Dynamic Cloud Resource Allocation Considering Demand Uncertainty

Seyedehmehrnaz Mireslami, *Student Member, IEEE,* Logan Rakai, *Member, IEEE,* Mea Wang, *Member, IEEE* and Behrouz Homayoun Far, *Member, IEEE*

Abstract—Cloud computing provisions scalable resources for high performance industrial applications. Cloud providers usually offer two types of usage plans: reserved and on-demand. Reserved plans offer cheaper resources for long-term contracts while on-demand plans are available for short or long periods but are more expensive. To satisfy incoming user demands with reasonable costs, cloud resources should be allocated efficiently. Most existing works focus on either cheaper solutions with reserved resources that may lead to under-provisioning or over-provisioning, or costly solutions with on-demand resources. Since inefficiency of allocating cloud resources can cause huge provisioning costs and fluctuation in cloud demand, resource allocation becomes a highly challenging problem. In this paper, we propose a hybrid method to allocate cloud resources according to the dynamic user demands. This method is developed as a two-phase algorithm that consists of reservation and dynamic provision phases. In this way, we minimize the total deployment cost by formulating each phase as an optimization problem while satisfying quality of service. Due to the uncertain nature of cloud demands, we develop a stochastic optimization approach by modeling user demands as random variables. Our algorithm is evaluated using different experiments and the results show its efficiency in dynamically allocating cloud resources.

Index Terms—Cloud computing, Demand uncertainty, Quality of service, Stochastic programming, Pricing and resource allocation.

1 Introduction

Cloud computing is a popular networking paradigm that provides resources via Internet [1], [2]. Cloud computing helps web service providers reduce hardware infrastructure expenses for deploying their applications. In addition, easy resource management and fast response time are the other interesting characteristics that bring the attentions to the cloud computing [3], [4]. In this paper, the focus is on cloud Infrastructure-as-a-Service (IaaS), where infrastructure resources such as network, computing, database, etc. are offered by cloud providers.

Cloud providers usually offer two types of IaaS resource provisioning plans, reserved and on-demand plans, to web service providers that have different charging schemes based on the resource usage [5]–[7]. The reserved plans are often offered for relatively long-term contracts. Using reserved plans, web service providers can get discount rates on reserved resources and pay once for the contract time period (e.g. one-year contract or three-year contract for Amazon EC2) [8]. Through on-demand plans, cloud providers offer more flexible resource pricing strategies. On-demand plans charge cloud web service providers on a pay-as-you-go basis and enable them to start or terminate instances at any moment according to their needs without paying any penalty. However, comparing the costs of resources per unit of time, on-demand resources are often more expensive than the reserved ones.

With the reserved plans, web service providers reserve instances in advance for long-term contracts. Due to ignorance of demand uncertainty in the reserved plans, resource provisioning only with the reserved instances is a challenging task. The purchased resources may not be enough to handle the demands all the time that leads to under-provisioning. This may result in failure in meeting web service providers' Quality of Service (QoS) criteria which is a crucial concern for both cloud providers and web service providers in presence of the uncertainty in the demands [9]. On the other hand, over-provisioning may happen if the allocated resources are excessive to handle actual arrived demands most of the time [1], leading to unnecessary costs.

Recently, some research studies have addressed cloud resource allocation, as optimizing resource provisioning costs has become important for cloud web service providers [9]–[11]. Most of the existing approaches in cloud resource allocation, model resource allocation problem as a single-phase algorithm [12]–[15]. In these works, the authors ignore demand uncertainty by assuming deterministic values for demands. Therefore, the elastic nature of cloud-based applications is not considered. To address the demand uncertainties, in [16]–[19], several dynamic resource allocation

- *S. Mireslami is with the Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB, T2N1N4, Canada. E-mail: smiresla@ucalgary.ca.*
- *L. Rakai is with the Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB, T2N1N4, Canada. E-mail: lmrakai@ucalgary.ca.*
- *M. Wang is with the Department of Computer Science, University of Calgary, Calgary, AB, T2N1N4, Canada. E-mail: meawang@ucalgary.ca.*
- *B. Far is with the Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB, T2N1N4, Canada. E-mail: far@ucalgary.ca.*

- *The binary executable of this work is released here: "https://www.ucalgary.ca/mireslami/publications"*

Manuscript received February 18, 2018

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2897304, IEEE Transactions on Cloud Computing

algorithms are developed. These algorithms are more flexible and allocate cloud resources dynamically to optimize resource provisioning costs. However, these works do not often exploit the cost benefits of the reserved plans that are offered by the cloud providers. Therefore, they may fail to achieve economical solutions.

In this paper, we propose a hybrid method to allocate cloud resources for deploying cloud-based web applications dynamically. We take advantage of reserved and on-demand resources and achieve a hybrid solution that minimizes total deployment cost and guarantees the QoS under demand uncertainties. The major contributions of this paper can be summarized as follows: Proposing Dynamic Cloud Resource Allocation (DCRA) algorithm that solves the resource provisioning in two reserved and dynamic provision phases, developing a stochastic optimization approach to model the user demands as random variables, and achieving 10% improvement in total deployment costs.

The proposed DCRA algorithm is evaluated using two different benchmark workloads in Amazon Web Services (AWS) [7] and Microsoft Azure cloud [6] as the cloud providers. The results show that the proposed DCRA algorithm finds solutions that minimize total deployment costs under the uncertainties in users' demands.

The rest of this paper is organized as follows. Relevant existing resource allocation algorithms are discussed in Section 2. In Section 3, the system model and assumptions are explained. Section 4 provides the detailed explanation of the proposed DCRA algorithm. In Section 5, we present the results of experiments and data analysis using real price models and benchmark workloads. Finally, conclusion and future work are given in Section 6.

2 RELATED WORKS

The problem of cloud resource provisioning has brought researchers' attention to providing resource allocation algorithms and techniques in the past few years [20]–[23]. Most of the resource allocation works model the problem as a single phase algorithm that only considers resources with reserved plans from the cloud providers [24]–[27]. These works either do not consider the uncertain nature of user's demands and denote the demand as a deterministic value [13] or simply use an expensive over-provisioning approach for the worst-case demand [12]. To cope with this problem, in [9], [28], [29], on-demand resource provisioning methods are proposed to allocate resources according to the dynamic cloud demand. In the following sections, the existing works are discussed in two deterministic resource provisioning and dynamic resource provisioning categories.

2.1 Deterministic Resource Provisioning

Jiao *et al.* developed a cost model for Online Social Network deployment [13]. The goal of this work is to optimize the monetary cost spent for using cloud resources while satisfying the QoS such as access latency. Considering that the modeled optimization problem is NP-hard, a heuristic algorithm is proposed to ensure the QoS requirements are met. However, the optimization is performed based on a deterministic demand of geo-distributed clouds and the work does not consider dynamic provisioning. Moreover, since the QoS requirements are based on the user location. If the location of the user is changed, the existing solution may not satisfy the QoS requirements anymore. Using a heuristic algorithm, the cost is optimized but the achieved solution may be only a local optimum. Similarly, in [27], a multi-objective algorithm is proposed to minimize total deployment cost and maximize the QoS performance at the same time, in spite of being competing objectives. This algorithm includes a single phase optimization that only considers the reserved cloud resources for deployment. In addition, the application demands are assumed to be deterministic and the elasticity of cloud environment is ignored. The results show that the proposed algorithm achieves the optimal resources for web application deployment.

In [12], Imai *et al.* proposed an application-agnostic performance modeling for a broad types of applications to minimize the cost. To enhance the prediction accuracy, a probabilistic prediction capability is added to the model to foresee the application performance. To meet the Service Level Agreement (SLA) such as latency violation, user's maximum throughput is considered to create a model. Although the results show that the demand changes can be covered by the proposed model, it cannot satisfy the SLA during the whole execution time. Also, the authors do not provide any analytical model for cost and SLA.

All these works consider deterministic approaches to model different resource allocation problems, that may lead to under- and over-provisioning. To this end, in this paper, a dynamic cloud resource allocation algorithm is proposed to allocate resources according to the dynamic user demands.

2.2 Dynamic Resource Provisioning

To handle the uncertainties in user demands, in [17], stochastic programming is used to propose the Optimal Cloud Resource Provisioning (OCRP) algorithm. OCRP algorithm minimizes total cost by achieving a trade-off between reservation and on-demand resources. Each stage of the algorithm may include three provisioning phases including reservation, expending, and on-demand phases. In the first phase, resources are provisioned without considering the users's demand. Demands and prices are observed in the expanding phase. In case that purchased resources cannot satisfy the demands, more resources will be purchased in on-demand phase. Since the uncertainty model is discrete, no convex optimization technique can be used, *i.e*, no global optimal solution can be guaranteed. Although a decomposition method is employed in the OCRP algorithm to solve the subproblems in parallel and save the runtime, it affects the algorithm's quality of results.

Similarly, Chaisiri *et al.* minimized the resource provisioning cost by proposing a Robust Cloud Resource Provisioning (RCRP) algorithm in [9]. The proposed RCRP provisions cloud resources in three phases. Three types of uncertainties are used in this work, which include the uncertainties in demand, price and cloud resource availability. In the first phase, reservation is done where a specific amount of resources is assigned to the application. The usage of resources is observed in the expending phase to recognize that resources are under- or over-provisioned. If reserved

resources cannot satisfy the incoming demands, in the on-demand phase, further resources should be assigned in the form of on-demand resources. The results show that the RCRP algorithm obtains minimum on-demand costs compared to the existing works. However, since the proposed RCRP algorithm employs robust optimization, it takes a pessimistic approach that yields to a relatively high cost. In addition, this algorithm does not differentiate the computing and database cloud resource types in problem modeling.

