile. Fig. 6.2 is an illustration of the different ways of allocating ($V_1 = \{c_1 = 30\ MIPS$, $q_1 = 0.2$, $\ell_1 = 87.295\}$), ($V_2 = \{c_2 = 20\ MIPS$, $q_2 = 0.4$, $\ell_2 = 76.803\}$), and ($V_3 = \{c_3 = 30\ MIPS$, $q_3 = 0.6$, $\ell_3 = 50.06\}$) onto a three-server cloud cluster with capacity $\mathbf{C} = 800\ MIPS$. We can see that R_V has five possible static VM allocation strategies. R_A's gain under those allocation strategies are given in the payoff matrix G as follows:

$$G = \begin{bmatrix} V_j/A_z & A_1 & A_2 & A_3 & A_4 & A_5 \\ V_1 & 17.4590 & 17.4590 & 20.4626 & 22.0672 & 25.0708 \\ V_2 & 30.7212 & 36.7284 & 30.7212 & 41.1966 & 47.2038 \\ V_3 & 30.0360 & 43.8605 & 45.7491 & 30.0360 & 59.5736 \end{bmatrix}$$

If the number of servers is reduced to $n = 2$ and the capacity of each server is reduced to $\mathbf{C} = 5 \times 10^3\ MIPS$, the allocation strategies $\{A_j; j = 1, 3, 5\}$) are not feasible anymore. The only feasible allocations now are A_2 and A_4 in which either V_1 or V_3 is hosted alone.

R_A's $(m = 3 \times \zeta = 2)$ payoff matrix G is then defined as follows:

$$G = \left[\begin{array}{cccccc} V_j/A_z & A_1 & A_2 & A_3 & A_4 & A_5 \\ V_1 & \square & 17.4590 & \square & 22.0672 & \square \\ V_2 & \square & 36.7284 & \square & 41.1966 & \square \\ V_3 & \square & 43.8605 & \square & 30.0360 & \square \end{array}\right]$$

If R_V chooses the allocation strategy A_4, R_A will directly attack V_2 because it maximizes his/her gain (e.g., $G_4^* = g_{24} = 41.1966$). Note that V_2 imposes security risks (e.g., $q_h \times q_2 \times \ell_1$) on V_1. However, R_V can minimize his/her WPC loss from $L_4^* = -41.1966$ to $l_{22} = -36.7284$ by switching to the strategy A_2. R_A now can improve his/her gain by attacking V_3, i.e. $G_2^* = g_{32} = 43.8605$. To this end, R_V can minimize his/her loss from $L_3^* = l_{32} = -43.8605$ to $l_{34} = -30.0360$ by switching to the strategy A_4. This attack/defend scenario goes on. As we can see, R_V and R_A constantly change their strategies and cannot reach stable decisions, i.e. they cannot reach the pure NE strategy profile. Therefore, before we apply existing NE search algorithms, we need to study if a pure NE strategy profile exists at all for a given problem setting. Later on in this chapter, we will introduce several lemmas and theorems that characterize the existence of a pure NE strategy profile .

Second, even though the commonly used NE search algorithms, such as the Minimax method [180,186], take polynomial time concerning the size of strategy spaces of players, R_V's strategy space increases rapidly with the numbers of VMs and servers (e.g., see Fig. 6.4). which increases the time and space complexities of NE algorithms. It is, thus, necessary to study the characteristics of the game model and develop more computationally efficient algorithms to solve this problem.

In the next section, we implement two common NE search algorithms, for comparison reasons, using the MP approach to find the NE strategy profiles at which R_V's WPC loss is minimized.

6.5.1 Obtaining the optimal VM allocation solution using the Minimax method

We first use the *MP* approach to implementing the Minimax method [180, 186] to find the pure *NE* strategy profile of our VM allocation game model. The Minimax method was introduced by John Von Neumann [135]. The name of this method refers to the way it searches for the *NE* strategy profiles of a game.

The maximin payoff value of R_A is the highest potential cybersecurity gain that he/she can be sure to get without knowing the allocation strategy of R_V. On the other hand, the minimax payoff value of R_V is the smallest potential cybersecurity loss that R_V can be sure to get without knowing the targeted VM by R_A. In a zero-sum game, if the maximin and minimax payoffs are equal, the strategy profile solution of the maximin and minimax is the same as the *NE* strategy profile of the game model. The *NE* strategy profile is the optimal solution for both R_V and R_A given that each of them knows the other opponent well. Same as our notion of a pure strategy with an excellent worst-case bound.

Using the game terminology, we need to find R_A's and R_V's pure strategy probability vectors (e.g., $Y = \{y_1, y_2, ..., y_m\}^T$ and $X = \{x_1, x_2, ..., x_\zeta\}$; where $\Sigma_{j=1}^{m} y(j) = 1$; $\Sigma_{j=1}^{\zeta} x(j) = 1$; and Y and X are integer vectors) such that the pair $((Y,X) = (y_j, x_z)$; $j \in \{1, 2, ..., m\}$; $z \in \{1, 2, ..., \zeta\})$ is a pure *NE* strategy profile at which neither of R_V nor E_A can improve his/her payoff by unitarily deviating to another strategy.

If R_A successfully attacks a VM (e.g., V_j) given that R_V chooses an allocation strategy (e.g., A_z). R_V's potential loss is $L(j,z) = -G(j,z)$, i.e. a zero-sum game. R_V selects the allocation strategy A_z with a probability of $x_z = 1$. R_A attacks the VM that maximizes his/her payoff under the allocation A_z with probability $y_j = 1$. R_V's and R_A's pure strategy vectors $\{X = \{x_1, x_2, ..., x_\zeta\}$ and $Y = \{y_1, y_2, ..., y_m\}^T$ are consequently zero vectors except at the $z^t h$ and $j^t h$ positions where they have ones. Therefore, we can imply that

R_V and R_A can only choose one deterministic strategy at a time, i.e., they cannot choose multiple strategies with different probabilities and dynamically switch among them.

Let Y^* denote R_A's pure strategy solution to his/her maximin problem. Let X^* denote R_V's solution to his/her minimax problem. Based on the Minimax method [135], if Equation 6.15 holds, the strategy profile (V^*, A^*) that has the payoff $G(V_{j^*}, A_{z^*}) = Y^{*T} \times G \times X^{*T)})$ is a pure NE strategy profile and the deterministic allocation strategy A_{z^*} optimally minimizes R_V's WPC; where $x_{z^*} = y_{j^*} = 1$.

$$\begin{aligned} \max_{\forall j \in \{1,2,\ldots,m\}} \min_{\forall z \in \{1,2,\ldots,\zeta\}} Y^{*T} G\, X^T = \\ \min_{\forall z \in \{1,2,\ldots,\zeta\}} \max_{\forall j \in \{1,2,\ldots,m\}} Y^T G\, X^{*T} = \\ Y^{*T} \times G \times X^{*T} = G(V_{i^*}, A_{z^*}) \end{aligned} \tag{6.15}$$

Therefore, to solve Problem 1, we have to find the solution vectors Y^* and X^*.

The attacker's maximin solution

If R_V uses a strategy X and R_A uses a strategy Y, then R_A's expected payoff given R_V's selected strategy is:

$$\Sigma_{z=1}^{\zeta} \Sigma_{j=1}^{m} y_j \times G_{jz} \times x_z = Y^T \times G \times X^T \tag{6.16}$$

Let Fx denotes the $(1 \times \zeta)$ vector that is all zeros except for a one in the z^{*th} position, i.e., a deterministic strategy. The inner optimization is given by:

$$\min_{\forall z \in \ \{1,2,\ldots,\zeta\}} Y^T \times G \times X^T = \min_{\forall z \in \{1,2,\ldots,\zeta\}} Y^T \times G \times Fx^T \tag{6.17}$$

Now, if we introduce a scalar variable u_a representing the value of the inner minimization, we can express the maximin problem of R_A as an integer linear programming problem given in Fig. 6.7.

```
    Maximize $(u_a)$, subject to:
1: $u_a \leq Y^T \times G \times Fx^T$ **and** $\Sigma_{j=1}^{m} Y_j = 1$ **and** $\sum_{z=1}^{\zeta} Fx_z = 1$
2: $\forall j \in \{1,2,...,m\}; Y_j = \begin{cases} 1 \ \textit{if attacker attacks } V_j; \\ 0 \ \textit{otherwise}; \end{cases}$
3: $\forall z \in \{1,2,...,\zeta\}; Fx_z = \begin{cases} 1 \ \textit{if the provider selects the allocation} \\ \textit{strategy } A_z; \\ 0 \ \textit{otherwise}; \end{cases}$
```

Figure 6.7: MP-based implementation of the attacker's maximin problem

```
    Minimize $(u_v)$, subject to:
1: $u_v \geq Fy^T \times G \times X^T$ **and** $\Sigma_{j=1}^{m} Fy_j = 1$ **and** $\sum_{z=1}^{\zeta} x_\zeta = 1$
2: $\forall j \in \{1,2,...,m\}; Fy_j = \begin{cases} 1 \ \textit{if attacker attacks } V_j \\ 0 \ \textit{otherwise} \end{cases}$
3: $\forall z \in \{1,2,...,\zeta\}; x_\zeta = \begin{cases} 1 \ \textit{if the provider selects the allocation} \\ \textit{strategy } A_z; \\ 0 \ \textit{otherwise}; \end{cases}$
```

Figure 6.8: MP-based implementation of the provider's minimax problem

The provider's minimax problem

Using similar reasoning we see that R_V selects the strategy X that leads to:

$$\min_{\forall z \in \{1,2,...,\zeta\}} \max_{\forall j \in \{1,2,...,m\}} Y^T G X^T \tag{6.18}$$

and the MP formulation of the Minimax method of R_V is given in Fig. 6.8.

Where Fy denotes the vector that is all zeros except for a one in the j^*th position.

For example, let G=[0.375 , 0.530 ; 0.25 , 0.625; 0.156 , 0.756]. R_A seeks maximizing his/her minimum gain. The minimum potential gain of attacking any VM is the minimum of each row, i.e. {min [0.375 , 0.530] ; min [0.25 , 0.625], min [0.156 , 0.756]} = {0.3750 ; 0.2500 ; 0.1560}. Consequently, to maximize his/her minim potential gain, R_A should attack the first VM and thus Gain = max (min ({0.3750 ; 0.2500 ; 0.1560})) = 0.3750. Similarly, R_V wishes to minimize his/her maximum potential loss at each allocation strategy (column), i.e. {max {0.3750 ; 0.2500 ; 0.1560}, max {0.5300 ; 0.6250 ; 0.7560}}=

$\{0.3750, 0.7560\}$. As a result, R_V should choose the first allocation strategy such that the maximum loss is minimized, i.e. L = min (max $\{0.3750, 0.7560\}$)=0.3750. Therefore, the strategy profile (x=1,y=1) in which R_A attacks the first VM given that R_V chooses the first allocation is a pure NE strategy profile because none of the players can improve one's payoff by making a unitarily change to another strategy.

The Minimax method [135] returns a single NE strategy profile that optimally minimizes R_V's WPC loss and solves Problem 1. However, this method only works with zero-sum game models. Additionally, there might exist several pure NE strategy profiles with the same WPC loss (e.g., multiple saddle points [186, 187]), and with different energy and operating costs. However, the Minimax method, unfortunately, cannot identify all feasible solutions to choose the best among them (e.g., it does not solve Problem 2.). Moreover, calculating the maximin and minimax values of R_A and R_V is done in a worst-case approach. Therefore, when the number of VMs increases, the sizes of the payoff matrices quickly increase, and the algorithm cannot find the solution in a practically short time.

We implement a general-sum NE search algorithm next. This algorithm is based on searching the pure strategy support vectors of equal sizes to find the best responses of R_A to R_V and vice versa [181].

6.5.2 MP-based implementation of the Support Testing and Enumerating (STE) algorithm

In this section, we use the MP approach to implement the Bimatrix best response (e.g., we call it in this chapter the Support Testing and Enumerating (*STE*) algorithm) [181] to find R_V's optimal VM allocation strategies that correspond to a pure NE strategy profile.

Let the pure strategy support vectors of R_A and R_V be $M = \{V_1, V_2, ..., VM_m\}$ and $A = \{A_1, A_2, ..., A_\zeta\}$, respectively. M comprises all possible attack strategies for R_A, and A enumerates all R_V's different ways to allocate m distinguishable VMs to n identical servers. Let (G, L=-G) be a Bimatrix game, where G and L are $(m \times \zeta)$ matrices representing the payoffs of R_A and R_V, respectively (e.g., zero-sum game). Let the $(1 \times m)$ column vector Y and the $(1 \times \zeta)$ row vector X be the pure strategy vectors of for choosing a single VM to attack and a unique VM allocation strategy by R_A and R_V, respectively.

The *STE* algorithm depends on the best response condition in finding all *NE* strategy profiles. Lemma 6.1 restates the best response proposition presented in [181].

Lemma 6.1. *A pure strategy X of R_V is the best response to a pure strategy Y of R_A if and only if:*
$\forall j \in \{1, 2, ..., m\}$; *if* $r = Y^T \times G$ *and* $X_z > 0 \Longrightarrow r_j = a = \max_{\forall j \in \{1,2,...,m\}}\{r_j\}$
and vice versa for R_A.

For example, the best response to the pure strategy $Y > 0$ of R_A is a pure strategy $X > 0$ that minimizes R_V's expected security loss (e.g., $X \times L \times Y^T$). Similarly, the best response to the pure strategy $x > 0$ of R_V is a pure strategy $Y > 0$ that maximizes R_A's potential security gain (e.g., $Y^T \times G \times X$).

According to the Lemma 6.1, a pure *NE* strategy profile is a strategy profile (Y_{i^*}, X_{z^*}) in which each of the strategies is the best response to the other strategy. According to Nisan [181], Lemma 6.1 implies that if R_V's payoff $(X \times L \times Y^T)$ is linear in X, and if it is maximized on a face of the simplex of R_V's mixed strategies, it is also optimized on any vertex, i.e., pure strategy, of that face. Moreover, if it is optimized on a set of vertices, then it is also maximized on any convex combination of them.

This implication is significant because it states a finite condition about finding all pure strategies rather than using the infinite set of all mixed policies. Fig. 6.9 shows how to find

```
Input: G, L=-G
Output: All Nash equilibria of the game
1: for each $k \in \{1,2,...,min\{m,\zeta\}\}$ do
2:     for each pair $(Sub_I, Sub_\zeta)$ of k-sized subset where $Sub_I \subset \{1,2,...,m\}$ and $Sub_Z \subset \{1,2,...,\zeta\}$ do
3:         $find\ u_v \leq \Sigma_{\forall z \in Sub_Z} x_\zeta \times G_{jz};\ for\ j \in Sub_I$
4:         $find\ u_a \geq \Sigma_{\forall j \in Sub_I} G_{jz} \times Y_j;\ for\ z \in Sub_Z$
5:         if $X > 0$ & $Y > 0$ & Lemma 1 holds for both $X$ and analogously $Y$ then
6:             The strategy profile (V˙i,A˙z) is a pure Strategy Nash equilibrium
7:         end if
8:     end for
9: end for
```

Figure 6.9: MP-based implementation of the STE algorithm

```
   $max\ u_v = \Sigma_{j=1}^{m} y_j\ G_j z^*$, subject to:
1: $u_v \geq \Sigma_{j=1}^{m} y_j\ G_{iz}; \forall z \in \{1,2,...,\zeta\}$
2: $k = 1$
3: $z* \leq \{1,2,...,\zeta\}$
4: $X_{z*} > 0$
5: $Y_{i^*} > 0; i^* \in \{1,2,...,m\}$
6: $Y_j = \begin{cases} 1\ if\ attacker\ chooses\ to\ attack\ V_j; \\ 0\ otherwise; \end{cases}$
7: $x_\zeta = \begin{cases} 1\ if\ provider\ chooses\ allocation\ A_z; \\ 0\ otherwise; \end{cases}$
8: $\Sigma_{\forall z \in \{1,2,...,\zeta\}}\ x_\zeta = 1$
```

Figure 6.10: Calculating the provider's best response u_v

all pure *NE* strategy profiles for the security allocation game based on the best response condition [181].

For only finding pure *NE* strategy profiles, we can set k to 1. For example, X_{z^*} is the best response to Y_{j^*}, and Y_{j^*} is the best response to X_{z^*} if there exists $u_v = u_a$ for the same i^* and z^*, such that u_v is the solution to the optimization program in Fig. 6.10.

For example, using the same payoff matrix from the previous example, L=-G=[-0.375 -0.530; -0.25 -0.625; -0.156 -0.756]. The best response to $x_1 = 1$ or to $X = (1,0)$ is $y_1 = 1$ or $Y = \{[1,0,0]^T\}$ because $\max_{\forall i \in \{1,2,3\}}\{[100] * G\} = G_{11} = 3.75$.

```
    $max\ u_a$, subject to:
 1: $u_a = \Sigma_{\forall z \in \{1,2,\dots,\zeta\}}\ G_{i^*z}\ x_\zeta$
 2: $u_a \geq \Sigma_{\forall z \in \{1,2,\}}\ G_{jz}\ x_\zeta; for j \in \{1,2,\dots,m\}$
 3: $k = 1$
 4: $j^* \in \{1,2,\dots,m\}$
 5: $Y_{j^*} > 0$
 6: $X_{z*} > 0; z* \in \{1,2,\dots,\zeta\}$
 7: $Y_j = \begin{cases} 1\ if\ the\ attacker\ chooses\ to\ attack\ V_j \\ 0\ otherwise \end{cases}$
 8: $\Sigma_{\forall j \in \{1,2,\dots,m\}}\ Y_j = 1$
 9: $x_\zeta = \begin{cases} 1\ if\ the\ provider\ chooses\ the\ allocation\ A_z \\ 0\ otherwise \end{cases}$
10: $\Sigma_{\forall z \in \{1,2,\dots,\zeta\}}\ x_\zeta = 1$
```

Figure 6.11: Calculating the attacker's best response u_a

The pure strategy $y_j > 0$ is the best response to $X_z > 0$ if and only if u_a is the solution to the optimization program given in Fig. 6.11.

Similarly, y_1 is the best response to x_1 because $\max_{\forall z \in \{1,2\}} G * [10]^T = G_{11} = 0.375$. Therefore, (x_1, y_1) is a pure *NE* strategy profile.

The general-sum *STE* algorithm can find all the pure *NE* strategy profiles of our game model which optimally solves Problem 1. Moreover, we can use R_V's OLC instead of the loss matrix L to solve Problem 2. Nonetheless, *STE* has a worst-case execution time as the numbers of VMs and servers increase. Chen and Deng [188] proved that finding an *NE* strategy profile is PPAD complete. PPAD problems are a class of hard decision problems, but they are weaker than NP problems. However, according to Chen and Deng [188], while the question of whether an optimal solution exists for a given game model cannot be NP because the answer is always –yes–, the question of whether a second solution exists is NP-hard.

One common approach to solve Problem 1 and Problem 2 with the manageable computational cost is to employ bin-packing algorithms [92, 131]. The problem is how to integrate the characteristics unique to the problem itself to improve the effectiveness and efficiency of the bin-packing approach.

In what follows, we analyze the game model and present our interesting finding, based on which, we then develop several effective and efficient allocation algorithms that outperform the commonly adopted allocation methods in cloud data centers.

6.6 Game theoretical analysis of the security-aware VM allocation problems

In this section, we present some interesting properties of the security-aware VM allocation problem and its game model. For ease of our presentation, we first introduce the following definitions:

Definition 6.1. The allocation strategy A_z is a Static Equilibrium (SE) allocation strategy (e.g., A^*) if the strategy profile (V_j, A_z) is a pure NE strategy profile (e.g., (V^*, A^*)).

Definition 6.2. A VM (e.g., $V^\circ = \{c^\circ, q^\circ, \ell^\circ\}$) is a Primary VM (e.g., $PrVM$) if it has the highest potential cybersecurity loss when it is successfully and directly attacked. i.e.:

$$q^\circ \times \ell^\circ \geq q_j \times \ell_j; \forall j \in \{1, 2, ..., m\} \tag{6.19}$$

Now we can present some interesting properties for the security-aware VM allocation problems. We start by introducing Theorem 6.2 which studies how the number of VMs or servers affects the R_V's WPC loss in cloud clusters.

Theorem 6.2. Let A_1 and A_2 be two different allocation strategies with WPC losses (L_1^*) and (L_2^*), sets of VMs $(M_1 = \{V_j;\ j = 1, 2,\ ...,\ m_1\})$ and $(M_2 = \{V_j;\ j = 1, 2,\ ...,\ m_1,\ ...,\ m_2\})$, and number of idle servers $(n_1' \leq n_1)$ and $(n_2' \leq n_2)$, respectively. $(G_1^* \leq G_2^*)$ if one of the following is true:

1. $M_1 \subseteq M_2$ and $n'_1 = n'_2$;

2. $n'_1 \geq n'_2$ and $M_1 = M_2$;

Proof. First, R_V's worst potential cybersecurity losses are $L^*_1 = -G^*_1$ and $L^*_2 = -G^*_2$ under A_1 and A_2 in which R_A attacks V_j and V_k when they are allocated to S_{i_1} and S_{i_2}. If $M_1 \subseteq M_2$ and $n'_1 = n'_2$, we have $m_1 \leq m_2$. From Equations 6.2 and 6.4 we can write:

$$G^*_1 = \max_{\forall j \in \{1,2,\dots,m_1\}} g_{j1} \tag{6.20}$$

$$G^*_2 = \max(G^*_{j1}, \max_{\forall j \in \{1,2,\dots,m_2\}} g_{j2}) \tag{6.21}$$

Therefore, $G^*_1 \leq G^*_2$, and consequently (1) is correct.

Second, if $n'_1 \geq n'_2$ and $M_1 = M_2$, the average number of VMs per server under A_1 is less than the number of VMs under A_2. Consequently from Equation 6.2

$$g_{j1} \leq g_{j2}; \forall j \in \{1,2,\dots,m_1 = m_2\} \tag{6.22}$$

From Equation 6.22 and Equation 6.4, we conclude that $G^*_1 \leq G^*_2$ when $n'_1 \geq n'_2$ and $M_1 = M_2$ □

According to Theorem 6.2, the WPC loss increases as the number of VMs increases or as the number of non-idle servers decreases. In the next Lemma, we identify R_V's WPC loss upper and lower bounds.

Lemma 6.2. The provider's WPC losses ($L^*_z \in [LB, UB]; z = 1,2,\dots,\zeta$); where the Lower-Bound (LB) and Upper-Bound (UB) are R_V's minimum and maximum WPC losses and defined as follows.

$$LB = q^\circ \times \ell^\circ; where\ (V^\circ : \{c^\circ, q^\circ, \ell^\circ\})\ is\ a\ primary\ VM. \tag{6.23}$$

$$UB = \max_{\forall j \in \{1,2,\dots,m\}} (q_j \times \ell_j + q_j \times q_h \times \Sigma^m_{k=1,k\neq j} \ell_k) \tag{6.24}$$

Proof. According to Theorem 6.2, the more the number of available servers is, the larger R_V's WPC loss. In other words, R_V suffers the most loss under the allocation strategies with the minimum number of non-empty servers and vice versa. That is:

$$LB = q^\circ \times \ell^\circ \tag{6.25}$$

where all the primary VMs (e.g., $(V^\circ : \{c^\circ, q^\circ, \ell^\circ\})$) and all other non-primary VMs are allocated alone to separate servers. Moreover, when all VMs are allocated to a single server, R_A directly attacks the VMs that maximizes his/her gain, i.e., Equation 6.24. □

As mentioned before, it is important to investigate the existence of a pure NE strategy profile in our game model before we develop efficient pure NE search algorithms. A pure NE strategy profile [183] is a pair of allocation and attack Equilibrium strategies. However, R_V may or may not have a possible optimal VM allocation strategy [183]. In this regard, Lemma 6.3 states that when the number of servers is larger than the number of VMs, there always exists at least a single pure NE strategy profile.

Lemma 6.3. A strategy profile (V^*, A^*) is a pure NE strategy profile if R_A's selected strategy is to attack a $PrVM$ (e.g., $V^* = V^\circ$) and if R_V's selected allocation strategy (e.g., $A^* = A_{z^*}$) is to allocate each VM alone to a separate server when feasible.

Proof. If the allocation policy A_{z^*}, in which R_V statically allocates a single VM to each server, is feasible (e.g., $(n \geq m)$), we find according to the Minimax method [135] that:

$$\begin{gathered} G^* = \max_{\forall j \in \{1,2,\ldots,m\}} \big(\min_{\forall z \in \{1,2,\ldots,\zeta\}} g_{jz} \big) = \\ \max\{g_{1z^*}, g_{2z^*}, \ldots, g_{mz^*}]^T\} \end{gathered} \tag{6.26}$$

where A_{z^*} is R_V's selected strategy in which R_V allocates each VM separately.

Moreover, from Equations 6.4 and Equation 6.19, we can write:

$$G^* = G^*_{z^*} = q^\circ \ell^\circ \tag{6.27}$$

where ($V^\circ : \{c^\circ, q^\circ, \ell^\circ\}$)) is one of the primary VMs. That is, R_V's WPC loss (e.g., $L^* = -G^*$) is equal to the WPC loss under the allocation strategy A_{z^*} when R_A's attacks a $PrVM$ (e.g., V°).

We have based on Equations 6.19:

$$\min_{\forall z \in \{1,2,\dots,\zeta\}} \left(\max_{\forall j \in \{1,2,\dots,j\}} g_{jz} \right) = G^*_{z^*} = q^\circ \ell^\circ \tag{6.28}$$

From Equations 6.19, Equation 6.26, and Equation 6.28, we see that:

$$\begin{aligned} G^* &= \max_{\forall j \in \{1,2,\dots,m\}} \left(\min_{\forall z \in \{1,2,\dots,\zeta\}} g_{jz} \right) \\ &= \min_{\forall z \in \{1,2,\dots,\zeta\}} \left(\max_{\forall j \in \{1,2,\dots,j\}} g_{jz} \right) = G^*_{z^*} = q^\circ \ell^\circ \end{aligned} \tag{6.29}$$

From Equations 6.29 and Equation 6.28 and according to the Minimax method [135], the strategy profile (V°, A_{z^*}) is a pure NE strategy profile. □

Note that there may be multiple pure NE strategy profiles, i.e. multiple SE allocation strategies with the same minimum WPC loss for R_V, [186, 187], even though not all VMs are hosted separately. For example, when R_V allocates each $PrVM$ alone to a separate server but multiplexes other non-primary VMs on the rest of servers (e.g., A_{z^*}), and when R_A's gain from attacking any primary VM, i.e. V°, is better than his/her gain from attacking other non-primary VMs, (V°, A_{z^*}) is a pure NE strategy profile. This conclusion is formulated in Lemma 6.4 as follows:

Lemma 6.4. A strategy profile (V°, A^*) is a pure NE strategy profile if:

1. R_V's selected allocation strategy is $(A^* = A_{z^*})$ in which all $PrVMs$ are hosted alone is feasible;
2. R_A's strategy that maximizes his/her gain is to attack a $PrVM$ (e.g., (V°)); i.e.

$$(G^*_{z^*} = q^\circ \ell^\circ) \geq g_{jz}; \forall j \in \{1,2,\dots,m\} \tag{6.30}$$

Proof. For a general two-player zero-sum game, a necessary and sufficient condition for a Saddle Point, i.e. a pure NE strategy profile (V^*, A_{z^*}), to exist is the presence of a payoff value $G^*_{z^*}$ so that: $G^*_{z^*} = q_{j^*}\ell_{j^*}$ which satisfies the following [186]:

$$\begin{gathered}\max_{\forall j \in \{1,2,\dots,m\}} \min_{\forall zin\{1,2,\dots,\zeta\}} g_{jz} = \\ \min_{\forall zin\{1,2,\dots,\zeta\}} \max_{\forall j \in \{1,2,\dots,m\}} g_{jz} = G^*_{z^*}\end{gathered} \tag{6.31}$$

Moreover, several saddle points may exist with the same worst potential loss [187].

First, if R_V statically allocates all primary VMs alone to individual servers and if the rest of the servers' total capacity can accommodate the rest of the VMs, i.e.,:

$$\Sigma^m_{j=1;j\neq j^*}(a^z_{ij} \times c_j) \leq \Sigma^n_{i=1;i\neq i^*}\mathbf{C} \tag{6.32}$$

$$\mathbf{C} = \frac{\Sigma^m_{j=1;j\neq\{k_1,k_2,\dots k_\psi\}}(a^z_{ij} \times c_j)}{n-\psi} \tag{6.33}$$

where $\{V^\circ_k; k \in \{k_1, k_2, \dots k_\psi\}\}$ is the set of all primary VMs.

Second, if Equation 6.33 and Equation 6.30 hold, we find that:

$$\begin{gathered}\max_{\forall j \in \{1,2,\dots,m\}} \min_{\forall z \in \{1,2,\dots,\zeta\}} g_{jz} = \\ G^*_{z'} = q_{j^\circ}\ell_{j^\circ}; z' \in \{1,2,\dots,\zeta'\}; \zeta' < \zeta\end{gathered} \tag{6.34}$$

From Equations 6.30, we have:

$$\begin{gathered}\min_{\forall z \in \{1,2,\dots,\zeta\}} \max_{\forall j \in \{1,2,\dots,j\}} g_{jz} = \\ G^*_{z'} = q_{j^\circ}\ell_{j^\circ}; z' \in \{1,2,\dots,\zeta'\}; \zeta' < \zeta\end{gathered} \tag{6.35}$$

From Equations 6.34 and Equation 6.35 and according to the Minimax method [135], the strategy profile (V°, A_{z^*}) is a pure NE strategy profile. □

When any $PrVM$ is collocated with other VMs, Lemma 6.3 and Lemma 6.4 won't help to find the pure NE strategy profiles. Theorem 6.3, on the other hand, helps to quickly identify the existence of a pure NE strategy profile by searching only a subset of R_V's strategy space.

Theorem 6.3. Given our system model with m VMs and n servers ($m \geq n$), assume that there exists at least one *SE* allocation strategy (e.g., A^*), then there is at least one allocation strategy $A_{z^*} = A^*$, such that all available servers are non-idle.

Proof. Let the augmented payoff matrices $AugG$ be a sub-matrix from G,i.e., $Aug\Gamma = \{A_{1'}, A_{2'}, ..., A_{\zeta'}; \zeta' = S2^{nd}(m,n) \ll \zeta\}$, that only includes the allocation strategies which has zero non-empty servers.

The sufficiency condition: If the game $(\{R_A, R_V\}, \{M, Aug\Gamma\}, \{AugG, -AugG\})$ admits (V_{j^*}, A_{z^*}) as a pure *NE* strategy profile (e.g., the strategy A_{z^*} is an *SE* strategy), the game $(\{R_A, R_V\}, \{M, \Gamma\}, \{G, L = -G\})$ also admits (V_{j^*}, A_{z^*}) as a a pure *NE* strategy profile.

If the game $(\{R_A, R_V\}$, $\{M, Aug\Gamma\}$, $\{AugG, -AugG\})$ has a pure *NE* strategy profile (V_{j^*}, A_{z^*}), and according to the Minimax method [135]:

$$\begin{gathered} \max_{\forall j \in \{1,2,...,m\}} \min_{\forall z \in \{1,2,...,\zeta'\}; \zeta' < \zeta} g_{jz} = \\ \min_{\forall z \in \{1,2,...,\zeta'\}; \zeta' < \zeta} \max_{\forall j \in \{1,2,...,m\}} g_{jz} = G^*_{z^*} \end{gathered} \tag{6.36}$$

From Theorem 6.2, each row element in the augmented payoff matrix $AugG$ is less than or equal to any row element in the payoff matrix G because the number of servers utilized in each of the allocation strategies in $Aug\Gamma = \{A_{1'}, A_{2'}, ..., A_{\zeta'}; \zeta' = S2^{nd}(m,n) \ll \zeta$ is larger than or equal to the number of servers used in each of the allocation strategies $\Gamma = \{A_z; z = 1, 2, ..., \zeta\}$; where Γ is the set of R_V's all possible allocation strategies of m VMs to n server. Therefore:

$$\begin{gathered} \max_{\forall j \in \{1,2,...,m\}} \min_{\forall z \in \{1,2,...,\zeta'\}; \zeta' < \zeta} g_{jz} \\ \max_{\forall j \in \{1,2,...,m\}} \min_{\forall z \in \{1,2,...,\zeta\}} g_{jz} = G^*_{z^*} \end{gathered} \tag{6.37}$$

and

$$\begin{gathered} \min_{\forall z \in \{1,2,...,\zeta'\}; \zeta' < \zeta} \max_{\forall j \in \{1,2,...,m\}} g_{jz} = \\ \min_{\forall z \in \{1,2,...,\zeta\}} \max_{\forall j \in \{1,2,...,m\}} g_{jz} = G^*_{z^*} \end{gathered} \tag{6.38}$$

From Equation 6.37 and Equation 6.38 we see that if the strategy profile (V_{j^*}, A_{z^*}) is a pure NE strategy profile for game $(\{R_A, R_V\}, \{M, Aug\Gamma\}, \{AugG, -AugG\})$ (e.g., the allocation strategy A_{z^*} is an SE), then based on the Minimax method [135], (V_{j^*}, A_{z^*}) must be a pure NE strategy profile as well for the game $(\{R_A, R_V\}, \{M, \Gamma\}, \{G, L\})$.

The necessity condition: If the game $(\{R_A, R_V\}, \{M, Aug\Gamma\}, \{AugG, -AugG\})$ does not admit a pure NE strategy profile (e.g., there is no SE strategy), the game $(\{R_A, R_V\}$, $\{M, \Gamma\}$, $\{G, L = -G\})$ also does not admit a a pure NE strategy profile.