In [28], Ran *et al.* proposed a cost-efficient provisioning strategy for resource allocation problem in distributed cloud environment. The objective of this work is to minimize the cost of purchasing resources and to maximize the profit by considering both demand and cost uncertainties. The proposed strategy includes two stochastic programming stages. First, the deterministic equivalent formulation is expressed to minimize the cost of purchasing Virtual Machines (VM). In the second stage, the profit is maximized by allocating the resources optimally. Although the deployment cost is minimized, the main focus of this work is just the dynamic resource allocation part and reserved instances are ignored.

Yu *et al.* proposed a virtual cloud allocation algorithm to minimize the bandwidth occupancy cost [29]. Stochastic virtual cluster is proposed by considering demand uncertainty to ensure the tenants' bandwidth demands are satisfied. In this paper, normal distribution is considered to represent the bandwidth demand uncertainty. The proposed dynamic programming based allocation algorithm has exponential time complexity which makes it inefficient for applications that require a large number of VMs. To cope with this problem, a heuristic algorithm is developed to decrease the time complexity by considering limitation for the capacity of allocated VMs. Although the bandwidth cost is minimized, this work cannot guarantee to achieve the global minimum cost due to of the nature of heuristic algorithms.

In [26], the main goal is minimizing the total energy consumption. A stochastic linear programming model is developed in this work to provision cloud resources. The cloud service uncertainty and time variability are assumed in the proposed model. Monte Carlo simulations are used to produce the input parameters. The results show that this work can reduce the power consumption by up to 60%.

In this paper, we propose the DCRA algorithm to obtain the minimum total deployment cost in two reserved and dynamic provision phases. Also, an optimization approach is developed to consider cloud demand uncertainty to enable efficient dynamic cloud resource allocation for web applications. Compared to [9] and [17] that both take pessimistic approaches by employing worst-case robust optimization, we propose a stochastic optimization approach to obtain a balanced solution and avoid unnecessary costs due to pessimistic solutions. Unlike [28] where the decisions on reserved instances are left to the the web service providers and are never optimized, our proposed algorithm optimizes both required reserved and on-demand resources. Compared to [29] that propose a heuristic algorithm, we employ convex optimization that is efficiently solved and achieves a global optimum solution for resource provisioning. Finally, unlike the work proposed in [26] that considers the service uncertainty from the cloud provider's point of view, our proposed DCRA algorithm addresses the problem of dynamic cloud resource allocation from the cloud web service providers' point of view.

3 SYSTEM MODEL AND ASSUMPTIONS

In almost all the cloud providers such as Amazon [7] and Microsoft Azure [6], the reserved resources are secured through long-term (*e.g.*, months or years) contracts, whereas on-demand resources are acquired for any arbitrary time period, typically in hours. We consider the reserved resources to have a one-time cost (independent of the time period) while the on-demand resources are charged for the time period they are utilized. To meet diverse demands, cloud providers supply different types of VMs with different resource configuration. The price of these resources differs by the subscription length, long-term or short-term. As exemplified, the Amazon pricing model [7] and the GoGrid [30] pricing model are presented in Table 1.

Since user requests for web applications are not deterministic, for cost efficiency, web service providers subscribe to a combination of reserved and on-demand cloud services [9], [15], [21]. However, existing cloud resource allocation works, fail to consider both reserved and on-demand resource types leading to under-provisioning or over-provisioning [25], [31]. In this paper, we propose a Dynamic Cloud Resource Allocation (DCRA) algorithm that minimizes the combined cost of reserved and on-demand cloud services. In DCRA, we consider the dual price model from the cloud provider and the QoS requirements from the web service provider. DCRA takes a realistic approach to cope demand uncertainty in cloud-based web service provisioning. This leads to a solution that maintains the web application performance under demand changes.

Fig. 1 illustrates where DCRA fits in the cloud-based web service deployment. The algorithm is deployed as a module on the primary server of the web service. This is usually a proxy or load-balancing server that is the first contact point (*i.e*, the server associated to the domain name in the URL). Using the proposed DCRA algorithm enables the web service providers to optimize the resource provision costs. This module takes as input the QoS requirements from the web service provider and the price model from the cloud service provider. It runs the DCRA algorithm to determine the optimal resource allocation. As will be shown in Sec. 5, the runtime of DCRA is relatively low, hence it can be effectively used to determine the resource allocation in real-time according to the changing demands.

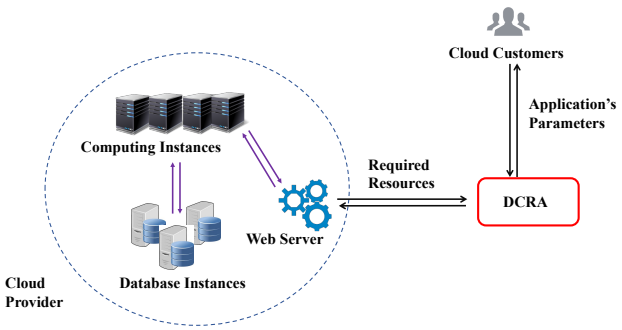


Fig. 1. A Model of Web Service Deployment in Cloud.

3.1 Problem Definition

In this paper, parameters from the cloud provider and web service provider are considered to model the resource allocation problem. In order to meet various user demands, cloud providers supply different types of VMs with different resource configuration. In this paper, this information is collected as the input to the cloud deployment problem. Most cloud providers, such as Amazon [7], Microsoft Azure [6] and GoGrid [30], set the price of resources based on the two resource provisioning plans: reserved plan and on-demand plan. Therefore, two different prices are associated with the VMs in cloud models as explained below.

To model the cloud deployment problem from web service provider's point of view, in addition to the cloud providers' information and pricing, the web service provider's application requirements should be integrated into the model. Different types of resource requirements and QoS performance criteria to support the web service providers' application are explained below.

Parameters from cloud provider (algorithm inputs):

P^r_{db} Price for reserved database instance ($)
P^o_{db} Pay-per-use price for on-demand database instance ($\frac{\$}{\text{hour}}$)
P^r_c Price for reserved computing instance ($)
P^o_c Pay-per-use price for on-demand computing instance ($\frac{\$}{\text{hour}}$)
P^r_s Price of storage for each reserved database instance ($\frac{\$}{GB}$)
P^r_{IO} Price of reserved Input/Output (I/O) request capacity ($)
P^o_{IO} Pay-per-use price for on-demand I/O request capacity ($)
R^r_c Number of requests per hour a reserved computing instance can handle ($\frac{1}{\text{hour}}$)
R^o_c Number of requests per hour an on-demand computing instance can handle ($\frac{1}{\text{hour}}$)
R^r_{db} Number of requests per hour a reserved database instance can handle ($\frac{1}{\text{hour}}$)
R^o_{db} Number of requests per hour an on-demand database instance can handle ($\frac{1}{\text{hour}}$)

Parameters from web service provider (algorithm inputs):

d_t Total deployment service time (hour)
d_p Dynamic cloud provisioning duration (hour)
r_{min_c} Minimum number of user requests per hour for computing servers ($\frac{1}{\text{hour}}$)
$r_{min_{db}}$ Minimum number of user requests per hour to access the database servers ($\frac{1}{\text{hour}}$)
$\mathrm{t_{db}}$ Maximum time for accessing database server (s)
$\mathrm{t_c}$ Maximum time for serving a user request (s)
s Required storage for web application (GB)

Problem varibales (algorithm outputs):

α^o_c Number of on-demand computing instances
α^r_c Number of reserved computing instances
β^o_{db} Number of on-demand database instances
β^r_{db} Number of reserved database instances
μ^r_s Required storage for each reserved database instance (GB)
γ^o_{Rc} Minimum on-demand service rate given by computing instances ($\frac{1}{\text{hour}}$)
γ^r_{Rc} Minimum reserved service rate given by computing instances ($\frac{1}{\text{hour}}$)
λ^o_{Rdb} Minimum on-demand service rate given by database instances ($\frac{1}{\text{hour}}$)
λ^r_{Rdb} Minimum reserved service rate given by database instances ($\frac{1}{\text{hour}}$)

4 Dynamic Cloud Resource Allocation Algorithm

In contrast to existing cloud resource allocation algorithms [14], [32], we aim to provide dynamic cloud resource allocation that considers the dual price plan from cloud providers and finds the balance between reserved and on-demand resources. In this section, we propose Dynamic Cloud Resource Allocation (DCRA) algorithm, a two-phase algorithm that minimizes the cost of the web service deployment. In the first phase (referred to as the reservation phase), resources from the reserved plan are allocated for web application deployment to meet the minimum QoS requirements. In the second phase (referred to as the dynamic provision phase), non-deterministic user demands are modeled as random variables. A stochastic optimization approach is proposed to dynamically allocate on-demand resources to minimize deployment costs under the on-demand plan subject to QoS requirements.