Assume that the game $(\{R_A, R_V\}$, $\{M, Aug\Gamma\}$, $\{AugG, AugL\})$ has no NE strategy profile (e.g., SE) strategy profile. If the game $(\{R_A, R_V\}$, $\{M, \Gamma\}, \{G, L = -G\})$ has a pure NE strategy profile (V_{j^*}, A_{z^*}), according the Minimax method [135] we have:

$$\begin{aligned} &\max_{\forall j \in \{1,2,\dots,m\}} \min_{\forall z \in \{1,2,\dots,\zeta\}} g_{jz} = \\ &\min_{\forall z \in \{1,2,\dots,\zeta\}} \max_{\forall j \in \{1,2,\dots,m\}} g_{jz} = G^*_{z^*} \end{aligned} \tag{6.39}$$

From Theorem 6.2, each row element in the augmented payoff matrix $AugG$ is less than or equal to any row element in the payoff matrix G. Therefore, from Equation 6.39 and Theorem 6.2, we conclude that $A_{z^*} \in Aug\Gamma$. Consequently, the payoff element $G^*_{z^*} \in AugG$ and the strategy profile (V_{j^*}, A_{z^*}) is a pure NE strategy profile (e.g., SE) to the game $(\{R_A, R_V\}, \{M, Aug\Gamma\}, \{AugG, -AugG\})$. However, this contradicts our assumption that the game $(\{R_A, R_V\}, \{M, Aug\Gamma\}, \{AugG, -AugG\})$ has no pure NE strategy profile. Consequently, when the game $(\{R_A, R_V\}, \{M, Aug\Gamma\}, \{AugG, -AugG\})$ has no pure NE strategy profile (e.g., there exists no SE allocation strategy), the game $(\{R_A, R_V\}, \{M, \Gamma\}, \{G, L = -G\})$ also has no pure NE strategy profile. □

Next, we propose two equilibrium-based VM allocation heuristics based on our analysis insight.

6.7 Game theoretic approaches for security-aware VM allocation strategies

In this section, we present two game-base security-aware VM allocation algorithms. The first approach, called *Nash Equilibrium-based VM (NEVM) allocation algorithm*, tries to identify all the possible pure *NE* strategy profile and, thus, the optimal static VM allocation (e.g., *SE*) policies according to the game model presented in Section 6.6. The second approach, called the *Primary VM-based (PVM) bin-packing algorithm*, is a variation of the vector bin-packing method centering around the allocation of the *PrVMs*. More details about these two approaches are described next.

6.7.1 Nash Equilibrium-based VM (NEVM) allocation algorithm

The key to the success of this approach is to identify the pure *NE* strategy profiles if they exist. Recall that, as discussed in Section 6.6, even though the *NE* searching strategies take polynomial time, the strategy space of R_V exponentially increases with the increasing numbers of VMs and servers. In the meantime, the *NE* strategy profile properties presented earlier make it possible to significantly reduce the strategy space for R_V, and may make it possible to find the optimal allocation strategies in a reasonable time frame.

Our first algorithm, i.e. *NEVM*, is illustrated in Fig. 6.16. First, *NEVM* identifies all the primary VMs (e.g., line 1). According to Lemma 6.3 and Lemma 6.4, *NEVM* should only search for *SE* policies among those allocations when all primary VMs are hosted alone (e.g., lines [2-9]). This can significantly reduce the size of the strategy space (e.g., ($\zeta_0 = STN(m-m_0, n-m_0)$)). Note that *NEVM* uses the recursive function, i.e. Equation 6.12 in Section 6.3, to reduce the space complexity when generating those allocation strategies (e.g., lines [3-10]).

```
Input: $M = \{V_1, V_2, ..., V_m\}$ and $Sr = \{S_1, S_2, ..., S_n\}$
Output: The EQuilibrium $EQ$ set of all $SE$ VM allocation policies
 1: Identify all $PrVMs$, (e.g., $M^\circ = \{V_j^\circ = \{c_j^\circ, q_j^\circ, \ell_j^\circ\}; j = 1, 2, ..., m_0\}$
    using Equation 17
 2: if $n$ and $\mathbf{C}$ are sufficient to separately host each $PrVM$ alone then
 3:     Construct the $(m \times \zeta_0)$ payoff matrix $G$ in which only the
        allocation strategies that host each $PrVM$ alone are generated;
        where $(\zeta_0 = STN(m - m_0, n - m_0))$
 4:     $G^* \leftarrow q_j^\circ \times \ell_j^\circ$; where $(j = 1, 2, ...,$ or $m_0)$
 5:     for Each $z \in \{1, 2, ..., \zeta_0\}$ do
 6:         if $G_z^* = G^*$ then
 7:             Add $A_z$ to $EQ$
 8:         end if
 9:     end for
10: else
11:     Construct the $(m \times \zeta_1)$ payoff matrix $G$ in which only the allo-
        cation strategies, i.e. $\Gamma_1 = \{A_z; z = 1, 2, ..., \zeta_1\}$, that have no idle
        servers are generated; where $(\zeta_1 = STN(m, n))$
12:     if $\max_{\forall j \in \{1,2,...,m\}} \min_{\forall z \in \{1,2,...\zeta_1\}} G^T =$
        $\min_{\forall z \in \{1,2,...,\zeta_1\}} \max_{\forall j \in \{1,2,...,m\}} G^T = G^*$ then
13:         Generate all allocation strategies
            $(\{\Gamma_2 = \{A_z; z \in \{1, 2, ..., \zeta\}; G_z^* = G^*\})$
14:         $EQ \leftarrow \Gamma_2$
15:     end if
16: end if
17: Return EQ
```

Figure 6.12: Nash Equilibrium-based VM (NEVM) allocation algorithm

If it is not feasible to host all the primary VMs alone, based on Theorem 6.3, $NEVM$ constructs the $(m \times \zeta_1)$ payoff matrix G in which only the allocation strategies, i.e. $\Gamma_1 = \{A_z; z = 1, 2, ..., \zeta_1\}$, that have no idle servers are generated (e.g., line 11). As a result, the strategy space for R_V can be significantly reduced (e.g., $(\zeta_1 = STN(m, n))$). For example, Fig. 6.4 shows that when the numbers of VMs and servers are $n = m = 15$, the number of R_V's pure strategies, based on Equation 6.12, is large (e.g., $\zeta = 1.3830e+09$). However, $NEVM$ can cut down the total number of those strategies to $\zeta_1 = 190899322$, which is more than 85% reduction for R_V's original pure strategy space. After R_V's reduced strategy space is generated, we can employ the existing NE search approaches (such as the Minimax method [135] which take only computational complexity of $O(m\zeta)$) to find the pure NE strategy profiles, if they exist (e.g., line 12). If there exists any pure NE with an optimal WPC loss $(L^* = -G^*)$, $NEVM$ generates and returns its corresponding allocation strategy in which the WPC loss is equal to G^* (e.g., lines [13-14]). If a pure NE strategy profile does not exist, $NEVM$ returns empty set (e.g., Line 17).

Besides significantly reducing the size of R_V's strategy space, one unique advantage of using $NEVM$ is that it can identify all possible pure NE strategy profiles (if they exist) which share the same WPC loss and solve Problem 1. We can then further optimize the allocation policies according to other criteria (e.g., energy costs and operating expenses) to solve Problem 2. Nonetheless, $NEVM$ has a computation complexity, i.e., time and space, that increases exponentially with the increasing numbers of VMs and servers. Also, as discussed before, there may exist no pure NE strategy profile at all for some game models. In what follows, we develop another bin-packing-based approach, which can work for more massive cloud clusters.

6.7.2 The Primary VM-based bin-packing (PVM) algorithm

The VM allocation problem we address here is, in general, a resource management problem with multiple optimization criteria, i.e., energy/operating expense and cybersecurity loss. Traditionally, it is a common practice to transform this problem into a *VBP* problem. While typical *VBP* approaches (e.g., [131]) are available, the key to the success of the solution is how to incorporate the problem properties into the bin-packing heuristics to improve their effectiveness and efficiency. In Section 6.6, our research has shown the significant role that $PrVMs$ may play in the security-aware VM allocation. In what follows, we discuss The Primary VM-based bin-packing (PVM) algorithm to identify the optimized VM allocation policies, as shown in Fig 6.13.

```
Input: $M = \{V_1, V_2, ..., V_m\}$, $Sr = \{S_1, S_2, ..., S_n\}$, and $Thrsh$
Output: The VM allocation policies $A_z$ in which $R_V$'s $WPC$ loss (e.g.,
    $L_z^* = -G_z^*$) is minimized (e.g., Equation 4)
 1: Sort VMs based on the $WPC$ loss, i.e. $(q \times \ell)$ and identify all $PrVMs$,
    (e.g., $M^\circ = \{V_j^\circ = \{c_j^\circ, q_j^\circ, \ell_j^\circ\}; j = 1, 2, ..., m_0\}$) using Equation 17
 2: $LC \leftarrow q_j^\circ \times \ell_j^\circ$; where $(j = 1, 2, ..., \ or \ m_0)$
 3: Allocate $PrMVs$ to the larger possible number of servers and remove
    them from $M$
 4: if a primary server $S_i$ has more than one $PrVM$ then
 5:     $LC = \max_{\forall j \in \{1,2,...,m_i\}; V_j \in S_i} g_{jz}$
 6:     if $LC > Thrsh$ then
 7:         Return allocation failed!
 8:     end if
 9: end if
10: Sort the non-primary VMs according to the decreasing value of their
    weighted sum of security and resource usage using Equations 37, 38,
    and 39
11: if Allocating the non-primary VMs (e.g., $\{V_j; j \in \{m_0+1, m_0+2, ..., m\}\}$) to the rest of servers ($\{S_i; i \in \{m_0+1, m_0+2, ..., n\}\}$)
    using FFD if feasible & if $G_z^* \leq LC$ then
12:     Remove all VMs from M
13:     Return allocation results
14: else
15:     Sort the servers in $Sr$ by placing the empty non-primary servers
        first
16:     for each non-primary VM and non-primary server do
17:         if the current server is non-primary with sufficient capacity
            then
18:             if $LC \geq G_z^* = \max_{\forall j \in \{1,2,...,m_i\}; V_j \in S_i} g_{jz}$ then
19:                 Allocate the current VM and remove it from $M$
20:             else if the current server is a primary with sufficient capacity
                then
21:                 Allocate the current VM and remove it from $M$ if the
                    new $LC < Thrsh$
22:                 $LC = \max(LC, G_z^* = \max_{\forall j \in \{1,2,...,m_i\}; V_j \in S_i} g_{jz})$
23:                 Make $S_i$ the last server in order in $Sr$
24:             end if
25:         end if
26:     end for
27:     if $M \neq []$ then
28:         Return allocation results
29:     else
30:         Return allocation failed!
31:     end if
32: end if
```

Figure 6.13: The Primary VM-based (PVM) bin-packing algorithm

PVM in general consists of two phases, i.e., Phase I (lines [1-13]) and Phase II (e.g., lines [13-32]). During Phase I, *PVM* sorts VMs based on their *WPC* loss and individually allocates each *PrVM* to a primary server, if possible, (e.g., lines 1-9). The rest of the VMs are sorted and assigned, based on a variation of the *FFD*-based *VBP* methods [131], to the rest of the servers with the most significant security loss no more than those when directly attacking a *PrVM*, (e.g., lines 10-13). If all VMs are successfully allocated, based on Lemma 6.4, the algorithm has identified a pure *NE* strategy profile that optimally minimizes R_V's *WPC* loss. If not, the algorithm enters the second phase in which primary servers are allowed to host non-primary VMs as well.

Specifically, we sort the rest of VMs in M based on the number of servers, the *WPC* loss, and capacity requirements of VMs (e.g., line 7). Specifically, for a VM $V_j : \{c_j, q_j, \ell_j, \}$, we define the scalar that combines both optimization criteria as W_j such that:

$$W_j = (a_L \times q_j \times \ell_j) + (a_c \times c_j) \tag{6.40}$$

where a_L and a_c are the averaged sum of the cybersecurity loss and resource requirements normalized to the total number of servers. They allow us to combine the demands of each VM across both the security and resource dimensions according to the importance of each VM's demand. We calculate a_L and a_c as follows.

$$a_L = \frac{1}{n} \Sigma_{j=1}^{m} q_j \times \ell_j \tag{6.41}$$

$$a_c = \frac{1}{n} \Sigma_{j=1}^{m} c_j \tag{6.42}$$

PVM sorts the servers by placing the servers hosting the primary VMs (e.g., primary servers) at the end of the queue of servers (e.g., line 15). *PVM* then allocates the rest of the non-primary VMs using *FFD* based on W_j to the servers with no primary VMs, if it is feasible, and if the *WPC* loss is always less than the *Loss Capacity* (*LC*) (e.g., lines

[16-19]). The algorithm bounds the value of LC by the cybersecurity loss threshold, i.e., $LC <= Thresh$. When PVM is used to solve Problem 1, $Thrsh$ can be set to the WPC loss's upper bound (e.g., $Thrsh = UB$, i.e. $LB \leq Thrsh \leq UB$). If there are no more feasible servers with no existing primary VMs, PVM allocates a non-primary VM to one of the primary servers, increases LC as long it does not exceed the value of $Thrsh$, and moves that server to the end of the queue (e.g., lines [20-26]). This process judiciously load-balances resource requirements and potential cybersecurity losses across all servers. PVM continues until no VM can be further allocated, indicating that there is not enough computing capacity to hold any of the rest VMs, or all VMs have been successfully allocated (e.g., lines [27-32]).

Algorithm PVM has a computational complexity of $O(max(mn, m\log m))$ during Phase I, and $O(max(m^2n, m^2\log m))$ during Phase II, which is much smaller than $NEVM$. However, it is worth mentioning that $NEVM$ returns all optimal solutions in terms of minimizing R_V's WPC loss (if there exists any pure NE strategy profile). PVM, on the other hand, returns only one solution, which may or may not be optimal.

In the next section, we use simulation to study the effectiveness and efficiency of our proposed approaches.

6.8 Experimental validation

In this section, we evaluate the performance of the VM allocation approaches we proposed earlier in this chapter. First, we describe the simulation setup and the methods we implemented, against which, to compare the performance of our proposed algorithms. Second, we study the performance of the mathematical programs MPL and MPR, which we developed in Section 6.4, under different numbers of VMs and servers and using different server's capacity. Third, we analyze the performance of the NE-based allocation algorithm (e.g., $NEVM$). Finally, we evaluate the performance of our proposed heuristic PVM under cloud clusters with small, moderate and large sizes.

6.8.1 Simulation setup

We randomly generated the numbers of servers and VMs. A server could have a processing rate capacity ($\mathbf{C}$) in the range ($[5,\ 25] \times 10^3\ MIPS$). The operating expense Eop_z of the cluster under an allocation A_z is calculated using Equation 6.5 where the operating expense of each server e_i^z is randomly generated so that it is correlated to the server's capacity. The processing rate requirements of a VM were randomly generated out of the five sizes $\{1,\ 2,\ ...,\ or,\ 5\} \times 10^3\ MIPS$. The following values were also generated randomly and uniformly in the associated ranges: ($q_j \in [0.1,\ 0.9]$, $q_h \in [0.3,\ 0.5]$, and $\ell_j \in [\$10,\ \$100] \times 10^3$).

We ran all experiments on an HP Workstation Z800 with two Intel Xeon Six-Core E5645 2.40 GHz, 12 MB cache, 1333 MHz DDR3 memory of size 32 GB, and 1 TB disk space.

We model the server's power (Watt) usage Per Hour (WPH) as in Equation 6.7. The maximum WPH for each server per hour is proportional to its capacity (e.g., between $[500, 2500]\ WPH$).

Now, to calculate a cluster's energy cost under an allocation A_z over a year, we use Equation 6.43:

$$E^z = \left(\frac{Operating\ Hours(i.e., 24 \times 365) \times (Power\ Consumption\ WPH)}{1000}\right) \times Electricity\ cost\ per\ KWH\ (i.e.,\ 0.116\ per\ KWH\ in\ FL) \tag{6.43}$$

The *Total energy Cost and operating expense (TC_z)* of R_V under an allocation A_z over a year is then calculated using Equation 6.8. R_V's *Overall WPC Loss and total energy Cost* *and operating expense (OLC_z)* under an allocation A_z is then calculated using Equation 6.9.

We implemented the *MPL* and *MPR* approaches, described in Section 6.4, using the package JuMP [189]. JuMP is a domain-specific modeling language for mathematical optimization embedded in Julia. JuMP uses a generic solver-independent interface provided by the MathProgBase package [189]. JuMP allowed us to adopt the Gurobi Optimizer as a solver [190]. We parallelized the run of the mathematical program instances over 12 cores.

We also implemented the Minimax method [135] and *STE* [181] discussed in Section 6.5.

Moreover, we implemented the following *FFD*-based *VBP* heuristics that are commonly used for VM placement in data centers and cloud clusters:

- *AvgSum*: This algorithm is an implementation of the FDD-based average sums of the VM's potential cybersecurity loss and cybersecurity [191]. The weight of a V_i is calculated as follows:

$$W_j = a_1 \times (\ell_j \times q_j) + a_2 \times c_j; j \in \{1, 2, ..., m\} \tag{6.44}$$

  and the weights of the demands are calculated as follows:

$$a_1 = \frac{1}{m} \Sigma_{j=1}^{m} \ell_j \times q_j \tag{6.45}$$

$$a_2 = \frac{1}{m} \Sigma_{j=1}^{m} c_j \tag{6.46}$$

- *Prod*: This algorithm is a variant of the First Fit Decreasing based Production (*FFDProd*) algorithm [131] that calculates the weight of a VM (e.g., V_j) by multiplying its potential cybersecurity value (e.g., $\ell_j \times q_j$) by its processing rate requirement (e.g., c_j). That is:

$$W_j = \ell_j \times q_j \times c_j; j \in \{1, 2, ..., m\} \tag{6.47}$$

- *Probabilistic Weighing with Norm-based FFD (PWN)*: This heuristic uses the Geometric DotProduct [133]. It sorts all VMs after each allocation according to the FFD bin packing method and based on the weight of VMs that are defined as follows:

$$W_j = a_1 \times (\ell_j \times q_i \times (Thrsh - g_{jz})) + a_2 \times c(i) \times (\mathbf{C} - \Sigma_{k=1}^{m} c_k) \tag{6.48}$$

where $j \in \{1, 2, ..., m\}$; g_{jz} is the potential cybersecurity loss given an attack on V_j (e.g., Equation 6.2).

$$a_1 = \frac{1}{m} \Sigma_{i=1}^{m} \ell_j \times q_i \tag{6.49}$$

$$a_2 = \frac{1}{m} \Sigma_{i=1}^{m} c_i \tag{6.50}$$

When there are no security constraints (e.g., Problem 1), the value of $Thrsh$ can be set to the WPC loss' upper bound given in Equation 6.24, i.e. $Thrsh = UB$.

In the next section, we study the performance of the mathematical programs MPL and MPR, which we developed in Section 6.4, under different numbers of VMs and servers and using different server's capacity.

6.8.2 Studying the performance of the mathematical programs MPL and MPR

In this section, we study the computation performance of the mathematical programs, MPL, and MPR.

MPL and MPR computation performances under different number of servers and VMs

First, we study the computation time performance of MPL and MPR. In Fig. 6.14(a), we varied the number of servers and VMs from 25 servers and 50 VMs to 250 servers and

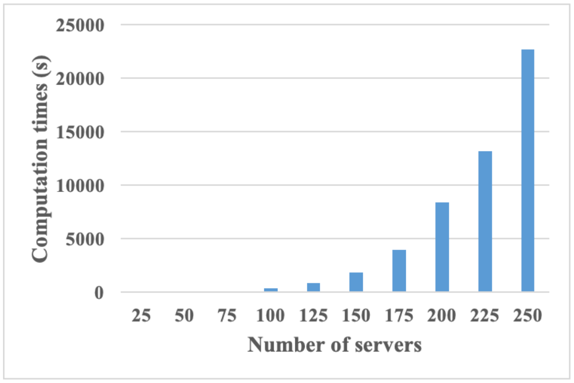
500 VMs by increasing the number of VMs and servers two times at each run. We ran *MPL* and *MPR* 10 times for each setting and averaged the results. Fig. 6.14(a) shows the final results of both programs divided by two as both mathematical programs have similar computation performances. We observe that the average computation time of *MPL* and *MPR* exponentially increases when the number of VMs and servers increase. For example, the average computation time when using 25 servers and 50 VMs increases by roughly 8000 times when increasing the number of servers and VMs to 250 servers and 500 VMs, respectively.

Fig. 6.14(c) shows R_V's WPC performance using *MPL* under different server's capacities. We set the number of VMs to a 100 VMs and the number of servers to 30 servers. We varied the server's capacities from 5×10^3 *MIPS* to 25×10^3 by adding 5000 *MIPS* each time, ran the simulation a 100 times, and averaged the results. Fig. 6.14(c) shows that as the server's capacity increases, R_V's WPC loss increases because the loss upper bound UB increases (e.g., Equation 6.24). Moreover, a larger server's capacity means that more and more VMs can be possibly multiplexed together on a single server, which decreases the average computation time of the *MPL* and *MPR* programs. For example, Fig. 6.14 (b) shows that as the capacity increases by five times, the average computation time of *MPL* and *MPR* decreases by almost 600%.

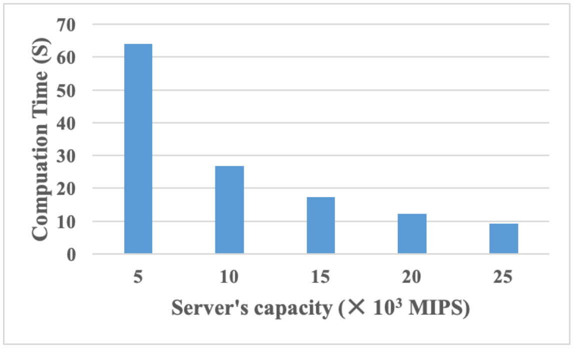
Next, we study the performance of the *NEVM* algorithm.

6.8.3 Studying the performance of the NEVM algorithm

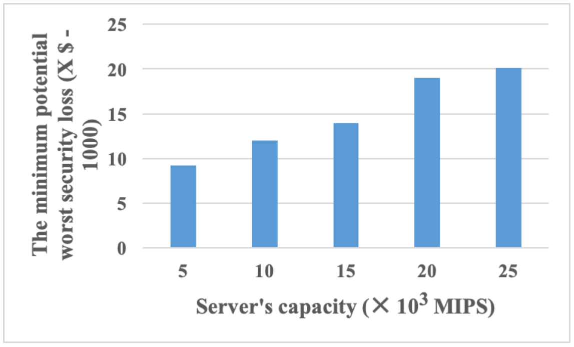
In this section, we evaluate the performance of our *NEVM* algorithm in terms of minimizing R_V's and R_A' strategy spaces and in terms of optimizing the *NE* search time compared to the Minimax and *STE NE* search methods.



(a) Average computation time of MPL and MPR under variable number of servers (n) and VMs $(m = 2 \times n)$



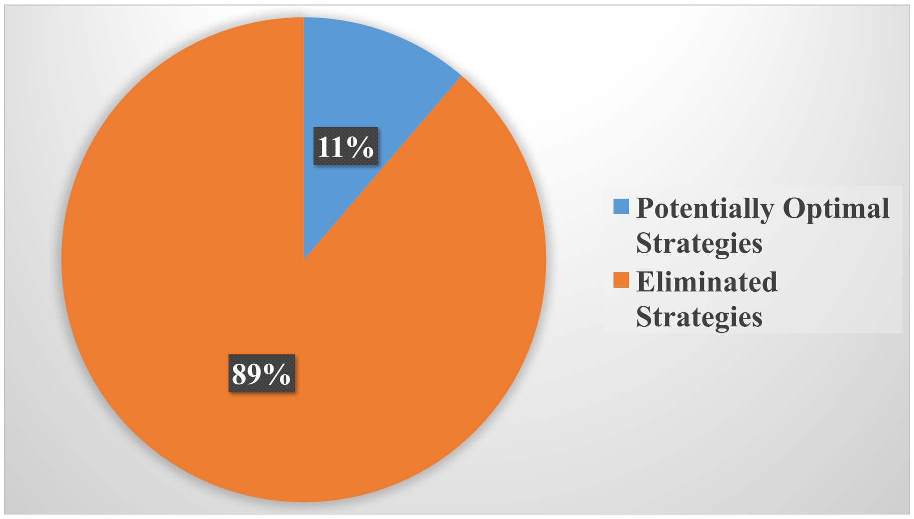
(b) Average computation time of MPL and MPR under different server's capacities



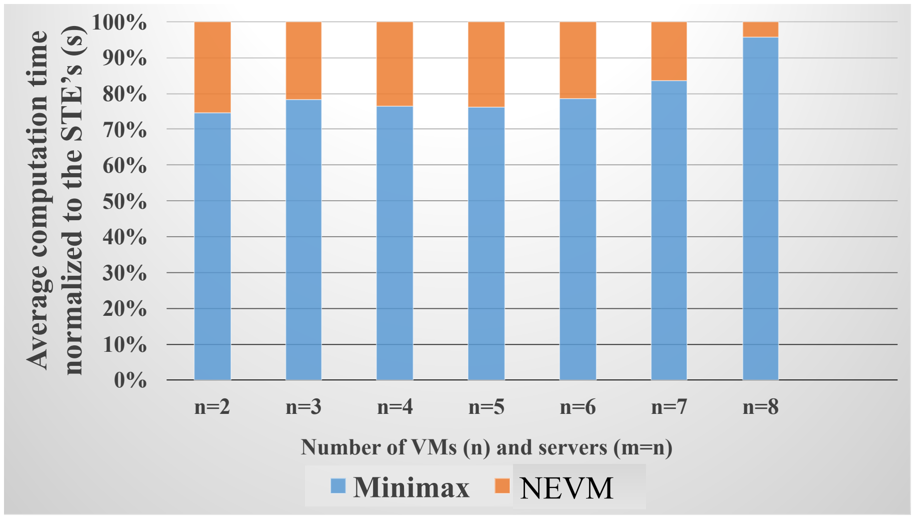
(c) The minimum worst potential cybersecurity loss of MPL under different server's capacities

Figure 6.14: The performance of the mathematical programs MPL and MPR

Fig. 6.15(b) illustrates the average performance for $NEVM$ algorithm in minimizing the size of R_V's pure allocation strategy support-vector. The number of VMs (m) and



(a) NEVM performance in minimizing the size of the provider's pure allocation strategy space



(b) Computation time comparison of NEVM against the Minimax and STE algorithms

Figure 6.15: The performance of NEVM algorithm

servers (n) were randomly generated in the range $[2,20]$. The capacity of the servers was randomly generated in the range $[5,25] \times 10^3$ $MIPS$. Unlike the STE and Minimax methods that always search all the pure strategy spaces of R_A and R_V, $NEVM$ only searches the allocation strategies that are potentially optimal, as discussed in Lemma 6.4 and Theorem

6.3. For example, based on Theorem 6.3, $NEVM$ only searches 11% of R_V's total possible allocation strategies in order to identify all SE allocation strategies. Such minimization in the total number of the allocation strategies significantly optimizes the computation time of $NEVM$ compared to the computation times of the Minimax and STE, as illustrated in Fig. 6.15(b). The last figure compares the computation time of the $NEVM$ and Minimax normalized to the execution time consumed by the STE algorithm. We can see that as the number of VMs increases from $n = 2$ to $n = 8$, the computation time ratio of Minimax increases from almost 70% to 95 % of $NEVM$'s computation time.

Next, we evaluate the performance of our proposed heuristic PVM under small cloud clusters.

6.8.4 Performance evaluation of PVM using clusters with small numbers of VMs and servers

One major limitation of the $NEVM$ algorithm is the space complexity, i.e., $O(m^{\zeta})$, related to generating the payoff matrices G and L. This complexity makes it infeasible to test $NEVM$ with large numbers of VMs and servers. Therefore, in Fig. 6.16(a), we test the computation time performance of PVM compared to $NEVM$ and other methods with number of VMs ranging from $(m = 2)$ VMs to $(m = 8)$ VMs and with random number of servers $(n \leq m)$. We assumed that $Thrsh = UB$, i.e., we used MPL instead of MPR. Moreover, since PVM, $AvgSum$, $Prod$, PWN, and MPL returns a single solution, we used Minimax method rather than the STE that returns all possible solutions to Problem 1. We ran the simulation a 1000 times for each setting, averaged the results, and presented them in Fig. 6.16(a).

As the figure shows, the NE search time of $NEVM$ is roughly 50%, on average, better than the search time of Minimax, but it is at least a thousand times smaller than the computation time of *MPL*. We observe that the search time exponentially increases for Minimax and *MPL* algorithms with the increase of the numbers of VMs and servers, i.e., the computation times of Minimax and *MPL* increased by 18% and 93% as the number of VMs increases from 2 to 8. On the other hand, *NEVM*'s computation time grows at a much slower pace when increasing the number of VMs due to the minimization in R_V's and R_A's strategy spaces (e.g., *NEVM* only increases 10% as the number of VMs increases from 2 to 8).

It is unsurprising to notice that all the *VBP*-based heuristics have better computation times than the *NE* search approaches and a much slower computation time increase rates with the increase in the numbers of VMs and servers compared to *NEVM*, Minimax, and *MPL*. It is also worth mentioning that the computation time performance of *PVM* is the worst among *AvgSum*, *Prod*, and *PWN* because *PVM* consists of two phases in which it may have to sort and allocate VMs twice compared to other *VBP* heuristics. On the contrary, *AvgSum* has the best computation time performance compared to the *Prod* and *PWN* which execute a larger number of operations at each run compared to the *AvgSum*. Although *PWN* dynamically sorts all VMs every time it tries to allocate a new VM, the sorting criteria *PWN* uses to maximize the loss and resource capacity speeds up its allocation process compared to the offline sort and allocation of the *Prod* method [131].

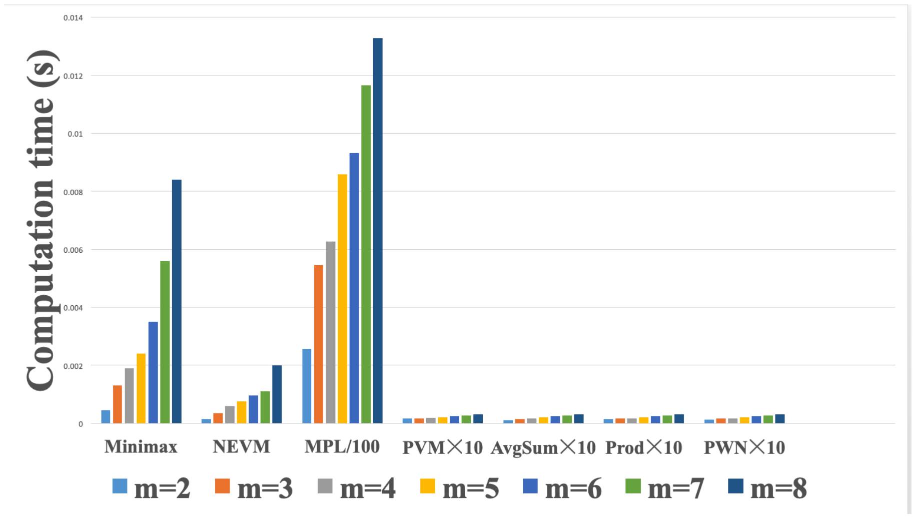
The feasibility performance of *PVM* under the same simulation setup is illustrated in Fig. 6.16(b). We see that *MPL* can always obtain the optimal results when it can complete its computation times in polynomial time. On the other hand, *NEVM* returns no pure *NE* strategy profile solutions 26% of the time, i.e., when the available resources are tight as explained in Theorem 6.2 and Theorem 6.3. Although the *VBP*-based heuristics have better feasibility performances than *NEVM*, they are not guaranteed to return optimal solutions. The worst feasibility performance is for *PWN* at 25% failure percentage in successfully allocating all VMs. Although *AvgSum* has a better average computation time than *Prod*, the last method, in fact, exceeds *AvgSum* in the allocation feasibility by 2%. The reason is that *Prod* outperforms *AvgSum* in terms of minimizing the total number of allocated servers which in turn increases the allocation feasibility [131, 133, 191]. On the other hand, having two phases of FFD bin packing based allocation allowed PVM to outperform *AvgSum*, *Prod*, and PWN by 6%, 4%, and 22%, respectively.

We evaluate the performance of PVM under medium cloud clusters next.

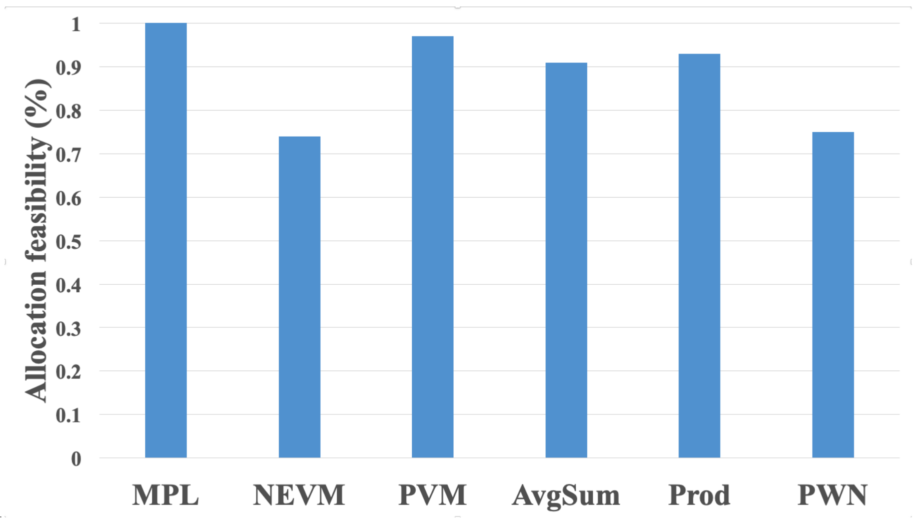
6.8.5 Performance evaluation of PVM under clusters with moderate numbers of VMs and servers

In this section, we study the performance of PVM under medium-sized cloud clusters with no more than 500 VMs and no more than 250 servers. We excluded $NEVM$ from this simulation as it is infeasible to generate its payoff matrices with such numbers of VMs and servers.

We first study the computation time performances of PVM and MPL under variable numbers of VMs and servers ranging from a $(m = 100)$ and $(n = 50)$ to $(m = 500)$ and $(n = 250)$ VMs and servers. We ran MPL and PVM 100 times for each configuration and averaged the results. As shown in Fig. 6.17, the computation time of MPL is rapidly increasing with the increasing numbers of VMs and servers. For example, the average computation time of MPL, using a 100 VMs, increases by roughly 10000 times when increasing the number of VMs to 500 VMs. On the other hand, the average computation time of PVM slowly increases when increasing the number of VMs and servers which justifies the need for using heuristics in the VM allocation problem with cybersecurity awareness.