4.1 DCRA Flowchart Overview

Due to the uncertain nature of the demand, we propose DCRA, a two-phase resource allocation algorithm that reserves computing and database instances according to the dual price plan to cost-effectively handle the dynamic changes in the demand. The two phases of DCRA, namely, reservation phase and dynamic provision phase, are presented in Fig. 2. Cloud providers offer certain amount of resources as reserved resources to web service providers in form of long-term contracts at lower prices (under the reserved plan). For example, Amazon EC2 [7] reserved plan is available in one-year or three-year contracts while GoGrid [30] offers annual and monthly contracts. In the reservation phase, the algorithm determines the minimum resources to meet the minimum user requests r_{min} (in requests per hour) for the web services. Note that r_{min} is given as an input by the web service provider just once at the beginning of the deployment service time and is never calculated nor updated later. Therefore, including it into the input parameters does not lead to any extra overhead. More specifically, the cloud resource reservation optimization model takes the minimum requests r_{min} as the input and outputs the minimum required cloud resources (computing and database instances). The reservation optimization is modeled using a geometric programming approach, as will be detailed in Sec. 4.2. The output of the optimization model is fed to the resource control unit to reserve the resources in advance at a flat rate to provide the minimum QoS guarantee.

In the dynamic provision phase, the DCRA algorithm monitors the actual user demands for the web application for a time period d_p (*e.g.*, a 15-minute period). Although

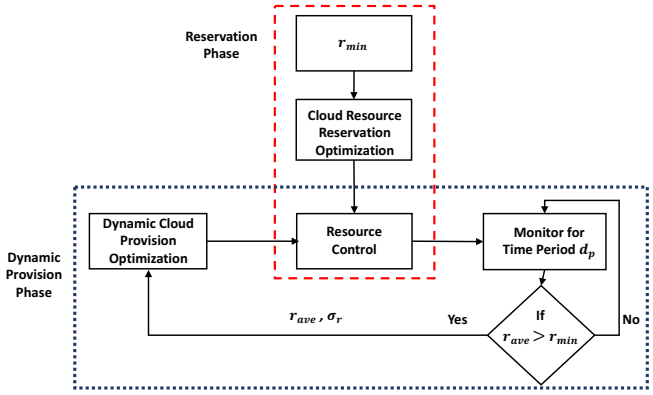


Fig. 2. Flowchart of Dynamic Cloud Resource Allocation (DCRA) algorithm.

for DCRA, very short time periods d_p can be considered as the algorithm takes a very low runtime, 15-minute is a reasonable choice for d_p as it is close to the length of the monitoring period of the major public cloud providers' scaling tools, *e.g.*, [33]. As the DCRA's runtime is relatively low, the runtime bottleneck for having a shorter dynamic cloud provisioning duration is the time needed to provision and boot a new cloud instance. If setting up a new cloud instance takes a few minutes, then, it does not make sense to have a dynamic cloud provisioning duration d_p shorter than that. In addition, most of the major public cloud providers charge the full hour price for a cloud instance, no matter if it has been used for 1 second or 59 minutes. Therefore, in case of over-provisioning, having shorter dynamic cloud provisioning durations does not really help the deployment costs. However, it can help in the case of under-provisioning where new cloud instances can be secured as soon as the user demands grow.

As described in Fig. 2, if the monitored average user requests r_{avg} exceeds the minimum expected requests, r_{min}, the web application is under-provisioned. DCRA applies the dynamic provision optimization to secure on-demand resources to meet the dynamic demands. This optimization model takes the average requests r_{avg} as the input and outputs the minimum required on-demand cloud resources (computing and database instances). Dynamic cloud provision optimization is modeled using a stochastic optimization approach, as will be detailed in Sec. 4.3, to achieve solutions that are robust to the uncertainties in the user demands. The output of the optimization model is fed to the resource control unit to secure the on-demand resources at higher costs to meet the actual demand. The next round of dynamic provision begins again with the monitoring. The output of each iteration of the algorithm is a solution with minimum total deployment cost that maintains the QoS under demand uncertainty. In reality, the demand may increase or decrease over time. In DCRA, as long as the observed requests r_{avg} is higher than r_{min}, it leaves it to the dynamic provision optimization to dynamically adjust the provisioning of the on-demand resources. In case, the observed requests r_{avg} is lower than r_{min}, all on-demand resources should be turned off, and the resources from the reservation phase should be adequate to satisfy the demands.

Overall, DCRA determines and allocates cloud resources dynamically to provide a flexible optimization that meets the QoS requirements of the web application under demand uncertainty with a minimum cost solution. In this paper, we focus on developing and solving the two major optimization blocks: reservation optimization and dynamic provision optimization. The resource control unit and monitoring are often provided by the cloud providers.

4.2 Cloud Resource Reservation Optimization

In this paper, the resource reservation optimization is proposed to help web service providers to make business decisions and obtain optimal solutions for their reserved resources. In the reservation phase, our goal is to secure the resources to meet the minimum QoS requirements with minimum cost. Since the DCRA is designed based on the dual price model from the cloud provider, this phase takes the price model from the reserved plan and the minimum user request r_{min} as inputs. We employ geometric programming [25] to model the optimization problem as follows:

$$\begin{aligned} \min . \quad & \beta^r_{db} \times P^r_{db} \\ & + \beta^r_{db} \times P^r_s \times {\mu_s}^r \\ & + \alpha^r_c \times P^r_c \\ & + [\lambda^r_{Rdb} + \gamma^r_{Rc}] \times P^r_{IO} \times d_t \end{aligned} \tag{1}$$

The cost function is defined according to the cloud providers' reserved pricing model. The deployment cost in (1) includes the cost of reserved database instances (term 1), the cost of required storage for reserved databases (term 2) and the cost of reserved computing instances (term 3) for a long-term contract d_t (e.g. one-year). For the storage term, we assume that the storage required by the computing instances are negligible because they either process the requests or query the database instances. Thus, the storage requirement is associated with the database instances only. Communication of reserved computing and database servers for the given contract period d_t is also considered (term 4). The reserved resources' communication costs are calculated based on the usage over time [7]. For the communication cost, the cost is associated with the duration of the service d_t because the request rates are in request per hour. This formulation is subject to a set of constraints explained in the following paragraphs. Collectively, we wish to minimize the total cost of the resources but also need to meet several constrains.

$$\alpha^r_c \times {R^r}_c \geq r_{min_c} \tag{2}$$

$$\beta^r_{db} \times {R^r}_{db} \geq r_{min_{db}} \tag{3}$$

As defined in Eqn. 2 and Eqn. 3, we first need to ensure that we secure sufficient database instances and computing instances such that their respective service rate meet the minimum requests for database service $r_{min_{db}}$ and the minimum requests for computing service r_{min_c}. The R^r_{db} and R^r_c

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2897304, IEEE Transactions on Cloud Computing

are the maximum service rate of each database instance and computing instance, respectively.

$$\frac{\frac{1}{\gamma^r_{Rc}}}{1-\frac{r_{min_c}}{\gamma^r_{Rc}}} \leq \mathrm{t_c} \tag{4}$$

$$\frac{\frac{1}{\lambda^r_{Rdb}}}{1-\frac{r_{min_{db}}}{\lambda^r_{Rdb}}} \leq \mathrm{t_{db}} \tag{5}$$

The web application request arrival rate follows a Poisson process as shown in [31]. Considering the dependency between arrival rate and response time of VM instances, the constraints in Eqn. 4 and Eqn. 5 are defined to ensure the achievable service rates by computing instances γ^r_{Rc} and database instances λ^r_{Rdb} meet the expected response time constraints for the computing requests t_c and database requests t_{db}, respectively. The conversion from service rate to response time is based on the model proposed in [31].

$$\beta^r_{db} \times \mu^r_s \geq s \tag{6}$$

Storage constraint in Eqn. 6 controls that the reserved storage space for web application is sufficient.

$$\alpha^r_c + \beta^r_{db} \geq 1 \tag{7}$$

$$\alpha^r_c \in \{0,1,2,\ldots\} \tag{8}$$

$$\beta^r_{db} \in \{0,1,2,\ldots\} \tag{9}$$

To start deploying the web application, web service provider should purchase at least one reserved database or computing instance that is expressed by the constraint in (7). Integer constraints (8) and (9) indicate that computing and database variables take values from a set of non-negative integer numbers.

In the reservation phase of DCRA algorithm, the cloud resource reservation optimization is formulated and solved. Although the resource reservation optimization is not originally GP, it can be converted to GP form and be solved using convex optimization techniques [25]. This phase outputs an optimal cost combination of reserved resources that meets the minimum demands while meeting the QoS requirements. However, considering the elasticity of cloud applications and the demand uncertainty, such solution will not provide a robust performance during the deployment time. Therefore, in the next section, a novel dynamic cloud provision phase for the DCRA algorithm is proposed.

4.3 Dynamic Cloud Provision Optimization

The uncertaint nature of user demands for web applications in cloud environment makes any deterministic resource allocation optimization inadequate to satisfy web service provider's QoS. In other words, it is simply unrealistic to assume an a-priori deterministic constant value for the user demands to a web application. According to [9], [22], [29], [34], [35], the web application demand uncertainty can be modeled by a normal distribution. To cope with the demand uncertainty, we adopt a stochastic optimization approach considering random variables with normal distributions for the user demands. Therefore, in this phase of DCRA, we define the user demands as random variables, which is the key to robustness against demand uncertainty. The solution allows web service providers to dynamically secure on-demand resources in cloud according to the actual incoming demands.