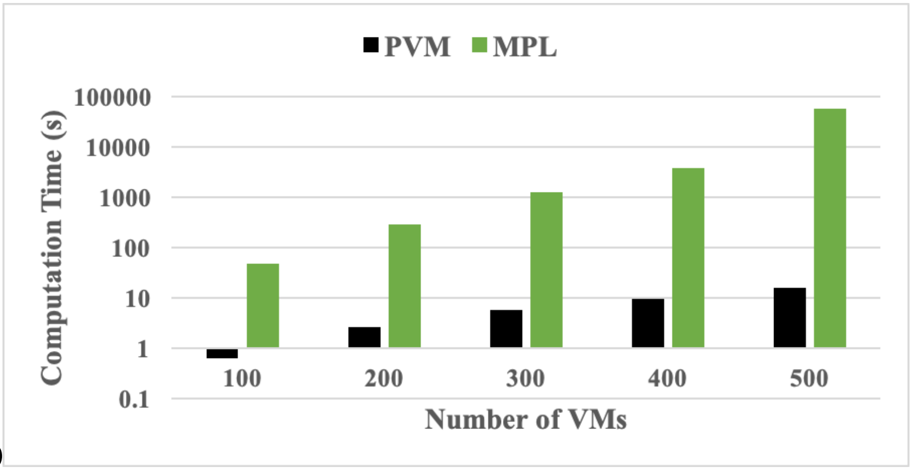
(a) Computation time performance



11

(b) Feasibility performance

Figure 6.16: Performance evaluation of PVM with small number of VMs and servers



00

Figure 6.17: Computation time performance of PVM with fair numbers of VMs and servers. Vertical axis has logarithmic units in seconds

In Fig. 6.18, we study the performance of PVM in minimizing R_V's WPC loss compared to the optimal solutions obtained by MPL. The number of VMs was randomly generated in the range [50,100]. We varied the numbers of servers from $n = 25\%$ to $n = 50\%$ of the total number of the randomly generated VMs. The server's capacity was randomly generated between 5×10^3 $MIPS$ and 25×10^3 $MIPS$. The figure shows that the worst approach in minimizing R_V's WPC loss is the $Prod$ VBP method which best minimizes the number of servers [131]. $Prod$ hence maximizes R_V's WPC loss according to Theorem 6.2. On the contrary, PVM either allocates the VMs so that an SE allocation strategy is reached under the first phase, or averages the WPC loss, given a successful attack on any VM, across all the servers which minimizes R_V's WPC loss compared to $AvgSum, Prod$, and PWN. For example, when the number of servers is bounded by 25% of the total number of VMs, PVM is 4 times worse than the optimal solution obtained by MPL, but is 0.2%, 100%, and 0.42%, better than $AvgSum, Prod$, and PWN, respectively.

Both $AvgSum$ and PWN outperform $Prod$ in minimizing R_V's WPC because they both load balance the WPC across the servers. However, PWN outperforms all other VBP

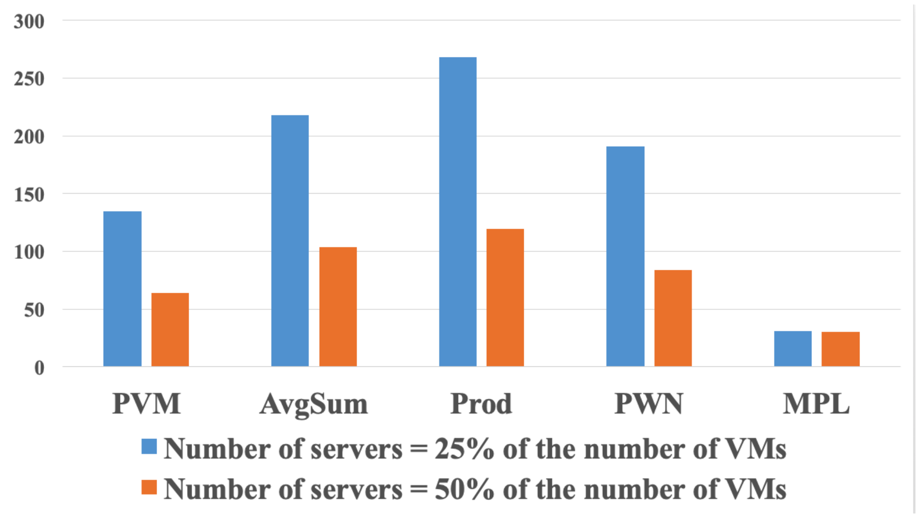
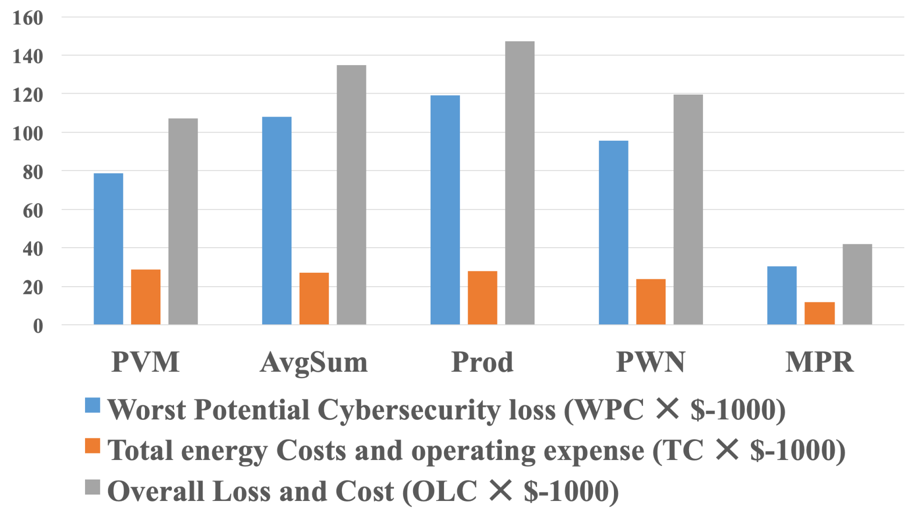


Figure 6.18: The WPC loss performance of PVM with fair numbers of VMs and servers

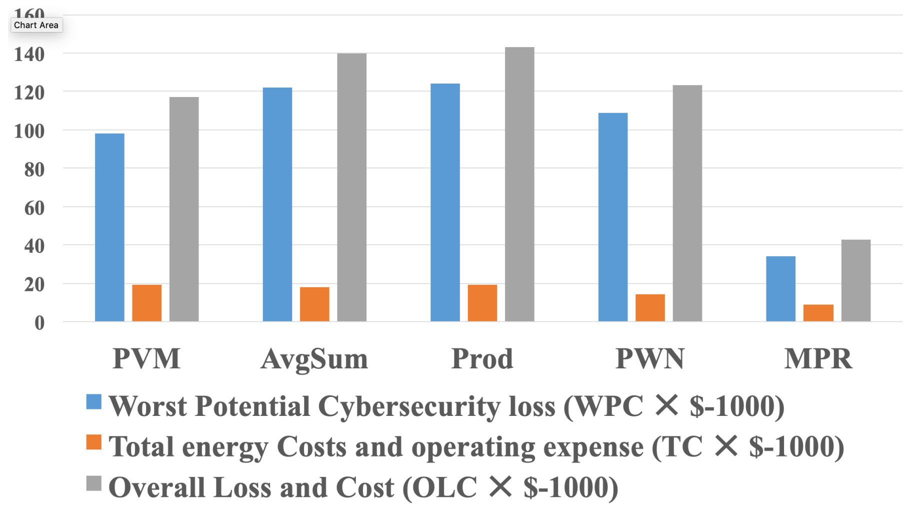
heuristics for the two-dimensional case (e.g., cybersecurity loss and resource requirements) as concluded in [133].

When the number of servers increases to 50% of the total number of the randomly generated VMs, R_V's WPC loss decreases under all methods, as stated in Theorem 6.2. Moreover, PVM is still the best after the mathematical program MPL since it increases the chances of reaching a pure NE strategy profile according to Lemma 6.4. For example, the WPC loss of PVM decreases by more than 100% when the number of servers increases twice. Nonetheless, the MPL algorithm only reduces R_V's WPC by less than 1% for the same increase in the number of servers. The reason is that, in both cases, the number of servers is relatively large enough to hosting all the primary VMs individually on separate servers which increases the chances of reaching an SE allocation, according to Lemma 6.4. Moreover, from Theorem 6.3, R_V's WPC under the first and second case is almost the same as if the resource requirements of VMs are small enough.

Fig. 6.19(a) and Fig. 6.19(b) show the performance of the PVM in optimizing resource usage, under cybersecurity constraints, compared to the optimal solutions obtained via MPR. The cybersecurity threshold was randomly generated between the lower and



(a) Server's capacity **C** =10000$ MIPS



(b) Server's capacity **C** =25000$ MIPS

Figure 6.19: The overall loss and cost performance of PVM with fair number of servers under different server's capacities

upper loss bounds (e.g., $LB \leq Thrsh \leq UB$). The number of VMs and servers were randomly generated in the ranges [50, 100] and [10%, 90%] of the total number of VMs. We varied the server's capacity from 15×10^3 *MIPS* to 25×10^3 *MIPS*, ran the simulation 1000 times for each configuration, and averaged the results.

There are several important observations one can draw from Fig. 6.19(a) and Fig. 6.19(b). Whereas PVM still outperforms other *VBP*-based heuristics in minimizing R_V's WPC loss via allocating the primary VMs to individual servers to reach a pure *NE* strategy profile, it has the worst performance among $AvgSum, Prod$, and PWN in optimizing resource usage as it underutilizes those primary servers. Nonetheless, PVM's overall performance in optimizing the WPC loss and energy and operating costs is the best since the WPC loss usually is more critical than resource usage. For example, PVM outperforms $AvgSum, Prod$, and PWN by 20%, 27%, and 10% when server's capacity is 15×10^3 *MIPS*, and by 16%, 18%, and 5% when server's capacity increases to 15×10^3 *MIPS*. Increasing the capacity of servers allows R_V to allocate more VMs to the same server, which in turn increases the WPC loss for all heuristics by almost 10% compared to only 1% increase of *MPR* (e.g., Theorem 6.2).

It is also worth mentioning that PWN illustrates the best performance in terms of minimizing R_V's total WPC loss, energy costs, and operating expense, compared to *AvgSum* and *Prod*, which conforms with conclusions about PWN performance when considering two dimensions in the FFD-based bin packing [133].

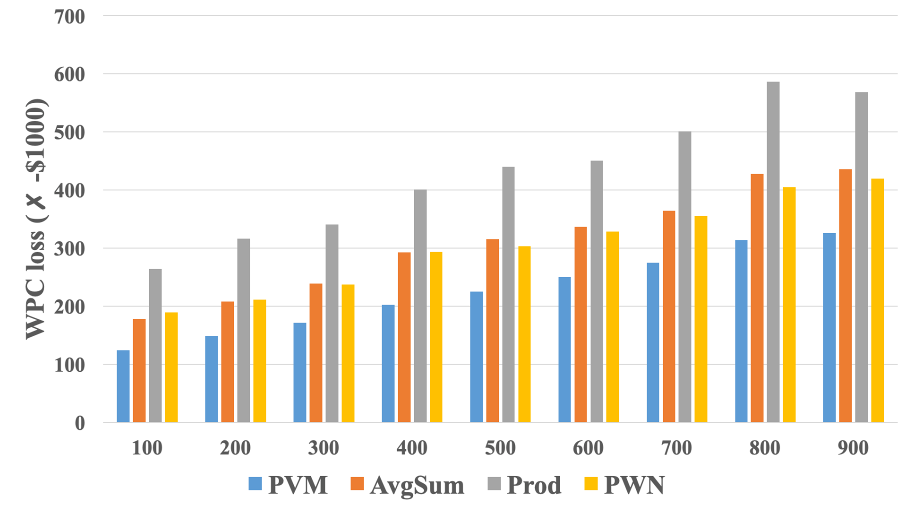
Next, we study the performance of PVM under cloud clusters with more than 500 VMs.

6.8.6 Performance evaluation of PVM under clusters with a large number of VMs

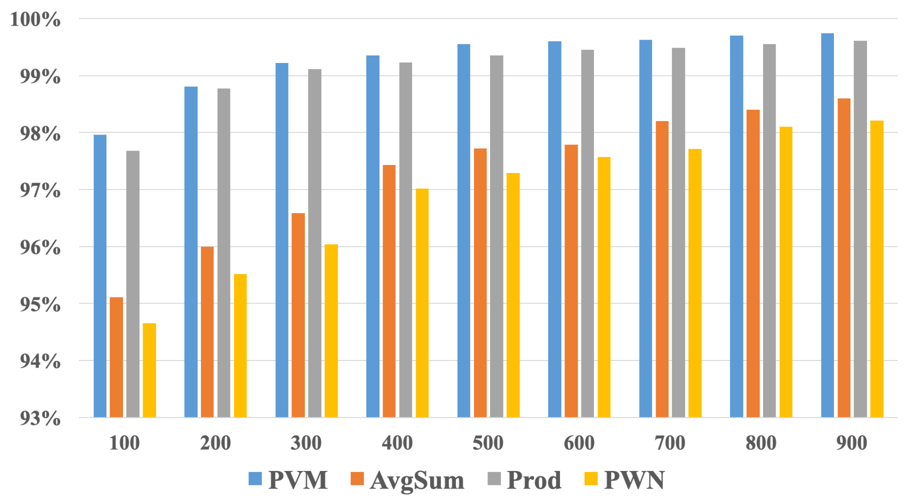
In this section, we first study the scalability, worst cybersecurity loss minimization, and feasibility performances of PVM under clusters with a number of VMs ranging from 100 to 900 VMs. The capacity of the servers was randomly generated between 5×10^3 $MIPS$ and 25×10^3 $MIPS$. The total number of servers was randomly generated between (10%) and (90%) of the total number of VMs. We repeated each run 1000 times and averaged the results.

Fig. 6.20(a) shows PVM's performance in minimizing the WPC loss of R_V compared to other VBP-based heuristics. Similar to what we observed in small and medium cloud clusters, PVM outperforms $AvgSum, Prod$, and PWN, respectively. As we stated earlier in Theorem 6.2, Fig. 6.20(a) shows that increasing the number of VMs increases R_V's WPC loss under all heuristics by at least 100 times when increasing the number of VMs from 100 to 900 VMs. Moreover, according to Lemma 6.4, PVM allocates VMs in a way that increases the chances of reaching a pure NE strategy profile to minimize R_V's WPC loss even with a large number of VMs. For example, R_V's WPC loss using PVM is, on average, $30\%, 50\%$, and 26% better than the WPC loss obtained using $AvgSum$, $Prod$ and PWN.

On the other hand, Fig. 6.20(b) shows that PVM outperforms other heuristics in terms of the allocation feasibility. We noticed that $Prod$ performance is very close to PVM (e.g., less than 1% difference). The reason is that $Prod$ uses the $FFDProd$ which performs better than other heuristics in minimizing the number of servers with a large number of VMs, as discussed in [131]. On the other hand, PWN is on average 2%-3% less than the first two methods. Further, we can see that increasing the number of VMs and servers generally improves the feasibility performance of all approaches.

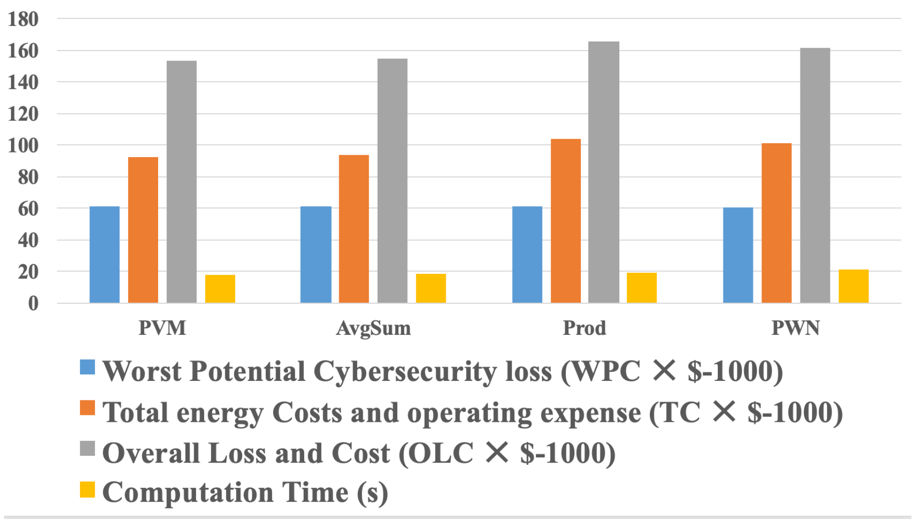


(a) The WPC loss minimization performance of PVM

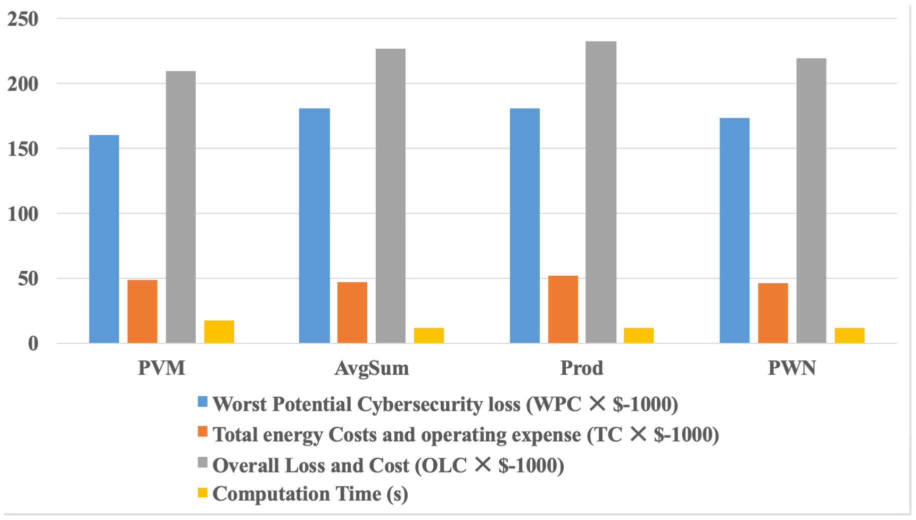


(b) Scheduling feasibility of PVM

Figure 6.20: The performance of PVM with large numbers of VMs and servers

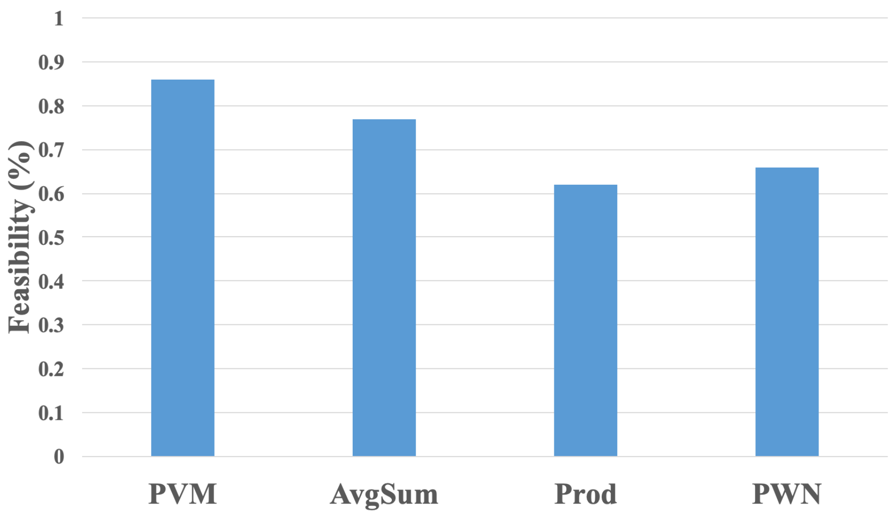


(a) Thrsh = 2× LB

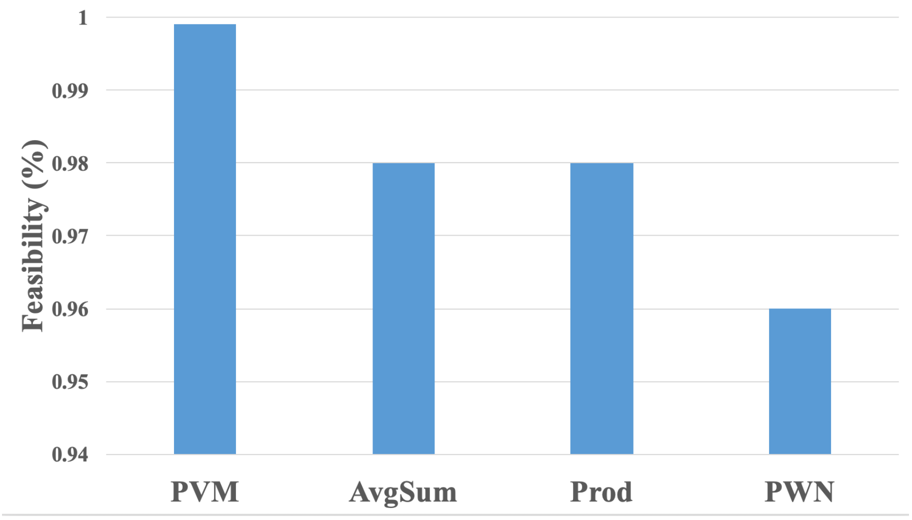


(b) Thrsh = 6× LB

Figure 6.21: The performance of PVM in optimizing resource usage under different cybersecurity thresholds



(a) Thrsh = 2× LB



(b) Thrsh = 6× LB

Figure 6.22: The feasibility performance of PVM when optimizing resource usage under different cybersecurity thresholds

Now, we move on to study the performance of PVM in maximizing the resource usage while guaranteeing that the WPC loss does not exceed a given cybersecurity loss threshold (e.g., $Thrsh$). We compared the performance of PVM to the performances of $AvgSum, Prod$, and PWN under several loss thresholds. In Fig. 6.21(a) and Fig. 6.21(b), we set the cybersecurity thresholds to $(2 \times LB)$ and $(6 \times LB)$, respectively; where $(LB = q^* \times L^*)$ is the Lower-Bound (LB) of R_V's cybersecurity loss (e.g., Equation 6.23). The server's capacity, number of servers, and number of VMs were set to 15×10^3 $MIPS$, 200 servers, and 500 VMs. We ran the simulation 1000 times for each threshold setting and averaged the results.

Fig. 6.21(a) shows that when $Thrsh = 2 \times LB$, $PVM, AvgSum$, and $Prod$ roughly have the same WPC loss, whereas PWN outperforms all the rest in less than 1%. On the other hand, PVM outperforms $AvgSum, Prod$, and PWN in minimizing R_V's overall loss and cost by $2\%, 12\%$, and 9%, respectively. Moreover, PVM's computation time is less than $AvgSum, Prod$, and PWN by $1, 2$, and 4 seconds. The reason is that the PVM allocates all primary VMs within the first allocation phase which minimizes the allocation times for the rest of the VM under a very strict cybersecurity threshold and improves its allocation feasibility. For example, Fig. 6.22(a) shows that when $Thrsh = 2 \times LB$, PVM allocation feasibility is $9\%, 24\%$, and 20% more than the feasibilities of $AvgSum, Prod$, and PWN.

Although increasing the cybersecurity threshold from $Thrsh = 2 \times LB$ to $Thrsh = 6 \times LB$ (e.g., Fig. 6.22(b)), improves the allocation feasibility of all approaches, PVM still outperforms other approaches by at least 1% in the allocation feasibility. It is also expected that PVM's computation time performance is almost the same under different cybersecurity threshold because PVM is bounded by the potential cybersecurity loss given an attack on any primary VM (e.g., V°; where $q^\circ \times \ell^\circ = LB < Thrsh$). Nonetheless, the computation times of other approaches improve significantly, outperforming the computation time of PVM by 6 seconds on average, when the cybersecurity threshold increases from $Thrsh = 2 \times LB$ to $Thrsh = 6 \times LB$.

Moreover, based on Theorem 6.2, the increase in the cybersecurity threshold causes all methods to increase their WPC loss to maximize resource utilization and minimize R_V's overall loss and cost by almost 50%, compared to the first case in which $Thrsh = 2 \times LB$. Although increasing the cybersecurity threshold improves R_V's energy and operating costs, it causes R_V's overall loss and cost to increase by $27\%, 32\%, 29\%$, and 0.26% under $PVM, AvgSum, Prod$, and PWN due to the decrease in the number of the allocated servers (e.g., Theorem 6.2).

6.9 Summary

In this chapter, we address the problems of static VM allocation with resource and cybersecurity loss constraints that are well-known NP-hard problems. We first formulate them as quadratic programming problems that are solvable only for small size cloud clusters. We then model the problems as a two-player zero-sum game. We showed that *NE* search approaches (e.g., Minimax and *STE*) can be applied to solve our VM allocation problems. For example, they take polynomial time concerning a given size of the strategy spaces. Unfortunately, the computation complexities of those *NE* search approaches become non-polynomial as the strategy space of the provider grows exponentially with the increase of the numbers of VMs and servers. This motivates us to study the characteristics of the pure *NE* strategy profile and, based on which, to develop computationally efficient heuristics. Intensive simulation results show that our proposed allocation algorithms outperform the commonly used *VBP* heuristics in solving similar allocation problems. Our future work aims to investigate dynamic allocation strategies based on mixed-strategy *NE* strategy profiles considering several optimization criteria, such as performance, cybersecurity, and costs.

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this chapter, we summarize the contributions that we present in this dissertation and discuss the possible directions for our future research work.

7.1 Summary

As adopting cloud computing has recently become the dominant trend for continuously delivering online services for individual and organizations over the internet, successful and well-planned resource management policies supporting those services became a critical component for any agile, consolidated and dynamically scalable cloud data center that provides valuable and high-quality cloud services with efficient power consumption.

In this dissertation, we study the research problem of developing efficient and effective cloud resource management policies and techniques. Our goal is to help cloud service providers optimize the trade-offs among several performance and cybersecurity criteria.

1. To facilitate our experiment and validation work, we first develop a cloud computing prototype that closely mirrors industry-compatible cloud platforms. Our prototype can provide cloud services as any middle-sized cloud service provider does. The prototype generates and runs several general and cloud-specific benchmarks under an isolated and well-controlled environment. We further incorporate new workload scheduling, resource provisioning, and performance monitoring schemes that we proposed in this dissertation into the platform.
2. Different from previous studies that employ separate VMs for hosting requests with different *QoS* requirements, we develop a cloud service multiplexing method, based on the queuing model with reneging, that enables requests of the same service type,

but with different *QoS* constraints, to share the same VM. To our best knowledge, this is the first approach by which different requests with different *QoS* requirements can be hosted on a single node to increase resource utilization. We also devise a novel methodology that correctly discards potential failure requests as soon as possible to minimize processing rate demands, and to reduce total power consumption with statistically guaranteed *QoS*. We introduce a packing and consolidation algorithm that statistically ensures the *QoS* requirements of cloud requests in terms of deadline miss ratios. In addition to the analytical validation of our proposed methods, we experimentally verify them, under general and cloud-specific workloads, using our cloud platform. For example, we use the Data Caching benchmark that emulates the behavior of a Twitter caching server and assumes strict quality of service guarantees, such as, 95% of the request must finish within 200 ms. Extensive experimentation results show that our proposed methods widely outperform existing approaches in terms of *QoS* satisfaction, power consumption efficiency, resource demand minimization, and electricity cost saving.

3. A major limitation of consolidating VMs of different security requirements onto a single server is that it can result in VM-to-VM interdependent cybersecurity *IC* risks. For example, the odds of successfully compromising a secure critical VM are high when an attacker compromises the hosting hypervisor after a successful direct attack on one of its less secured, non-critical VMs. Therefore, we formulate the allocation problem with cybersecurity awareness into a non-cooperative, zero-sum theoretical game model between an attacker and the service cloud provider. We develop a set of new conditions to identify the existence of an equilibrium allocation strategy quickly. We mean by an –equilibrium– allocation strategy that the allocation policy in which neither the provider nor the attacker can benefit from unitarily deviating from their allocation or attack decisions, respectively. We then

incorporate several resource usage parameters into a non-zero-sum game model. We identify the cases under which several static and dynamic equilibrium allocation strategies exist. We also derive the lower-bound and upper-bound of the *IC* risks.

4. Finally, we extend our game models to include VMs with more general cybersecurity and resource requirements. We focus on the static VM allocation problem to study how to (1) minimize the provider's worst potential cybersecurity loss under constrained resource usage and how to (2) optimize cloud resource usage while ensuring that the worst potential cybersecurity loss is always less than a given cybersecurity threshold. We show later in this dissertation that a constrained-allocation problem is a typical NP-hard problem, and, thus, we formulate the security-constrained and resource-constrained VM allocation problems using the Mathematical Programming (*MP*) approach to obtain the optimal solutions, which will be used as a comparison baseline against other proposed approaches in this dissertation and when the problem size is small. We formally model the resource and security-constrained allocation problems as a non-cooperative two-player zero-sum game. We conduct a thorough analysis of the characteristics of the pure Nash Equilibrium (*NE*) strategy profiles in our game model, which we formulate as a series of lemmas and theorems. Based on the insights of our analysis, we develop several effective and computationally efficient algorithms to allocate VMs, of different resource and security requirements, with resource usage and security loss optimized. We have implemented our algorithms and studied their efficiency and effectiveness. Our extensive simulation results show that our novel approaches are good trade-offs when compared with the computational-intensive approaches, such as the ones based on the *MP* approaches, the existing *NE* search methods, or the computationally efficient multi-dimensional bin-packing methods.

7.2 Future work

In this dissertation, we primarily focus on developing static resource allocation mechanisms for time-sensitive cloud services and VMs under resource, quality, and cybersecurity constraints.Static resource allocation methods simplify the process of cloud resource management. The study of static VM allocation not only has solid theoretical values but is also useful and applicable in many practical scenarios. For instance, static VM allocation does not cause service interruption for cloud services and hosting VMs due to the VM live migration, and, hence, it can provide time-related *QoS* guarantees. Furthermore, static allocation approaches are ideal for batch cloud workload during each epoch, because batch cloud workloads are completed detailed and characterized by predictable performance and intensive computation time.

Although static VM allocation has many unique advantages (e.g., low overhead and reliable predictability), dynamic VM allocation methods are more flexible, adaptive, and potentially more effective in a highly dynamic environment in terms of optimizing cloud resource usage and performance. Therefore, dynamic resource allocation with performance and cybersecurity awareness will be the focus of our future research following the work presented in this dissertation.

Specifically, in our future work, we plan to focus on the following problems.

7.3 Dynamic allocation strategies with QoS guarantee in heterogeneous cloud platforms

We can extend our set of static scheduling mechanisms, proposed in Chapter 4, for cloud services with guaranteed *QoS* to include allocation strategies based on VM live migration.

Dynamic VM allocation maximizes resource utilization [92] and minimizes cybersecurity risks resulted from colocation attacks [51].

However, VM migration comes with a migration overhead that results in service interruption, performance degradation, and consequently Service Level Agreements *SLAs* violations [192]. The first challenge we have to address when adopting dynamic cloud resource allocation methodologies is how to timely and efficiently adapt to the fluctuating cloud workloads while ensuring a guaranteed *QoS* for cloud users. It is consequently crucial to study the timing and spatial characteristics of cloud workload to improve the predictability of the dynamically changing cloud environments.

What makes dynamic allocation problems with guaranteed *QoS* even more complicated problem is that current cloud data centers host a large number of service types with several resources, timing, and cybersecurity requirements. Moreover, those data centers offer different implementation requirements and styles (e.g., varying degrees of parallelism, as in Google data centers [82, 193].) For energy-efficient computing, different service types are allocated to servers with different architectures, i.e., heterogeneous servers [194–197]. We also intend to extend our work in Chapter 4 to include time-sensitive cloud services hosted on heterogeneous platforms with different power requirements, cybersecurity levels, and resource usage criteria (e.g., CPU, memory, storage, and network) while ensuring the *QoS* of those cloud services. In other words, we are interested in identifying the minimum required multi-dimensional resource demand vector for each service type when hosted alone and when hosted with different service types onto heterogeneous servers with *QoS* constraints and cybersecurity requirements. We are also interested in studying how to allocate cloud services with guaranteed *QoS* requirements and cybersecurity conditions onto heterogeneous servers with power consumption and operating expense constraints.

7.4 Cloud workload modeling and analysis

Modeling and analyzing cloud workloads are vital when developing efficient and effective dynamic resource allocation policies that optimize resource usage while assuring an adequate quality level for cloud users.

Although both industry and academia have developed several benchmarks and simulation tools to mimic actual cloud workloads [48,76,163], those tools are limited to specific application or cloud platform types, such as EC2 low-level workloads [198] and scientific workloads [199]. The limitation and inadequacy of workload studies and models are due to the underlying complexity of cloud systems [200], variety in types of workloads [111], and need for more cloud traces to be analyzed and studied [82, 193].

Studying the timing and spatial features of cloud workloads can help scholars predict the timing performance and resource demands of cloud requests under different stress levels and cloud platforms. We hope that our studies help accurately identify the factors that contribute to optimize and degrade the performance of cloud service, and thereupon provide vital insight to developing efficient and effective dynamic cloud resource allocations [166].

7.5 Game-theoretical based VM live migration in cloud clusters with cybersecurity awareness

One major limitation of any server consolidation algorithm is the interdependent cybersecurity risks. In Chapter 5 and Chapter 6, we proposed game-theoretical based static VM allocation strategies to minimize the provider's worst potential cybersecurity loss while optimizing the cloud provider's power consumption and operating costs. In the future, we plan to consider the dynamic allocation case based on the mixed NE strategy profiles, in where VMs can migrate from one server to another to minimize the chances of a successful attack on any of the VMs. We assume that a successful attack requires more time than the time between two successive migrations. The challenging question, nonetheless, is – what is the minimum allowed number of migrations within a certain period and between predefined or random allocation strategies and what are the optimal time intervals between every two consecutive migrations so that the cybersecurity risks, power consummation, and operating expense are minimized?–. Meanwhile, we also should guarantee that the performance degradation and service interruption due to VM the live migration are mitigated.

BIBLIOGRAPHY

[1] "Slideplayer," https://slideplayer.com.

[2] "Smechannel," https://smechannels.com.

[3] "Wordpress," https://wordpress.com.

[4] "Forbes," https://www.forbes.com.

[5] "Worldwide cloud it infrastructure revenues continue to grow by double digits in the first quarter of 2018 as public cloud expands, according to idc," 2018.

[6] "Cloud deployment materials i: Mirantis," http://www.mirantis.com/.

[7] "Cloud deployment materials ii: Rackspace," http://www.rackspace.com/.

[8] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[9] A. Weiss, "Computing in the clouds," *networker*, vol. 11, no. 4, pp. 16–25, 2007.

[10] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

[11] D. A. Patterson, "The data center is the computer," *Communications of the ACM*, vol. 51, no. 1, pp. 105–105, 2008.

[12] D. Schneider and Q. Hardy, "Under the hood at google and facebook," *Spectrum, IEEE*, vol. 48, no. 6, pp. 63–67, June 2011.