The optimization variables are the number of on-demand computing and database instances and their minimum service rates to meet the updated request, r_{ave_c} and $r_{ave_{db}}$, are received from the monitoring unit of DCRA algorithm. It is important for the web service providers to minimize the on-demand resources' costs during the given deployment period d_p which is considered as the problem minimization objective of the dynamic cloud provision optimization. $(\beta^o_{db} \times P^o_{db})$ and $(\alpha^o_c \times P^o_c)$ represent the per-time cost of all on-demand database and computing instances. The cost of communications between on-demand computing and database servers are modeled in the on-demand deployment cost as $(\lambda^o_{Rdb} + \gamma^o_{Rc}) \times P^o_{IO}$. Thus, the cost minimization objective function for the dynamic cloud provision optimization is defined as follows:

$$\begin{aligned} \min . \quad & (\beta^o_{db} \times P^o_{db} + \alpha^o_c \times P^o_c \\ & + [\lambda^o_{Rdb} + \gamma^o_{Rc}] \times P^o_{IO}) \times d_p \end{aligned} \tag{10}$$

This formulation is subject to a set of constraints. To avoid under-provisioning and over-provisioning, demand uncertainty is considered in Eqn. 11 and Eqn. 12 using a stochastic optimization approach. Based on [9], [22], [29], [34], [35], we model the demand uncertainty of database and computing resources (beyond the service guaranteed by the reserved plan) as random variables that follow normal distributions with expected values of r_{ave_c} and $r_{ave_{db}}$ and standard deviations of σ_{r_c} and $\sigma_{r_{db}}$, respectively. To ensure that the demands are met with 95% confidence interval, we meet the constraints for all realizations of the uncertain demands up to 2σ above the expected value [36]. In Section 5.3, we provide justification on the selection of this value. Note that the number of reserved computing and database instances that are obtained in the reservation phase are considered as constant values, N^r_{db} and N^r_c, in this phase.

$$(\alpha^o_c + N^r_c) \times {R^o}_c \geq r_{ave_c} + 2\sigma_{r_c} \tag{11}$$

$$(\beta^o_{db} + N^r_{db}) \times {R^o}_{db} \geq r_{ave_{db}} + 2\sigma_{r_{db}} \tag{12}$$

The constraints on response time of computing and database instances are modeled in Eqn. 13 and Eqn. 14. In these constraints, the demand uncertainty is considered to ensure that the secured reserved and on-demand instances satisfy web service providers' QoS performance requirements.

$$\frac{\frac{1}{\gamma^o_{Rc}+{R^r}_{Rc}}}{1-\frac{r_{ave_c}+2\sigma_{r_c}}{\gamma^o_{Rc}+{R^r}_{Rc}}} \leq \mathrm{t_c} \tag{13}$$

$$\frac{\frac{1}{\lambda^o_{Rdb}+{R^r}_{Rdb}}}{1-\frac{r_{ave_{db}}+2\sigma_{r_{db}}}{\lambda^o_{Rdb}+{R^r}_{Rdb}}} \leq \mathrm{t_{db}} \tag{14}$$

By adding constraints Eqn. 15 and Eqn. 16, we indicate on-demand computing and database instances are integers to guarantee physically meaningful solutions.

$${\alpha_c}^o \in \{0,1,2,\ldots\} \tag{15}$$

$${\beta_{db}}^o \in \{0,1,2,\ldots\} \tag{16}$$

TABLE 1
AWS and Microsoft Azure parameters for reserved and on-demand resources.

Cloud providers' data	AWS cloud provider		Microsoft Azure cloud provider	
	Reserved resources	On-demand resources	Reserved resources	On-demand resources
DB instance unit cost (P_{db})	297.84 ($\frac{\$}{\text{year}}$)	0.04 ($\frac{\$}{\text{hour}}$)	87.60 ($\frac{\$}{\text{year}}$)	0.03 ($\frac{\$}{\text{hour}}$)
Computing instance unit cost (P_c)	147.12 ($\frac{\$}{\text{year}}$)	0.02 ($\frac{\$}{\text{hour}}$)	72.51 ($\frac{\$}{\text{year}}$)	0.05 ($\frac{\$}{\text{hour}}$)
Storage unit cost (P_s)	1.20 ($\frac{\$}{GB}$)	-	0.56 ($\frac{\$}{GB}$)	-
Communication unit cost (P_{IO})	0.0000002 ($\frac{\$}{\text{hour}}$)	0.20 ($\frac{\$}{\text{hour}}$)	0.0016 ($\frac{\$}{\text{hour}}$)	0.00000001 ($\frac{\$}{\text{hour}}$)
Computing request rate (R_c)	4431 ($\frac{1}{\text{hour}}$)	4431 ($\frac{1}{\text{hour}}$)	3000 ($\frac{1}{\text{hour}}$)	3000 ($\frac{1}{\text{hour}}$)
DB request rate (R_{db})	5282 ($\frac{1}{\text{hour}}$)	5282 ($\frac{1}{\text{hour}}$)	2100 ($\frac{1}{\text{hour}}$)	2100 ($\frac{1}{\text{hour}}$)

In each iteration of the dynamic provision phase of DCRA algorithm, the dynamic cloud provision optimization is formulated and solved. This phase outputs a cost-efficient and robust combination of on-demand cloud resources that meets the web service providers' QoS requirements under user demand uncertainty. This matches the inherent indeterminism and elasticity in cloud web applications and addresses the shortcomings of existing deterministic cloud resource allocation models [14], [25], [31], [32].

With the proposed DCRA algorithm, we are able to minimize deployment costs of web application in the cloud, even with demand uncertainty. With the cloud resource reservation optimization, reservation cost is minimized as the minimum expected demands are assumed to be known a-priori. Then, the additional on-demand resources can be provisioned instantly with the dynamic cloud provision optimization when the reserved cloud resources are insufficient to accommodate the incoming demands.

5 Performance Analysis

In this section, the merits of the proposed algorithm are illustrated by analyzing the experimental results.

5.1 Experimental Setup

In our evaluation, any cloud computing environment can be used but we use two popular cloud providers, Amazon Web Services (AWS) [7] and Microsoft Azure [6], to acquire cloud resources for web application deployment. Both reserved and on-demand instances from AWS and Microsoft Azure are considered for resource provisioning. Under the uncertainty of users' demands, a combination of reserved and on-demand resources is employed to deploy the cloud web application and satisfy the QoS constraints. The experiments were run on an OS X El Capitan machine with 8 GB of memory and 1.6 GHz Intel Core i5 processor. We have developed a limited cloud environment simulator to test our proposed algorithm. The proposed DCRA algorithm as well as the experimental algorithms are developed using C++ programming language. To solve the optimization problems formulated as part of the proposed algorithm, Mosek 6.0 [37] solver is used.

Simulation setup is developed in C++ environment. The arrival rate of user demands is modeled based on the queueing theory-based model presented in [31]. To model cloud resource behaviors, the providers' performance data from the cloud providers' web sites [6], [7] are used. Moreover, the average time to respond to a single user request is derived based on the providers' specifications of resources as well as the average data transfer size of typical computing and database user requests for the given web application workload scenario.

AWS and Microsoft Azure provide reserved and on-demand instances with different characteristics and pricing strategies. In Table 1, pricing of the resources offered by these cloud providers is summarized. As shown in this table, the reserved resources and on-demand resources pricing from AWS provider are presented in columns 2 and 3 and from Microsoft Azure provider are in columns 4 and 5, respectively. In this paper, Amazon EC2 general purpose t2.small [8] and Amazon RDS standard instance db.t2.small [38] are considered as computing and database instances, respectively. From Microsoft Azure provider, standard virtual machines D_1 instance [39] and storage optimized virtual machine L_4 instance [40] are used as computing and database instances, respectively. These instances are used since they are relatively comparable instances from the performance/cost point of view and enable us to fairly compare between the solutions in the two different cloud environments. However, any instance types from any cloud provider can be used depending on the web service providers' preferences. For reserved resources, both AWS and Microsoft Azure offer one-year and three-year contracts. In this experiment, the price of a one-year contract is used for the reserved instances, while on-demand resources are charged hourly. In rows 2 and 3, the price of database and computing instances are presented. As storage requirements are rather static than dynamic, the storage is only considered for reserved plans based on the provider characteristics. Therefore, the cost of a one-year contract of storage per GB is presented. In row 5, communication costs for reserved and on-demand instances are given. The service rates, i.e. the number of requests a virtual machine can handle per hour, for a single computing instance and a single database instance are presented in rows 6 and 7, respectively. These service rates are calculated by dividing the instance dedicated throughput by the average file size for a user request which is the average amount of data that should be processed for a user request.

As seen in Table 1, the selected AWS instances offer a relatively higher performance compared to the selected instances from Microsoft Azure. However, the one-year contract cost of the AWS reserved instances are more expensive compared to Microsoft Azure. For on-demand resources, the hourly cost for the AWS database instances are higher than Microsoft Azure, but the hourly cost of the

computing instance for Microsoft Azure is higher than the AWS. Therefore, the selected instances from the two cloud providers are relatively comparable when considering both performance and reserved/on-demand costs. This enables us to have a fair comparison of the solutions in the two cloud environments.

The workload models that are considered in these experiments are divided into two distinct workload benchmarks. Each of these workload benchmarks has different characteristics, demands and QoS constraints. These benchmarks are implemented as the web services of the real workloads [41]. The first workload benchmark (RUBiS) is based on a simulator that characterizes an auction website [41]. To emulate the real users' behavior for RUBiS workloads, a client browser emulator is used. In [41], to extract a set of workload benchmarks (SPEC), a simulator is generated to produce a multi-tier application server. In this server, the simulator considers the direct communications of a web application with the application servers in a data center. To show the effectiveness of our proposed algorithm, we scaled seven different scenarios from these workloads to represent web applications with large-scale demands. Table 2 lists the required resources and QoS constraints for each scenario.

TABLE 2
The input parameters from seven different workload scenarios.