[13] P. Banerjee, R. Friedrich, C. Bash, P. Goldsack, B. Huberman, J. Manley, C. Patel, P. Ranganathan, and A. Veitch, "Everything as a service: Powering the new information economy," *Computer*, no. 3, pp. 36–43, 2011.

[14] Y. Duan, "Value modeling and calculation for everything as a service (xaas) based on reuse," in *Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD), 2012 13th ACIS International Conference on*. IEEE, 2012, pp. 162–167.

[15] J. Spillner and A. Schill, “A versatile and scalable everything-as-a-service registry and discovery.” in *CLOSER*, 2013, pp. 175–183.

[16] G. Li and M. Wei, “Everything-as-a-service platform for on-demand virtual enterprises,” *Information Systems Frontiers*, vol. 16, no. 3, pp. 435–452, 2014.

[17] I. Foster, Y. Zhao, I. Raicu, and S. Lu, “Cloud computing and grid computing 360-degree compared,” in *Grid Computing Environments Workshop, 2008. GCE’08.* Ieee, 2008, pp. 1–10.

[18] P. Mell, T. Grance *et al.*, “The nist definition of cloud computing,” 2011.

[19] M. Carroll, A. Van Der Merwe, and P. Kotze, “Secure cloud computing: Benefits, risks and controls,” in *Information Security South Africa (ISSA), 2011.* IEEE, 2011, pp. 1–9.

[20] M. Zivkovic, J. Bosman, J. Van den Berg, R. D. van der Mei, H. B. Meeuwissen, and R. Nunez-Queija, “Dynamic profit optimization of composite web services with slas,” in *2011 IEEE Global Telecommunications Conference-GLOBECOM 2011.* IEEE, 2011, pp. 1–6.

[21] “Pocket gems on google cloud platform,” http://cloud.google.com/customers/pocketgems/.

[22] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, “A performance analysis of ec2 cloud computing services for scientific computing,” in *International Conference on Cloud Computing.* Springer, 2009, pp. 115–131.

[23] “Sony music on google cloud platform,” http://cloud.google.com/customers/sony-music/.

[24] W. Zhao, D. Olshefski, and H. G. Schulzrinne, “Internet quality of service: An overview,” 2000.

[25] M. Dodani, “Cloud architecture.” *Journal of Object Technology*, vol. 8, no. 7, pp. 35–44, 2009.

[26] L. Tomás and J. Tordsson, “Improving cloud infrastructure utilization through overbooking,” in *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, 2013, p. 5.

[27] P. Ghumre, J. Li, M. Kesavan, A. Gavrilovska, and K. Schwan, "Evaluating the need for complexity in energy-aware management for cloud platforms," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 3, pp. 23–27, 2012.

[28] A. Beloglazov, R. Buyya, Y. C. Lee, A. Zomaya *et al.*, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances in computers*, vol. 82, no. 2, pp. 47–111, 2011.

[29] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 6, pp. 1107–1117, 2013.

[30] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM computer communication review*, vol. 39, no. 1, pp. 68–73, 2008.

[31] *Cloud Computing, Server Utilization, & the Environment*. [Online]. Available: https://aws.amazon.com/blogs/aws/cloud-computing-server-utilization-the-environment/

[32] I. Kumara, H. Jun, A. Colman, and M. Kapuruge, "Software-defined service networking: Performance differentiation in shared multi-tenant cloud applications," *IEEE Transactions on Services Computing*, 2016.

[33] C. A. Kamhoua, L. Kwiat, K. A. Kwiat, J. S. Park, M. Zhao, and M. Rodriguez, "Game theoretic modeling of security and interdependency in a public cloud," in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*. IEEE, 2014, p. 514–521.

[34] E. Knorr and G. Gruman, "What cloud computing really means," *InfoWorld*, vol. 7, pp. 20–20, 2008.

[35] T. Wang, G. Quan, S. Ren, and M. Qiu, "Topology virtualization for throughput maximization on many-core platforms," in *Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on*. IEEE, 2012, pp. 408–415.

[36] G. Perry, "How cloud & utility computing are different." *GIGAOM*, 2008.

[37] "Amazon aws," https://aws.amazon.com/s3/.

[38] I. Foster, "What is the grid?-a three point checklist," *GRIDtoday*, vol. 1, no. 6, 2002.

[39] V. Kundra, "25 point implementation plan to reform federal information technology management," EXECUTIVE OFFICE OF THE PRESIDENT WASHINGTON DC/OFFICE OF MANAGEMENT AND BUDGET OFFICE OF E-GOVERNMENT AND INFORMATION TECHNOLOGY, Tech. Rep., 2010.

[40] S. Moss, "Under trump, us federal data center consolidation progress remains unclear, 24th march 2017," *Datacenter Dynamics, The Business of Data Centers*, 2017.

[41] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke, "Smart grid technologies: Communication technologies and standards," *IEEE transactions on Industrial informatics*, vol. 7, no. 4, pp. 529–539, 2011.

[42] I. Berger, "Keeping cloud computing's prospects safe and sunny," 2010.

[43] K. McCabe and R. Nachbar, "Survey by ieee and cloud security alliance details importance and urgency of cloud computing security standards," 2010.

[44] C. Aradau, "Security that matters: Critical infrastructure and objects of protection," *Security Dialogue*, vol. 41, no. 5, pp. 491–514, 2010.

[45] M. Poess and R. O. Nambiar, "Energy cost, the key challenge of today's data centers: a power consumption analysis of tpc-c results," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1229–1240, 2008.

[46] Greenpeace, "How dirty is your data?" *a look at the energy choices that power cloud computing*, 2011.

[47] Q. Zhang and W. Shi, "Energy-efficient workload placement in enterprise datacenters," *Computer*, vol. 49, no. 2, pp. 46–52, 2016.

[48] G. Kousiouris, T. Cucinotta, and T. Varvarigou, "The effects of scheduling, workload type and consolidation scenarios on virtual machine performance and their prediction through optimized artificial neural networks," *Journal of Systems and Software*, vol. 84, no. 8, pp. 1270–1291, 2011.

[49] J. Son and R. Buyya, "Sla-aware and energy-efficient dynamic overbooking in sdn-enabled cloud data centers," in *4TH ANNUAL DOCTORAL COLLOQUIUM*, 2016, p. 43.

[50] C. Cai, L. Wang, S. U. Khan, and J. Tao, "Energy-aware high performance computing: A taxonomy study," in *IEEE 17th Int. (ICPADS)*, 2011, pp. 953–958.

[51] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-vm side channels and their use to extract private keys," pp. 305–316, 2012.

[52] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," pp. 199–212, 2009.

[53] Y. Zhang, A. Juels, A. Oprea, and M. K. Reiter, "Homealone: Co-residency detection in the cloud via side-channel analysis," in *Security and Privacy (SP), 2011 IEEE Symposium on*. IEEE, 2011, pp. 313–328.

[54] S. M. Parikh, N. M. Patel, and H. B. Prajapati, "Resource management in cloud computing: classification and taxonomy," *arXiv preprint arXiv:1703.00374*, 2017.

[55] L. Kwiat, C. A. Kamhoua, K. A. Kwiat, J. Tang, and A. Martin, "Security-aware virtual machine allocation in the cloud: A game theoretic approach," in *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*. IEEE, 2015, p. 556–563.

[56] D. Carrera, M. Steinder, I. Whalley, J. Torres, and E. Ayguadé, "Managing slas of heterogeneous workloads using dynamic application placement," in *Proceedings of the 17th international symposium on High performance distributed computing*. ACM, 2008, pp. 217–218.

[57] G. von Laszewski, L. Wang, A. J. Younge, and X. He, "Power-aware scheduling of virtual machines in dvfs-enabled clusters." in *CLUSTER*. IEEE, 2009, pp. 1–10.

[58] S. Wang, W. Munawar, J. Liu, J.-J. Chen, and X. Liu, "Power-saving design for server farms with response time percentile guarantees." in *IEEE Real-Time and Embedded Technology and Applications Symposium*, M. D. Natale, Ed., 2012, pp. 273–284.

[59] P. Lama and X. Zhou, "Efficient server provisioning with end-to-end delay guarantee on multi-tier clusters." in *IWQoS*. IEEE, 2009, pp. 1–9.

[60] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via vm multiplexing," in *Proceedings of the 7th international conference on Autonomic computing*. ACM, 2010, pp. 11–20.

[61] F. Caglar and A. Gokhale, "ioverbook: Intelligent resource-overbooking to support soft real-time applications in the cloud," in *CLOUD, IEEE 7th Int. Conference on*, 2014, pp. 538–545.

[62] K. C. Almeroth, A. Dan, D. Sitaram, and W. B. Tetzlaff, "Long term resource allocation in video delivery systems," in *INFOCOM'97. 16th Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution., Proceedings IEEE*, vol. 3, 1997, pp. 1333–1340.

[63] K. G. Shin and P. Ramanathan, "Real-time computing: A new discipline of computer science and engineering," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 6–24, 1994.

[64] L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real time scheduling theory: A historical perspective," *Real-time systems*, vol. 28, no. 2-3, pp. 101–155, 2004.

[65] S. T. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 702–710.

[66] M. Fan, "Real-time systems," in *Real-time Systems*. InTech, 2016.

[67] H.-E. Zahaf, A. E. H. Benyamina, R. Olejnik, and G. Lipari, "Energy-efficient scheduling for moldable real-time tasks on heterogeneous computing platforms," *Journal of Systems Architecture*, vol. 74, pp. 46–60, 2017.

[68] I. Hwang, T. Kam, and M. Pedram, "A study of the effectiveness of cpu consolidation in a virtualized multi-core server system," in *Proceedings of the ACM/IEEE int. symposium on Low power electronics and design*, 2012, pp. 339–344.

[69] V. Petrucci, M. A. Laurenzano, J. Doherty, Y. Zhang, D. Mosse, J. Mars, and L. Tang, "Octopus-man: Qos-driven task management for heterogeneous multicores in warehouse-scale computers," in *HPCA, IEEE 21st Int. Symposium on*, 2015, pp. 246–258.

[70] A. Sarkar, F. Mueller, and H. Ramaprasad, "Static task partitioning for locked caches in multicore real-time systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 14, no. 1, p. 4, 2015.

[71] C.-W. Chang, J.-J. Chen, T.-W. Kuo, and H. Falk, "Real-time task scheduling on island-based multi-core platforms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 2, pp. 538–550, 2015.

[72] E. D. Jensen, C. D. Locke, and H. Tokuda, "A time-driven scheduling model for real-time operating systems." in *RTSS*, vol. 85, 1985, pp. 112–122.

[73] P. Li, H. Wu, B. Ravindran, and E. D. Jensen, "A utility accrual scheduling algorithm for real-time activities with mutual exclusion resource constraints," *IEEE Transactions on Computers*, vol. 55, no. 4, pp. 454–469, 2006.

[74] W. Li, K. Kavi, and R. Akl, "A non-preemptive scheduling algorithm for soft real-time systems," *Computers & Electrical Engineering*, vol. 33, no. 1, pp. 12–29, 2007.

[75] F. Zhang, J. Cao, K. Li, S. U. Khan, and K. Hwang, "Multi-objective scheduling of many tasks in cloud platforms," *Future Generation Computer Systems*, vol. 37, pp. 309–320, 2014.

[76] Z. Liu and S. Cho, "Characterizing machines and workloads on a google cluster," in *Parallel Processing Workshops (ICPPW), 2012 41st International Conference on*. IEEE, 2012, pp. 397–403.

[77] Y. Song, H. Wang, Y. Li, B. Feng, and Y. Sun, "Multi-tiered on-demand resource scheduling for vm-based data center," in *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*. IEEE, 2009, pp. 148–155.

[78] M. Chen, H. Zhang, Y.-Y. Su, X. Wang, G. Jiang, and K. Yoshihira, "Effective vm sizing in virtualized data centers." in *Integrated Network Management*. Citeseer, 2011, pp. 594–601.

[79] S. Ren, Y. He, and F. Xu, "Provably-efficient job scheduling for energy and fairness in geographically distributed data centers," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*. IEEE, 2012, pp. 22–31.

[80] H. Ludwig, A. Keller, A. Dan, R. P. King, and R. Franck, "Web service level agreement (wsla) language specification," *Ibm corporation*, pp. 815–824, 2003.

[81] P. Ferguson and G. Huston, *Quality of service: delivering QoS on the Internet and in corporate networks*. John Wiley & Sons, Inc., 1998.

[82] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., Mountain View, CA, USA, Technical Report*, 2011.

[83] P. Lama and X. Zhou, "Efficient server provisioning with end-to-end delay guarantee on multi-tier clusters," in *Quality of Service, 2009. IWQoS. 17th International Workshop on*. IEEE, 2009, pp. 1–9.

[84] R. Sethi, "Exoskeleton: Fast cache-enabled load balancing for key-value stores," 2015.

[85] R. Bane, B. Annappa, and K. Shet, "Survey of dynamic resource management approaches in virtualized data centers," in *Computational Intelligence and Computing Research (ICCIC), IEEE Int. Conference on*, 2013, pp. 1–7.

[86] A. Verma, P. Ahuja, and A. Neogi, "pmapper: power and migration cost aware application placement in virtualized systems," in *Middleware*. Springer, 2008, pp. 243–264.

[87] M. A. Awan, P. M. Yomsi, G. Nelissen, and S. M. Petters, "Energy-aware task mapping onto heterogeneous platforms using DVFS and sleep states," *Real-Time Systems*, vol. 52, no. 4, pp. 450–485, 2016.

[88] D. Page, "Defending against cache-based side-channel attacks," *Information Security Technical Report*, vol. 8, no. 1, pp. 30–44, 2003.

[89] Z. Wang and R. B. Lee, "New cache designs for thwarting software cache-based side channel attacks," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 494–505, 2007.

[90] Y. Zhang, M. Li, K. Bai, M. Yu, and W. Zang, "Incentive compatible moving target defense against vm-colocation attacks in clouds," in *IFIP International Information Security Conference*. Springer, 2012, pp. 388–399.

[91] B. Danev, R. J. Masti, G. O. Karame, and S. Capkun, "Enabling secure vm-vtpm migration in private clouds," in *Proceedings of the 27th Annual Computer Security Applications Conference*. ACM, 2011, pp. 187–196.

[92] T. Wood, P. J. Shenoy, A. Venkataramani, M. S. Yousif *et al.*, "Black-box and gray-box strategies for virtual machine migration." in *NSDI*, vol. 7, 2007, pp. 17–17.

[93] S. Siewert, *Real-time embedded components and systems.* Cengage Learning, 2006.

[94] H. Kopetz, *Real-time systems: design principles for distributed embedded applications.* Springer Science & Business Media, 2011.

[95] S. Baruah and K. Pruhs, "Open problems in real-time scheduling," *Journal of Scheduling*, vol. 13, no. 6, pp. 577–582, 2010.

[96] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[97] A. K.-L. Mok, "Fundamental design problems of distributed systems for the hard-real-time environment," Ph.D. dissertation, Massachusetts Institute of Technology, 1983.

[98] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time cpu scheduling for mobile multimedia systems," in *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5. ACM, 2003, pp. 149–163.

[99] Y. Wang and M. Saksena, "Scheduling fixed-priority tasks with preemption threshold," in *Real-Time Computing Systems and Applications, 1999. RTCSA'99. Sixth International Conference on.* IEEE, 1999, pp. 328–335.

[100] H. Oh and S. Ha, "A static scheduling heuristic for heterogeneous processors," in *European Conference on Parallel Processing.* Springer, 1996, pp. 573–577.

[101] G. Singla, G. Kaur, A. K. Unver, and U. Y. Ogras, "Predictive dynamic thermal and power management for heterogeneous mobile platforms," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015.* IEEE, 2015, pp. 960–965.

[102] J. M. Rivas, J. J. Gutiérrez, and M. G. Harbour, "Fixed priorities or EDF for distributed real-time systems?" *ACM SIGBED Review*, vol. 10, no. 2, pp. 21–21, 2013.

[103] G. C. Buttazzo, "Rate monotonic vs. EDF: judgment day," *Real-Time Systems*, vol. 29, no. 1, pp. 5–26, 2005.

[104] S. K. Baruah, "The non-preemptive scheduling of periodic tasks upon multiprocessors," *Real-Time Systems*, vol. 32, no. 1-2, pp. 9–20, 2006.

[105] X. Wang, Z. Li, and W. M. Wonham, "Optimal priority-free conditionally-preemptive real-time scheduling of periodic tasks based on des supervisory control," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 7, pp. 1082–1098, 2017.

[106] X. Wang, Z. Li, and W. Wonham, "Priority-free conditionally-preemptive scheduling of modular sporadic real-time systems," *Automatica*, vol. 89, pp. 392–397, 2018.

[107] G. C. Buttazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications*. Springer Science & Business Media, 2011, vol. 24.

[108] J. A. Stankovic, "Misconceptions about real-time computing: A serious problem for next-generation systems," *Computer*, vol. 21, no. 10, pp. 10–19, 1988.

[109] H. Chen, X. Zhu, H. Guo, J. Zhu, X. Qin, and J. Wu, "Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment," *Journal of Systems and Software*, vol. 99, pp. 20–35, 2015.

[110] K. H. Kim, A. Beloglazov, and R. Buyya, "Power-aware provisioning of cloud resources for real-time services," in *Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science*. ACM, 2009, p. 1.

[111] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *Workload Characterization, 2007. IISWC 2007. IEEE 10th Int. Symposium on*, Sept 2007, pp. 171–180.