Scenario	s (GB)	$r_{min_{db}}$ $(\frac{1}{hour})$	r_{min_c} $(\frac{1}{hour})$	$\mathrm{t_{db}}$ (s)	$\mathrm{t_c}$ (s)
RUBiS800	320	32616	7060	4.00	2.70
RUBiS1600	320	37040	25360	4.50	2.90
RUBiS2400	320	40220	63920	5.40	3.30
RUBiS3200	320	42940	202920	6.80	4.50
SPEC10	760	69540	34600	3.20	3.60
SPEC20	760	71260	37180	5.40	4.10
SPEC40	760	73820	37260	6.30	5.30

Column 2 of Table 2 presents the required storage for each scenario. In columns 3 and 4, the web service providers' minimum database and computing requests are shown. The maximum response time to satisfy web service providers' QoS criteria for database and computing instances are given in columns 5 and 6.

Additional requirements for dynamic provision phase are explained in Table 3. These data have been extracted by fitting the workloads' data to a normal distribution to simulate the uncertainties in the user demands. The extracted mean (columns 2 and 3) and standard deviation (columns 4 and 5) of the fitted normal distribution are used as inputs to the dynamic provision phase.

TABLE 3
The input expected values and standard deviations of user demands for the workload scenarios to be used in the dynamic provision phase.

Scenario	$r_{ave_{db}}$ $(\frac{1}{hour})$	r_{ave_c} $(\frac{1}{hour})$	$\sigma_{r_{db}}$ $(\frac{1}{hour})$	σ_{r_c} $(\frac{1}{hour})$
RUBiS800	46594.2	10099.2	1630.8	353.5
RUBiS1600	52912.6	36230.0	1851.9	1268.1
RUBiS2400	57681.2	91317.6	2018.8	3196.0
RUBiS 3200	61334.0	289885.6	2146.7	10146.0
SPEC10	99340.2	49414.8	3476.91	1729.5
SPEC20	101808.2	53112.0	3563.3	1858.9
SPEC40	105462.0	53219.2	3691.2	1862.7

TABLE 4
The solutions of the proposed DCRA algorithm for deployment in AWS cloud environment.

Scenario	Reservation Phase			Dynamic Provision Phase	
	α_c^r $(\#Inst)$	β_{db}^r $(\#Inst)$	μ_s^r (GB)	α_c^o $(\#Inst)$	β_{db}^o $(\#Inst)$
RUBiS800	2	7	45.7	1	4
RUBiS1600	6	8	40.0	4	4
RUBiS2400	15	8	40.0	8	5
RUBiS3200	45	9	35.5	25	5
SPEC10	8	14	22.8	5	7
SPEC20	9	14	22.8	5	8
SPEC40	9	14	22.8	5	8

TABLE 5
The solutions of the proposed DCRA algorithm for deployment in Microsoft Azure cloud environment.

Scenario	Reservation Phase			Dynamic Provision Phase	
	α_c^r $(\#Inst)$	β_{db}^r $(\#Inst)$	μ_s^r (GB)	α_c^o $(\#Inst)$	β_{db}^o $(\#Inst)$
RUBiS800	4	11	29.1	2	6
RUBiS1600	13	13	24.6	7	7
RUBiS2400	31	14	22.8	17	8
RUBiS3200	97	15	21.3	52	8
SPEC10	17	24	13.3	9	13
SPEC20	18	24	13.3	10	13
SPEC40	18	25	12.8	10	14

The cloud resource allocation solutions of the proposed DCRA algorithm are compared with two existing works, QCost algorithm [25] and RCRP algorithm [9]. The QCost algorithm, which is a single phase algorithm, is chosen to show the effectiveness of our first contribution to exploit a combination of reserved and on-demand instances in DCRA algorithm. The other algorithm we compare our work with is RCRP algorithm which is the most comprehensive existing work on dynamic provisioning. This algorithm benefits from the advantages of both reserved resources and on-demand resources which makes it a fair choice to compare our proposed DCRA algorithm with. The performance comparisons are performed based on several metrics, e.g. the total deployment costs, QoS, and runtime. The intermediate solutions of each phase as well as the overall results are analyzed in detail in the following sections.

5.2 Numerical Results

Several experiments have been carried out on the mentioned workload scenarios to analyze the effectiveness of the proposed algotithm. We start with evaluating the first contribution of this paper that is exploiting combination of reserved and on-demand resources in the DCRA algorithm.

The solutions of the proposed DCRA algorithm for seven scenarios are presented in Table 4 and Table 5 for deployment in AWS and Microsoft Azure cloud environments, respectively. These tables present the DCRA solutions for the resources allocated during the reservation phase and dynamic provision phase separately. In these tables, the number of required computing and database instances allocated by the reservation phased are shown in Columns 2 and 3 while the number of computing and database instances secured by the dynamic provision phase are shown in Columns 5 and 6. The required amount of storage for each

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2897304, IEEE Transactions on Cloud Computing

TABLE 6
Evaluation of the solutions of the proposed DCRA algorithm in terms of total deployment costs, and QoS for deployment in AWS cloud environment and Microsoft Azure cloud environment.

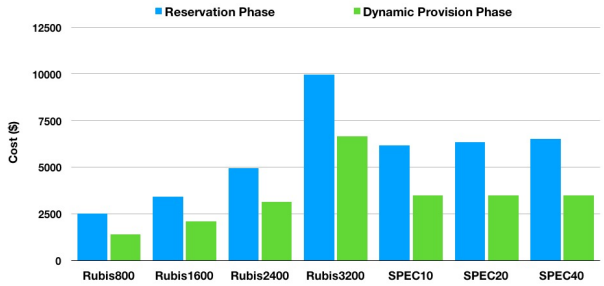
Scenario	AWS cloud provider			Microsoft Azure cloud provider		
	TC ($)	QoS_{db} (s)	QoS_{c} (s)	TC ($)	QoS_{db} (s)	QoS_{c} (s)
RUBiS800	3930.94	0.99	2.24	3543.37	1.11	2.85
RUBiS1600	5527.36	0.90	1.21	6792.56	1.10	1.36
RUBiS2400	8111.06	0.29	0.18	12488.25	0.93	0.58
RUBiS3200	16631.51	0.79	0.17	33011.27	0.87	0.18
SPEC10	9669.32	0.50	0.95	10826.59	0.54	1.06
SPEC20	9859.09	0.50	0.91	11322.82	0.54	1.00
SPEC40	10010.57	0.50	0.92	11754.82	0.52	1.02

reserved database instance is presented in Column 4. As seen in these tables, the number of required computing and database instances in Microsoft Azure are generally larger than those in AWS environment. This is expected because the selected instances in Microsoft Azure have a lower rate of request handling than the selected AWS instances.

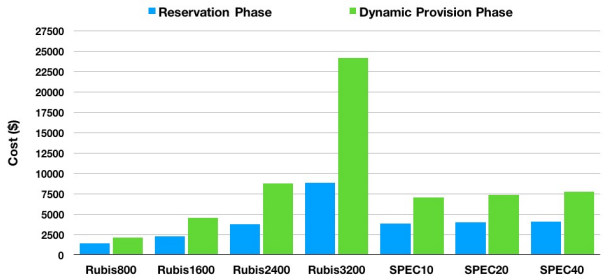
One main contribution of this paper is developing the reservation phase and dynamic provision phase and integrating them to achieve a hybrid solution of reserved and on-demand resources. To evaluate these solutions, the costs of reservation and dynamic provision phases for both AWS and Microsoft Azure cloud providers are compared in Figure 3(a) and Figure 3(b). As expected, the costs of DCRA solutions are different since cloud providers offer different pricing models and performance characteristics for their resources. As shown in Figure 3(a), since the unit cost of the on-demand resources are higher than reserved resources for deployment in AWS cloud provider, the DCRA algorithm finds solutions that invest more in reserved resources to decrease the total deployment costs. In Figure 3(b), the comparison of the reserved and dynamic costs of Microsoft Azure represents that the DCRA selects a relatively larger number of on-demand resources to minimize the total deployment costs which is due to the different pricing model. The integration of reservation phase and dynamic provision phase is important for the web service providers as it enables them to exploit both pricing plans and their flexibilities. The largest total deployment costs are observed for the RUBiS3200 workload scenario where, the DCRA selects 54 reserved instances and 30 on-demand instances for deployment in AWS, while in Microsoft Azure, 112 reserved and 60 on-demand instances are used.

To compare the results of the DCRA algorithm for AWS and Microsoft Azure cloud providers, the total deployment cost of the DCRA algorithm that is the summation of the cost of reserved and dynamic provision phases are shown in Columns 2 and 5 of Table 6. The database QoS and computing QoS for the DCRA algorithm are presented in columns 3 and 6, and 4 and 7, respectively, in terms of instance response times. The instance response times demonstrate that the achieved QoS for database and computing instances are way below the maximum allowed response times and meet the QoS requirements set by the web service providers.

To better illustrate the effectiveness of our proposed two-phase DCRA algorithm, a recent existing algorithm that solves the resource allocation problem in one single phase, QCost [25], is considered. To have a fair comparison and in order to enable QCost to handle the dynamic user demands, the QCost algorithm and DCRA's dynamic provision phase are run for 15-minute periods. Therefore, for QCost algorithm, only one type of cloud resources, i.e. on-demand resources, can be considered.



(a) Deployment in AWS

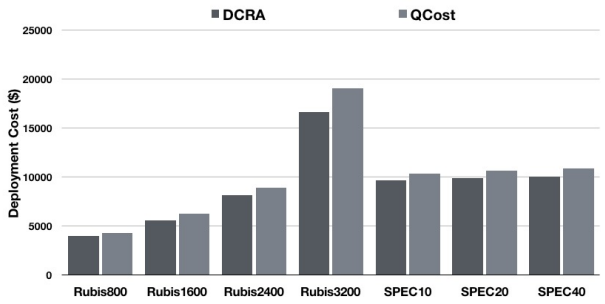


(b) Deployment in Microsoft Azure

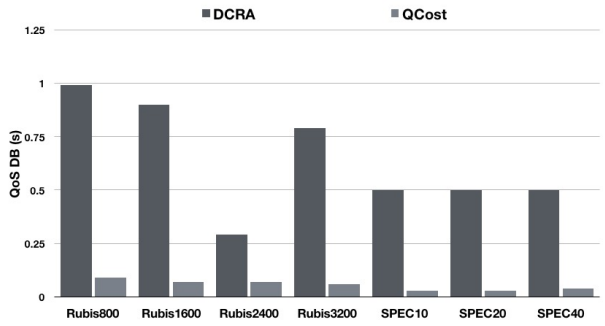
Fig. 3. Comparing the costs of the DCRA algorithm's reservation phase and dynamic provision phase solutions.

We use QCost and DCRA algorithms to solve the cloud resource allocation problem for the workload benchmarks. In Figure 4, the solutions from the proposed DCRA algorithm and QCost algorithm are evaluated. In Figure 4(a) the DCRA and QCost are compared in terms of total deployment cost. Response times (QoS criteria) of database and computing instances are presented in Figures 4(b) and 4(c), respectively. The evaluations in terms of total deployment cost clearly show that QCost over-provisions resources to handle the user demands. This can be observed by considering the QoS criteria of the QCost solutions which are significantly lower than the QoS constraints set by the web service providers in Table 2. DCRA algorithm on the other hand, provides more economical solutions to the web service providers by exploiting a combination of cheap but less flexible reserved resources with expensive pay-as-you-

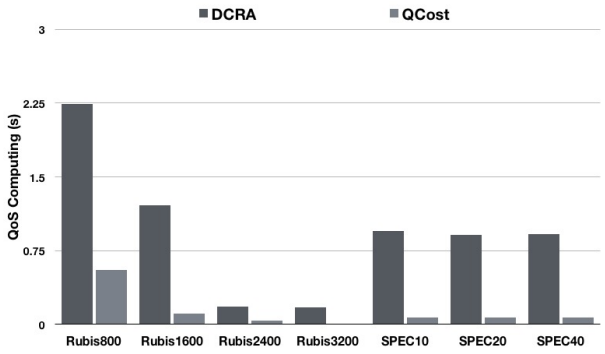
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2897304, IEEE Transactions on Cloud Computing



(a) Total Deployment Cost



(b) Database Response Time



(c) Computing Response Time

Fig. 4. Comparing the solutions of the proposed DCRA algorithm and QCost algorithm.

go on-demand resources.

The results show a modest reduction in total deployment cost by avoiding over-provisioning while DCRA solutions still comfortably meet the QoS constraints. Although when comparing the QoS criteria, the DCRA solutions may seem inferior compared to the QCost solutions, as we will show in the next section, they are more robust to the demand uncertainties and meet the QoS constraints in a significantly larger number of uncertainty realizations. The largest improvement in total deployment cost is achieved for RUBiS3200 (15%). For this scenario, the DCRA achieves the database QoS of 0.79s and the computing QoS of 0.17s which are lower than the allowed maximum times set by the web service provider, 6.80s and 4.50s for database and computing instances, respectively. The proposed DCRA on average obtains 10% improvement in total deployment cost compared to QCost.

In the next experiment, to show the effectiveness of the proposed DCRA algorithm in provisioning resources dynamically, a comparison among the proposed DCRA and one of the pioneer existing works that considers dynamic resource provisioning is performed and the results are presented in Table 7. This existing work is known as the robust cloud resource allocation (RCRP) algorithm [9]. The RCRP algorithm models the resource allocation problem as a robust optimization problem to deal with the demand uncertainties. Also, both reserved and on-demand virtual instances are considered in RCRP algorithm to exploit both pricing plans.

As shown in Table 7, the cost of RCRP algorithm's solution is consistently higher than the cost of the proposed DCRA algorithm's solution in all the scenarios (22.7% on average). The reason is that the RCRP algorithm is modeled as a robust optimization problem taking a pessimistic approach. Therefore, to provide reliable solutions, the worst-case scenarios are considered by the RCRP algorithm leading to relatively higher cost compared to the proposed DCRA algorithm. On the other hand, our proposed DCRA algorithm employs a stochastic approach to consider the demand uncertainties as random variables and optimizes based on the distribution rather than the worst-case scenarios. Therefore, it reduces the cost of virtual machine instances while still achieving reliable solutions. Although the DCRA's response times are slightly larger than RCRP by around 1.5% for database and 2% for computing instances, they are still well below the web service provider's QoS requirements.

TABLE 7
Comparison between the solutions of the proposed DCRA algorithm and the RCRP algorithm.

Scenario	DCRA algorithm			RCRP algorithm		
	TC ($)	QoS_{db} (s)	QoS_{c} (s)	TC ($)	QoS_{db} (s)	QoS_{c} (s)
RUBiS800	3930.94	0.99	2.24	4075.57	0.90	2.10
RUBiS1600	5527.36	0.90	1.21	6266.92	0.90	1.15
RUBiS2400	8111.06	0.29	0.18	10469.64	0.36	0.23
RUBiS3200	16631.51	0.79	0.17	24691.68	0.82	0.24
SPEC10	9669.32	0.50	0.95	10694.91	0.45	0.91
SPEC20	9859.09	0.50	0.91	10891.08	0.47	0.93
SPEC40	10010.57	0.50	0.92	11155.73	0.50	0.95

5.3 Demand Uncertainties

Another major contribution of this paper is developing a stochastic optimization approach to handle the uncertainties in the user demands. The goal is to obtain cloud resource allocation solutions that are robust to the changes in the user demands. Therefore, DCRA models the demands as random variables that follow normal distributions. Taking a pessimistic but reliable approach, we optimize for a 2σ ($2 \times$ standard deviation) range when solving the dynamic cloud provision optimization to guarantee meeting the constraints for around 95% of realizations of the uncertain user demands [36]. In this section, we have conducted a series of experiments to verify the effectiveness of this contribution by comparing DCRA with the QCost as a deterministic optimization approach where user demands are assumed to be constant values during optimization.

We perform Monte Carlo simulations [42] to simulate real-world scenarios where user demands change over the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2897304, IEEE Transactions on Cloud Computing

TABLE 8
The trade-off between total on-demand costs and solution robustness for 1000 Monte Carlo simulations.

Scenario	1σ		2σ		3σ		4σ		5σ		6σ	
	#Viols	Cost($)	#Viols	Cost($)	#Viols	Cost($)	#Viols	Cost($)	#Viols	Cost($)	#Viols	Cost($)
RUBiS800	145	1224	20	1366	3	1398	0	16124	0	1752	0	1887
RUBiS1600	153	1782	23	1997	5	2167	2	2348	0	2558	0	2726
RUBiS2400	145	2796	25	3083	10	3355	2	3504	1	3960	0	4194
RUBiS3200	162	5941	29	6307	14	6989	3	7709	1	8410	0	9086
SPEC10	159	3075	21	3153	4	3495	0	3854	0	4380	0	4543
SPEC20	174	3180	24	3504	7	3844	1	4205	0	4555	0	4892
SPEC40	162	3285	25	3504	11	4194	1	4556	0	4660	0	5242

deployment time. Monte Carlo simulations are often used to evaluate the solutions of the problems with uncertain nature [17], [28], [43]. By running a large number of Monte Carlo simulations (e.g. 1000 times), we can evaluate the performance of both DCRA and QCost algorithms in presence of demand uncertainties. Both algorithms are tested using the experimental algorithm presented in Algorithm 1.

Algorithm 1 Monte Carlo Simulations

Input: Cloud resource optimization solution, r_{ave} and σ
Output: #demandViols and #QoSViols

1: set #experiments 0
2: set #demandViols 0
3: set #QoSViols 0
4: **while** #experiments $<= 1000$ **do**
5: generate a random demand following a normal distribution with mean r_{ave} and std. dev. σ
6: **if** web service provider demand constraints violated **then**
7: #demandViols++
8: **if** web service provider QoS constraints violated **then**
9: #QoSViols++
10: #experiments++
11: **return** #demandViols and #QoSViols

The input parameters to the Monte Carlo Simulations algorithm are the cloud resource optimization solution of DCRA algorithm or QCost. As user demands to the web application are random variables following normal distributions, the demand's mean and standard deviation are also considered as the input parameters to the algorithm presented in Table 3. In the first step of Algorithm 1, the number of performed experiments is initialized to 0. Similarly, the number of violated demand and QoS constraints are initialized to 0 in steps 2 and 3. In each of the 1000 iterations of the while loop in step 4, a Monte Carlo simulation is run. For each simulation, first a random demand is generated according to the given random variable's distribution (step 5). Then, under the generated demand, we evaluate and check if the solution meets the user demand and QoS constraints, in steps 6 and 8. The number of uncertainty realizations that lead to violated constraints is recorded and output as a measure to evaluate the performance of a solution under demand uncertainties. The lower the number of the violations are the more robust the solution is to the uncertainties in the user demands.