[112] I. S. Moreno and J. Xu, "Neural network-based overallocation for improved energy-efficiency in real-time cloud environments," in *Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2012 IEEE 15th International Symposium on*. IEEE, 2012, pp. 119–126.

[113] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, "Server workload analysis for power minimization using consolidation," in *Proceedings of the Conference on USENIX Annual Technical Conference*, ser. USENIX'09, Berkeley, CA, USA, 2009, pp. 28–28.

[114] K. H. Kim, R. Buyya, and J. Kim, “Power aware scheduling of bag-of-tasks applications with deadline constraints on dvs-enabled clusters.” in *CCGRID*, vol. 7, 2007, pp. 541–548.

[115] M. Poess and R. O. Nambiar, “Energy cost, the key challenge of today’s data centers: a power consumption analysis of tpc-c results,” *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1229–1240, 2008.

[116] A. Beloglazov and R. Buyya, “Energy efficient resource management in virtualized cloud data centers.” in *CCGRID*. IEEE, 2010, pp. 826–831.

[117] S. Liu, S. Homsi, M. Fan, S. Ren, G. Quan, and S. Ren, “Scheduling time-sensitive multi-tier services with probabilistic performance guarantee,” in *2014 20th IEEE (ICPADS)*. IEEE, 2014, pp. 736–743.

[118] I. Ahmad and S. Ranka, *Handbook of Energy-Aware and Green Computing-Two Volume Set*. CRC Press, 2016.

[119] A. Hammadi and L. Mhamdi, “A survey on architectures and energy efficiency in data center networks,” *Computer Communications*, vol. 40, pp. 1–21, 2014.

[120] C. Mastroianni, M. Meo, and G. Papuzzo, “Probabilistic consolidation of virtual machines in self-organizing cloud data centers,” *IEEE T. Cloud Computing*, vol. 1, no. 2, pp. 215–228, 2013.

[121] F. Salfner, P. Tröger, and A. Polze, “Downtime analysis of virtual machine live migration,” in *DEPEND*, 2011, pp. 100–105.

[122] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, “Managing energy and server resources in hosting centers,” *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5, pp. 103–116, 2001.

[123] J. Liddle, “Amazon found every 100ms of latency cost them 1% in sales,” *The GigaSpaces*, vol. 27, 2008.

[124] F. Caglar, S. Shekhar, and A. Gokhale, “A performance interferenceaware virtual machine placement strategy for supporting soft realtime applications in the cloud,” *Institute for Software Integrated Systems, Vanderbilt University, TN, USA, Tech. Rep. ISIS-2013-105*.

[125] Y. C. Lee, C. Wang, A. Y. Zomaya, and B. B. Zhou, "Profit-driven service request scheduling in clouds," in *Proceedings of the 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing*. IEEE Computer Society, 2010, pp. 15–24.

[126] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[127] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong, "A game-theoretic method of fair resource allocation for cloud computing services," *The journal of supercomputing*, vol. 54, no. 2, p. 252–269, 2010.

[128] J. Xu and J. A. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*. IEEE Computer Society, 2010, pp. 179–188.

[129] J. Kunsemoller and H. Karl, *A Game-Theoretical Approach to the Benefits of Cloud Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 148–160. [Online]. Available: https://doi.org/10.1007/978-3-642-28675-9_11

[130] V. Vazirani, "Approximation algorithms springer-verlag," *New York*, 2001.

[131] M. R. Garey, R. L. Graham, D. S. Johnson, and A. C.-C. Yao, "Resource constrained scheduling as generalized bin packing," *Journal of Combinatorial Theory, Series A*, vol. 21, no. 3, pp. 257–298, 1976.

[132] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A scalable application placement controller for enterprise data centers," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 331–340.

[133] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder, "Heuristics for vector bin packing," *research. microsoft. com*, 2011.

[134] J. F. Nash *et al.*, "Equilibrium points in n-person games," *Proceedings of the national academy of sciences*, vol. 36, no. 1, pp. 48–49, 1950.

[135] J. von Neumann, "On the theory of parlor games," *Mathematische Annalen*, vol. 100, pp. 295–320, 1928.

[136] A. Avron, “Mitchell john c.. foundations for programming languages. foundations of computing. the mit press, cambridge, mass., and london, 1996, xix+ 846 pp.” *The Journal of Symbolic Logic*, vol. 64, no. 2, pp. 918–922, 1999.

[137] R. Porter, E. Nudelman, and Y. Shoham, “Simple search methods for finding a nash equilibrium,” *Games and Economic Behavior*, vol. 63, no. 2, pp. 642–662, 2008.

[138] V. Jalaparti and G. D. Nguyen, “Cloud resource allocation games,” Tech. Rep., 2010.

[139] P. S. Pillai and S. Rao, “Resource allocation in cloud computing using the uncertainty principle of game theory,” *IEEE Systems Journal*, vol. 10, no. 2, pp. 637–648, 2016.

[140] F. Teng and F. Magoulès, “A new game theoretical resource allocation algorithm for cloud computing,” in *International Conference on Grid and Pervasive Computing*. Springer, 2010, pp. 321–330.

[141] Y. Han, T. Alpcan, J. Chan, and C. Leckie, “Security games for virtual machine allocation in cloud computing,” in *International Conference on Decision and Game Theory for Security*. Springer, 2013, pp. 99–118.

[142] B. C. Vattikonda, S. Das, and H. Shacham, “Eliminating fine grained timers in xen,” in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. ACM, 2011, pp. 41–46.

[143] N. S. V. Rao, S. W. Poole, F. He, J. Zhuang, C. Y. T. Ma, and D. K. Y. Yau, “Cloud computing infrastructure robustness: A game theory approach,” in *2012 International Conference on Computing, Networking and Communications (ICNC)*, Jan 2012, pp. 34–38.

[144] T. Deshane, Z. Shepherd, J. Matthews, M. Ben-Yehuda, A. Shah, and B. Rao, “Quantitative comparison of xen and kvm,” *Xen Summit, Boston, MA, USA*, pp. 1–2, 2008.

[145] “Xen project,” http://www.xenproject.org/.

[146] C. Math, “The apache commons mathematics library,” : http://commons. apache. org/proper/commons-math/, 2014.

[147] P. Wendykier, “Jtransforms,” https://sites.google.com/site/piotrwendykier/software/jtransforms, 2009.

[148] “Openstack documentation.” [Online]. Available: http://docs.openstack.org/

[149] “Qfabric: Enabling the new data center,” http://www.slideshare.net/junipernetworks/qfabric-enabling-the-new-data-center.

[150] W. Tarreau *et al.*, “Haproxy-the reliable, high-performance tcp/http load balancer,” 2012.

[151] “Kvm,” http://www.linux-kvm.org/page/Main_Page.

[152] “Vmware,” https://www.vmware.com/.

[153] “Benchvm,” Benchvm,http://www.clarkson.edu/projects/virtualization/benchvm/.

[154] “Qfabric: Enabling the new data center,” http://www.slideshare.net/junipernetworks/qfabric-enabling-the-new-data-center.

[155] “Spec benchmark,” http://www.spec.org/benchmarks.html.

[156] “Rally benchmark,” https://wiki.openstack.org/wiki/Rally.

[157] “Autotest,” https://github.com/autotest/autotest.

[158] “Sensu,” http://sensuapp.org/docs/.

[159] “Najios,” http://www.nagios.org/.

[160] T. Robertazzi, *Computer Networks and Systems: Queueing Theory and Performance Evaluation*, ser. Telecommunication networks and computer systems,2000. Springer, 2000.

[161] V. Gupta, M. Harchol-Balter, J. Dai, and B. Zwart, “On the inapproximability of m/g/k: why two moments of job size distribution are not enough,” *Queueing Systems*, vol. 64, no. 1, pp. 5–48, 2010.

[162] “Cloudsuite benchmark,” http://parsa.epfl.ch/cloudsuite/overview.html.

[163] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the clouds: a study of emerging scale-out workloads on modern hardware," in *ACM SIGPLAN Notices*, vol. 47, no. 4, 2012, pp. 37–48.

[164] C. Delimitrou and C. Kozyrakis, "Quasar: resource-efficient and qos-aware cluster management," *ACM SIGPLAN Notices*, vol. 49, no. 4, pp. 127–144, 2014.

[165] D. Y. Barrer, "Queuing with impatient customers and ordered service," *Operations Research,*, vol. 5, no. 5, pp. pp. 650–656, 1957.

[166] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in *Grid Computing (GRID), 2011 12th IEEE/ACM Int. Conference on*, Sept 2011, pp. 26–33.

[167] Y. Ajiro and A. Tanaka, "Improving packing algorithms for server consolidation." in *Int. CMG Conference*. Computer Measurement Group, 2007, pp. 399–406.

[168] J. Li, Q. Wang, D. Jayasinghe, J. Park, T. Zhu, and C. Pu, "Performance overhead among three hypervisors: An experimental study using hadoop benchmarks," in *BigData Congress, IEEE Int. Congress on*, 2013, pp. 9–16.

[169] R. G. Michael and S. J. David, "Computers and intractability: a guide to the theory of np-completeness," *WH Free. Co., San Fr*, 1979.

[170] S. Homsi, S. Liu, G. A. Chaparro-Baquero, O. Bai, S. Ren, and G. Quan, "Workload consolidation for cloud data centers with guaranteed qos using request reneging," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 7, pp. 2103–2116, July 2017.

[171] B. Wilder, *Cloud architecture patterns: using microsoft azure*. " O'Reilly Media, Inc.", 2012.

[172] *Microsoft Azure Government Cloud Computing*. [Online]. Available: https://azure.microsoft.com/en-us/overview/clouds/government/

[173] J. Y. Arrasjid, "Virtustream and vmware enable mission critical hybrid cloud," 2017. [Online]. Available: Availableat:https://www.virtustream.com/videos/LHC3380BES_Virtustream-and-VMware-Enable-Mission-Critical-Hybrid-Cloud.pdf

[174] *Mission Critical Cloud*. [Online]. Available: https://www.cogecopeer1.com/services/cloud/mission-critical/

[175] M. Hadji and D. Zeghlache, "Mathematical programming approach for revenue maximization in cloud federations," *IEEE transactions on cloud computing*, vol. 5, no. 1, pp. 99–111, 2017.

[176] K. Gai, M. Qiu, and H. Zhao, "Cost-aware multimedia data allocation for heterogeneous memory using genetic algorithm in cloud computing," *IEEE transactions on cloud computing*, 2016.

[177] R. Meng, Y. Ye, and N.-g. Xie, "Multi-objective optimization design methods based on game theory," in *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*. IEEE, 2010, pp. 2220–2227.

[178] N. V. Sahinidis, "Optimization under uncertainty: state-of-the-art and opportunities," *Computers & Chemical Engineering*, vol. 28, no. 6, pp. 971–983, 2004.

[179] M. Li, Y. Zhang, K. Bai, W. Zang, M. Yu, and X. He, "Improving cloud survivability through dependency based virtual machine placement." in *SECRYPT*, 2012, pp. 321–326.

[180] M. Hazewinkel, "Minimax principle," *Encyclopedia of Mathematics, Springer*, 2001.

[181] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*. Cambridge University Press Cambridge, 2007, vol. 1.

[182] K. Garrett and E. Moore, "Teaching mixed strategy nash equilibrium to undergraduates," *International Review of Economics Education*, vol. 7, no. 2, pp. 79–87, 2008.

[183] J. Nash, "Non-cooperative games," *Annals of mathematics*, pp. 286–295, 1951.

[184] J. Quaintance, *Combinatorial identities for Stirling numbers: the unpublished notes of HW Gould*. World Scientific, 2015.

[185] *1000 VM per rack is the new minimum*. [Online]. Available: http://virtual-red-dot.info/1000-vm-per-rack-is-the-new-minimum/

[186] D. C. Llewellyn, C. Tovey, and M. Trick, "Finding saddlepoints of two-person, zero sum games," *The American Mathematical Monthly*, vol. 95, no. 10, pp. 912–918, 1988.

[187] M. Dresher, "Games of strategy: theory and applications," RAND CORP SANTA MONICA CA, Tech. Rep., 1961.

[188] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou, "The complexity of computing a nash equilibrium," *SIAM Journal on Computing*, vol. 39, no. 1, pp. 195–259, 2009.

[189] I. Dunning, J. Huchette, and M. Lubin, "Jump: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, 2017.

[190] G. Optimization, "Inc.,gurobi optimizer reference manual, 2015," *URL: http://www. gurobi. com*, 2014.

[191] M. Stillwell, D. Schanzenbach, F. Vivien, and H. Casanova, "Resource allocation algorithms for virtualized service hosting platforms," *Journal of Parallel and distributed Computing*, vol. 70, no. 9, pp. 962–974, 2010.

[192] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: A performance evaluation," in *IEEE International Conference on Cloud Computing*. Springer, 2009, pp. 254–265.

[193] C. Reiss, A. Tumanov, G. R. Ganger, Y. H. Katz, and M. A. Kozuch, "Towards understanding heterogeneous clouds at scale: Google trace analysis," 2012.

[194] V. Janapa Reddi, B. C. Lee, T. Chilimbi, and K. Vaid, "Web search using mobile cores: quantifying and mitigating the price of efficiency," *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3, pp. 314–325, 2010.

[195] R. Nathuji, C. Isci, and E. Gorbatov, "Exploiting platform heterogeneity for power efficient data centers," in *Autonomic Computing, 2007. ICAC'07. Fourth International Conference on*. IEEE, 2007, pp. 5–5.

[196] B. C. Lee and D. M. Brooks, "Illustrative design space studies with microarchitectural regression models," in *High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium on*. IEEE, 2007, pp. 340–351.

[197] S. Garg, S. Sundaram, and H. D. Patel, "Robust heterogeneous data center design: a principled approach," *ACM SIGMETRICS Performance Evaluation Review*, vol. 39, no. 3, pp. 28–30, 2011.

[198] S. Hazelhurst, "Scientific computing using virtual high-performance computing: a case study using the amazon elastic computing cloud," in *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*. ACM, 2008, pp. 94–103.

[199] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, "Performance analysis of high performance computing applications on the amazon web services cloud," in *2nd IEEE international conference on cloud computing technology and science*. IEEE, 2010, pp. 159–168.

[200] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 123–134, Aug. 2009.

VITA

SOAMAR HOMSI

2007	B.S., Computer Engineering Albaath University Homs, Syria
2014	M.S., Computer Engineering Florida International University Miami, Florid
2019	Ph.D., Electrical and Computer Engineering Florida International University (FIU) Miami, Florida

Selected Publications and Presentations in Discipline

- Homsi, S., Liu, S., Chaparro-Baquero, G. A., Bai, O., Ren, S., & Quan, G. (2017). Workload consolidation for cloud data centers with guaranteed QoS using request reneging. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 28(7), 2103-2116.
- Wang, T., Homsi S., Niu, L., Ren, S., Bai, O., Quan, G., & Qiu, M. (2017). Harmonicity-aware task partitioning for fixed priority scheduling of probabilistic real-time tasks on multi-core platforms. *ACM Transaction on Embedded Computing Systems (TECS)*, 16(4), 101.
- Homsi S., Wen. W, Gang Quan, Chaparro-Baquero G., & Njilla L. (2019, May). Game Theoretic-based Approaches for Cybersecurity-aware Virtual Machine Placement in Public Cloud Clusters. The 19th Annual IEEE/ACM International Symposium in Cluster, Cloud, and Grid Computing (CCGRID), Accepted.
- Homsi, S., Quan, G., & Njilla, L. (2018, November). Critical Workload Deployment in Public Clouds with Guaranteed Security Levels and Optimized Resource Usage and Energy Cost. *In Proceedings of the Future Technologies Conference (FTC)* (pp. 240-260). Springer, Cham.
- Chaparro-Baquero, G. A., Sha, S., Homsi, S., Wen, W., & Quan, G. (2017, October). Thermal-aware joint CPU and memory scheduling for hard real-time tasks on multicore 3D platforms. *In 2017 Eighth International Green and Sustainable Computing Conference (IGSC)* (pp. 1-8). IEEE.

- Chaparro-Baquero, G. A., Sha, S., Homsi, S., Wen, W., & Quan, G. (2017, March). Processor/memory Co-Scheduling using periodic resource server for real-time systems under peak temperature constraints. *In 2017 18th International Symposium on Quality Electronic Design (ISQED)* (pp. 360-366). IEEE.
- Chaparro-Baquero, G. A., Homsi, S., Vichot, O., Ren, S., Quan, G., & Ren, S. (2015, October). Cache allocation for fixed-priority real-time scheduling on multicore platforms. *In 2015 33rd IEEE International Conference on Computer Design (ICCD)* (pp. 589-596). IEEE.
- Liu, S., Homsi, S., Fan, M., Ren, S., Quan, G., & Ren, S. (2015, March). Power minimization for data center with guaranteed QoS. In 2015 Design, *Automation & Test in Europe Conference & Exhibition (DATE)* (pp. 1347-1352). IEEE.
- Liu, S., Homsi, S., Fan, M., Ren, S., Quan, G., & Ren, S. (2014, December). Scheduling time-sensitive multi-tier services with probabilistic performance guarantee. *In 2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)* (pp. 736-743). IEEE.
- Homsi S., Quan G., & Njilla L. Secure Deployment of Critical Workloads in Energy Efficient Public Clouds. (2018, July). *The 5th Annual Colloquium on Game Theory applied to Cybersecurity.* Air Force Research Laboratory (AFRL), Rome, NY, July 2018.