The results of running Monte Carlo simulations of Algorithm 1 for the solutions of the proposed DCRA and the QCost algorithms are compared in Figure 5. In this figure, we can observe the effectiveness of our stochastic optimization approach to make the DCRA solutions more robust to the demand uncertainties compared to a deterministic algorithm such as QCost. The QCost solutions violate the constraints in around 50% of the realizations of the uncertain demand. DCRA solutions however only fail to meet the constraints in about 2% of the realizations which demonstrate the solution's robustness. This significant robustness improvement is achieved while the DCRA solutions also have lower deployment costs as they avoid over-provisioning.

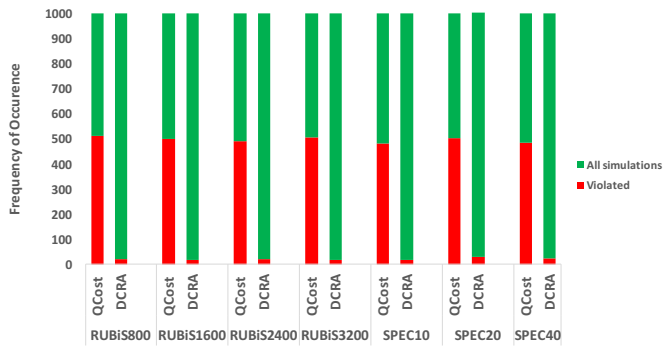


Fig. 5. Comparison of the results of Monte Carlo simulations (Algorithm 1) for the solutions from the proposed DCRA and the QCost algorithms.

Based on the web service provider's preference, our DCRA algorithm can be modified to take a more pessimistic or less pessimistic approach. However, there is indeed a trade-off between the on-demand resource costs and robustness of the achieved solutions. To demonstrate this trade-off, in Table 8, we present the results of an experiment where different multipliers for the standard deviation σ are used to see how they impact the total on-demand costs and the number of constraint violations. In this experiment, we use the multipliers 1 (68% distribution coverage), 2 (95% distribution coverage), 3 (97% distribution coverage), 4 (99% distribution coverage), 5 (almost 100% distribution coverage), and 6 (almost 100% distribution coverage) for the standard deviation σ. According to this table, our chosen default multiplier, 2, provides a relatively balanced trade-off between the cost and the robustness of the solution. However, it is a web service provider's decision to choose the best trade-off for the application based on the criticality and budget availability.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2897304, IEEE Transactions on Cloud Computing

TABLE 9
The solution of the proposed DCRA algorithm for PYXIS WorldView [44] deployment in Microsoft Azure cloud.

Scenario	Reservation Phase			Dynamic Provision Phase		TC ($)	QoS_{db} (s)	QoS_c (s)
	α_c^r ($\#Instance$)	β_{db}^r ($\#Instance$)	μ_s^r (GB)	α_c^o ($\#Instance$)	β_{db}^o ($\#Instance$)			
PYXIS Worldview	13	10	42	7	6	11327.71	0.7	1.2

TABLE 10
The input parameters from PYXIS WorldView [44] workload testcase.

Scenario	s (GB)	$r_{min_{db}}$ ($\frac{1}{hour}$)	r_{min_c} ($\frac{1}{hour}$)	t_{db} (s)	t_{c} (s)
PYXIS Worldview	420	30000	30000	6.0	3.0

5.4 Industry Case Study

In order to show the consistency between the performance of the proposed algorithm on the simulated workload scenarios and real-world testcases, an industrial workload testcase of the PYXIS WorldView from the PYXIS Innovation Inc. [44] is used in this experiment. PYXIS WorldView uses Microsoft Azure cloud as the public cloud provider. The input parameters from this workload testcase are presented in Table 10. The proposed DCRA algorithm is used to dynamically allocate resources for this workload testcase and the results are reported in Table 9. The DCRA algorithm allocates 10 reserved database and 13 reserved computing instances. It then allocates 6 on-demand database and 7 on-demand computing instances to handle the actual incoming demands in the testing period. During this deployment time the database and computing response times are well below the given limits with the maximum of 0.7s and 1.2s, respectively. Note that the reported total deployment costs in Table 9 are the estimated total deployment costs for one year of web application deployment which is below the allocated budget limit.

5.5 Runtime

In dynamic resource allocation, the turnaround time is critical due to the elasticity of the cloud environment. For our proposed DCRA algorithm to be practically used for dynamic resource allocation, it should maintain a low runtime. To visualize the runtime impact of solving resource allocation problem in two steps, in Figure 6, the runtime of the DCRA algorithm is compared to the RCRP algorithm and QCost algorithm.

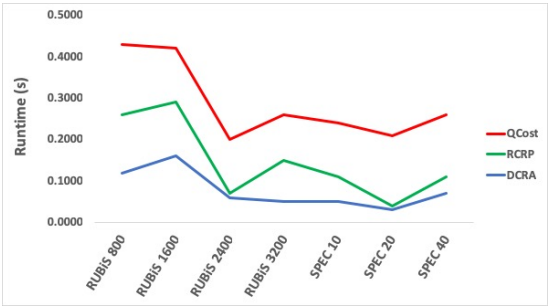


Fig. 6. The runtime comparison between the proposed DCRA algorithm, RCRP algorithm and the QCost algorithm.

As seen in this figure, for both algorithms there is not an obvious counterbalance between user demands increase and the runtime. This is because, the size of the demands does not affect the number of variables and constraints of our proposed optimization problems. The runtime to solve a convex optimization problem depends on the number of problem variables and constraints which are constant for our proposed algorithm and do not change with the size of user's demands. However, this runtime also depends on the tightness of the problem's constraints which explains the runtime changes in this figure. Although the DCRA algorithms includes two steps, we only go through one step at a time and we only solve one optimization problem (either cloud resource reservation optimization or dynamic cloud provision optimization) each time. Our DCRA runtime is lower than QCost for every single workload scenario, on average 45% improvement. DCRA runtime is also shorter than the RCRP. This shows that the proposed DCRA can dynamically allocate resources to a web application considering the average runtime is 0.07s.

6 Conclusion

A cost-efficient dynamic provisioning algorithm for cloud-based web applications in cloud environment is proposed that focuses on optimizing the total provisioning costs while considering the uncertainties in the user demands. The proposed DCRA is modeled in two-phases: reservation and dynamic provision phases. To evaluate the performance of DCRA, simulations have been performed for different workload scenarios. The results show that the proposed algorithm can achieve reliable and cost-effective solutions using a dynamic combination of reserved and on-demand resources for deploying cloud-based applications. The combination of solutions of both reservation and dynamic provision phases saves the total provisioning costs significantly. In addition, the proposed DCRA is cloud provider independent and can be used for major cloud providers such as AWS [7], Microsoft Azure [6], and GoGrid [30]. The future work may be expanding the DCRA to dynamically allocate resources to applications from multiple cloud providers to achieve lower costs and better redundancy. Moreover, an integrated monitoring procedure can be added into DCRA algorithm.

7 Appendix

7.1 Geometric Programming

Geometric Programming (GP) is a class of optimization problems with the general form as [45]:

$$\begin{aligned} \min \quad & f_0(\mathbf{x}) \\ \text{s.t.} \quad & f_i(\mathbf{x}) \leq 1, \ i = 1, \ldots, n \\ & h_i(\mathbf{x}) = 1, \ i = 1, \ldots, m \end{aligned} \tag{17}$$

if the objective function, $f_0(\cdot)$ and inequality constraint functions $f_i(\cdot)$ are posynomials and equality constraint functions $h_i(\cdot)$ are monomials. A monomial is a function in the form of:

$$g(\mathbf{x}) = cx_1^{\alpha_1}x_2^{\alpha_2}\cdots x_n^{\alpha_n}, \tag{18}$$

where the coefficient c is a positive real number, exponents, $\alpha_1, \ldots, \alpha_n$, are real numbers and $x_1, x_2, \ldots, x_n$ are non-negative variables. Then, a posynomial function is defined as a summation of monomials.

The advantage of GP is that it can handle non-linear functions encountered in complex engineering problems. Furthermore, although GP is not a convex optimization problem, it can be converted to a convex form [45].

7.2 Stochastic Optimization

Although there are uncertainties in variable and parameters of many optimization problems, usually, the variables and parameters are assumed as deterministic values in the problem modeling. The classic optimization techniques achieve nominal solutions and ignore uncertainties. The obtained nominal solutions can lead to infeasibility and constraint violation because of the parameters' fluctuation. Stochastic optimization is a popular approach in most of the applications that involve uncertainties and indeterministic variables. Stochastic optimization provides optimal solutions considering the parameters and variables with uncertainties as random variables. In stochastic optimization, the variables and parameters' uncertainties are characterized by specific probability distributions. The general form of a single variable linear programming problem is expressed as:

$$\begin{aligned} \min. \quad & h(x) \\ \text{s.t.} \quad & k(x) \leq 0 \end{aligned} \tag{19}$$

where the problem objective is $h(.)$, $k(.)$ represents the constraints and x is the problem variable. Then, a stochastic optimization formulation can be written as:

$$\begin{aligned} \min. \quad & E[h(\widetilde{x})] \\ \text{s.t.} \quad & E[k(\widetilde{x})] \leq 0 \end{aligned} \tag{20}$$

where E[.] is the expected value operator. This stochastic optimization formulation minimizes the expected value of the objective function by considering the probability density function of the random variable $\widetilde{x}$.

Although robust optimization can also be used in applications that include uncertainties, it is a pessimistic approach that considers the worst-case realizations of the uncertainties. However, stochastic optimization finds the more realistic and general solutions by imposing the probability distributions that parameters follow. Therefore, stochastic optimization is more effective when the uncertainties are probabilistic and a distribution is known for them.

ACKNOWLEDGMENTS

This research is supported by the Natural Sciences and Engineering Research Council of Canada and Alberta Innovates Technology Futures.

REFERENCES

[1] T. Chen, Y. Zhang, X. Wang, and G. B. Giannakis, "Robust workload and energy management for sustainable data centers," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 651–664, March 2016.

[2] M. Anastasopoulos, A. Tzanakaki, and D. Simeonidou, "Stochastic energy efficient cloud service provisioning deploying renewable energy sources," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3927–3940, Dec 2016.

[3] A. Johannes, N. Borhan, C. Liu, R. Ranjan, and J. Chen, "A user demand uncertainty based approach for cloud resource management," in *2013 IEEE 16th International Conference on Computational Science and Engineering*, Dec 2013, pp. 566–571.

[4] S. Hosseinalipour and H. Dai, "Options-based sequential auctions for dynamic cloud resource allocation," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[5] Rackspace. https://www.rackspace.com.

[6] Microsoft Azure. http://azure.microsoft.com/.

[7] Amazon web services. http://aws.amazon.com/.

[8] AWS Amazon EC2. https://aws.amazon.com/ec2/pricing/on-demand/.

[9] S. Chaisiri, B. S. Lee, and D. Niyat, "Robust cloud resource provisioning for cloud computing environments," in *2010 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, Dec 2010, pp. 1–8.

[10] N. Sfika, A. Korfiati, C. Alexakos, S. Likothanassis, K. Daloukas, and P. Tsompanopoulou, "Dynamic cloud resources allocation on multidomain/multiphysics problems," in *2015 3rd International Conference on Future Internet of Things and Cloud*, Aug 2015, pp. 31–37.

[11] R. I. Meneguette, A. Boukerche, A. H. M. Pimenta, and M. Meneguette, "A resource allocation scheme based on semi-markov decision process for dynamic vehicular clouds," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[12] S. Imai, S. Patterson, and C. A. Varela, "Cost-efficient elastic stream processing using application-agnostic performance prediction," in *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, May 2016, pp. 604–607.

[13] L. Jiao, J. Li, T. Xu, W. Du, and X. Fu, "Optimizing cost for online social networks on geo-distributed clouds," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 99–112, Feb 2016.

[14] H. Goudarzi, M. Ghasemazar, and M. Pedram, "SLA-based optimization of power and migration cost in cloud computing," in *the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, May 2012, pp. 172–179.

[15] W. K. Tan, D. M. Divakaran, and M. Gurusamy, "Uniform price auction for allocation of dynamic cloud bandwidth," in *2014 IEEE International Conference on Communications (ICC)*, June 2014, pp. 2944–2949.

[16] J. Chase and D. Niyato, "Joint optimization of resource provisioning in cloud computing," *IEEE Transactions on Services Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[17] S. Chaisiri, B. S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Transactions on Services Computing*, vol. 5, pp. 164–177, April 2012.

[18] J. N. Khasnabish, M. F. Mithani, and S. Rao, "Tier-centric resource allocation in multi-tier cloud systems," *IEEE Transactions on Cloud Computing*, vol. 5, no. 3, pp. 576–589, July 2017.

[19] B. Neethu and K. R. R. Babu, "Dynamic resource allocation in market oriented cloud using auction method," in *2016 International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE)*, Sept 2016, pp. 145–150.

[20] S. Chaisiri, R. Kaewpuang, B. S. Lee, and D. Niyato, "Cost minimization for provisioning virtual servers in amazon elastic compute cloud," in *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, July 2011, pp. 85–95.

[21] Y. Ran, Y. Shi, E. Yang, S. Chen, and J. Yang, "Dynamic resource allocation for video transcoding with qos guaranteeing in cloud-based dash system," in *2014 IEEE Globecom Workshops (GC Wkshps)*, Dec 2014, pp. 144–149.

[22] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *IEEE INFOCOM*, March 2010, pp. 1–9.

[23] C. T. Do, D. T. Do, N. H. Tran, D. H. Tran, K. Thar, and C. S. Hong, "Optimal resource allocation for multimedia application in single and multiple cloud computing service providers," in *16th Asia-Pacific Network Operations and Management Symposium*, Sept 2014, pp. 1–4.

[24] H. Goudarzi, M. Ghasemazar, and M. Pedram, "Sla-based optimization of power and migration cost in cloud computing," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID 2012)*, May 2012, pp. 172–179.

[25] S. Mireslami, L. Rakai, M. Wang, and B. H. Far, "Minimizing Deployment Cost of Cloud-Based Web Application with Guaranteed QoS," in *the 2015 IEEE Global Communications Conference (GLOBECOM)*, Dec 2015, pp. 1–6.

[26] M. Anastasopoulos, A. Tzanakaki, and D. Simeonidou, "Stochastic energy efficient cloud service provisioning deploying renewable energy sources," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3927–3940, Dec 2016.

[27] S. Mireslami, L. Rakai, B. H. Far, and M. Wang, "Simultaneous cost and qos optimization for cloud resource allocation," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 676–689, Sept 2017.

[28] Y. Ran, B. Yang, W. Cai, H. Xi, and J. Yang, "Cost-efficient provisioning strategy for multiple services in distributed clouds," in *2016 International Conference on Cloud Computing Research and Innovations (ICCCRI)*, May 2016, pp. 1–8.

[29] L. Yu and H. Shen, "Bandwidth guarantee under demand uncertainty in multi-tenant clouds," in *2014 IEEE 34th International Conference on Distributed Computing Systems*, June 2014, pp. 258–267.

[30] GoGrid. https://www.datapipe.com/gogrid/.

[31] X. Nan, Y. He, and L. Guan, "Optimal Resource Allocation for Multimedia Cloud Based on Queuing Model," in *the 13th IEEE International Workshop on Multimedia Signal Processing (MMSP)*, Hangzhou, China, October 17-19 2011, pp. 1–6.

[32] A. Dastjerdi, S. Garg, and R. Buyya, "QoS-aware deployment of network of virtual appliances across multiple clouds," in *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, November 2011, pp. 415–423.

[33] Microsoft Azure Monitoring. https://docs.microsoft.com/en-us/azure/cloud-services/cloud-services-how-to-monitor.

[34] Z. Cao, J. Lin, C. Wan, Y. Song, Y. Zhang, and X. Wang, "Optimal cloud computing resource allocation for demand side management," *IEEE Transactions on Smart Grid*, vol. PP, no. 99, pp. 1–13, 2016.

[35] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *IEEE INFOCOM*, April 2011, pp. 71–75.

[36] Y. L. Tong, *The Multivariate Normal Distribution*. Springer Series in Statistics, 1990.

[37] Mosek 6.0. http://www.mosek.com.

[38] AWS Amazon RDS. https://aws.amazon.com/rds/pricing/.

[39] Microsoft Azure. https://azure.microsoft.com/en-gb/pricing/calculator/#virtual-machines.

[40] Microsoft Azure. https://azure.microsoft.com/en-gb/pricing/details/virtual-machines/linux/.

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

[42] D. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*. Cambridge University Press, 2005.

[43] L. Rakai, A. Farshidi, D. Westwick, and L. Behjat, "Variation-aware geometric programming models for the clock network buffer sizing problem," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 4, pp. 532–545, April 2014.

[44] Pyxis innovation inc. https://www.pyxisinnovation.com/.

[45] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

Seyedehmehrnaz Mireslami is currently a Ph.D. student in the Department of Electrical and Computer Engineering at the University of Calgary, Canada. She received her B.Sc. degree in Applied Mathematics from Tafresh University, Iran, in 2011. She received her M.Sc. in Electrical Engineering (Software Specialization) from the University of Calgary, Canada, in 2013. Her research interests include cloud computing, Quality of Services (QoS), distributed systems, networking systems and algorithms, and optimization theory.

Logan Rakai Logan Rakai (M'12) received the B.Sc. degree in computer engineering with top honors, in 2007, and the M.Sc. and Ph.D. degrees in electrical engineerxing, in 2008 and 2012, respectively, all from the University of Calgary, Canada. He is currently an Adjunct Professor with the Department of Electrical and Computer Engineering, Schulich School of Engineering, University of Calgary where he strives to bring out the best in students.

He performed his industrial training at Mentor Graphics, Fremont, CA, USA. His current research interests include using distributed and high performance computing, machine learning, and specialized solvers to solve engineering problems in cloud computing, computer-aided design, and power systems.

Mea Wang is currently an Associate Professor in the Department of Computer Science at the University of Calgary. She received her Bachelor of Computer Science (Honours) degree from the Department of Computer Science, University of Manitoba, Canada, in 2002. She received her Master of Applied Science and Ph.D. degrees from the Department of Electrical and Computer Engineering at the University of Toronto, Canada, in 2004 and 2008, respectively. Her research interests include Peer-to-Peer networking, multimedia networking, cloud computing, as well as networking system design and development.

Behrouz Homayoun Far Dr. Far is currently a Professor of Software Engineering at the Schulich School of Engineering, University of Calgary, since 2001. His research interest areas are engineering of intelligent, distributed and heterogeneous networked systems, specifically on design, implementation, testing and verification of agent-based software systems. He established Intelligent Software Systems (ISS) Laboratory of the Schulich School of Engineering, and is co-director of the Intelligent Transportation Lab, University of Calgary.