Hierarchical Stochastic Models for Performance, Availability, and Power Consumption Analysis of IaaS Clouds

Ehsan Ataie, Reza Entezari-Maleki, Leila Rashidi, Kishor S. Trivedi, *Life Fellow, IEEE*
Danilo Ardagna, and Ali Movaghar, *Senior Member, IEEE*

Abstract—Infrastructure as a Service (IaaS) is one of the most significant and fastest growing fields in cloud computing. To efficiently use the resources of an IaaS cloud, several important factors such as performance, availability, and power consumption need to be considered and evaluated carefully. Evaluation of these metrics is essential for cost-benefit prediction and quantification of different strategies which can be applied to cloud management. In this paper, analytical models based on Stochastic Reward Nets (SRNs) are proposed to model and evaluate an IaaS cloud system at different levels. To achieve this, an SRN is initially presented to model a group of physical machines which are controlled by a management layer. Afterwards, the SRN models presented for the groups of physical machines in the first stage are combined to capture a monolithic model representing an entire IaaS cloud. Since the monolithic model does not scale well for large cloud systems, two approximate SRN models using folding and fixed-point iteration techniques are proposed to evaluate the performance, availability, and power consumption of the IaaS cloud. The existence of a solution for the fixed-point approximate model is proved using Brouwer's fixed-point theorem. A validation of the proposed monolithic and approximate models against both an ad-hoc discrete-event simulator developed in Java and the CloudSim framework is presented. The analytic-numeric results obtained from applying the proposed models to sample cloud systems show that the errors introduced by approximate models are insignificant while an improvement of several orders of magnitude in the state space reduction of the monolithic model is obtained.

Index Terms—IaaS cloud, performance, availability, power consumption, stochastic reward nets, fixed-point iteration.

1 Introduction

Cloud computing has become a consolidated computing paradigm, which allows sharing different kinds of resources over the network in an automated way [1], [2]. Infrastructure as a Service (IaaS) providers offer low level computing resources on which end users can run their own software, typically in the form of Virtual Machines (VMs) [3], [4].The use of VMs allows the creation of services that can ask for more or less resources depending on their demand variation in a scalable manner. For instance, by simply adding a new VM to an already existing service, it can process the requests of a larger number of users of that service. Another advantage of using VMs is that they offer the possibility to be transparently migrated from one physical server to another without stopping their execution [5].

- *E. Ataie, L. Rashidi, and A. Movaghar are with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. E-mail: {ataie, rashidy}@ce.sharif.edu; movaghar@sharif.edu*
- *R. Entezari-Maleki is with the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran. E-mail: entezari@ipm.ir*
- *K. S. Trivedi is with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA. E-mail: kst@duke.edu*
- *D. Ardagna is with the Dipartimento di Elettronica, Informazione e Bioingegneria Politecnico di Milano, Milan, Italy. E-mail: danilo.ardagna@polimi.it*

Providers of IaaS clouds offer Service Level Agreements (SLAs) to cloud users [6]. SLAs are contracts between customers and providers that specify the price for a service, the Quality of Service (QoS) levels required during the service provisioning, and the penalties associated with the SLA violations [7], [8]. SLA violations can cause loss of revenue and business reputation for an IaaS provider. For example, Google and Amazon reported significant revenue loss because of very small additional response delays [9]. In such a context, performance evaluation is one of the foremost concerns in cloud computing centers, since it allows system managers to evaluate the effects of different resource management strategies on the data center operation and to predict the corresponding costs and benefits. In addition to performance of cloud systems, availability of cloud resources and services is also considered as one of the most important factors of user satisfaction and provider achievement. In cloud data centers, outages occur for many reasons including software and hardware failures or power outages. In a survey of 200 data centers done by USA Today, it has been reported that the downtime cost of each data center exceeds $\$50,000$ per hour [10]. In another research [11], the IT cost due to hardware repair for a big data center consisting of more than $100,000$ servers has been estimated to be over *one million* dollars per year. Hence, the availability modeling and evaluation in a cloud data center is a key concern that should be considered.

On the other hand, cloud data centers are experiencing a growth rate of around 9% every year [9], and as a result,

their energy demands are continuing to increase as well. It has been estimated that data centers which power Internet-scale applications will consume about 8% of all worldwide electricity supply by 2020 [12]. Furthermore, the power consumption in cloud infrastructures is very inefficient due to several types of energy loss and waste at different components of data centers [13], [14]. Consequently, modeling and quantification of power consumption can help the cloud provider to improve the energy efficiency of the cloud infrastructure and reduce its associated cost.

There are various ways to evaluate the performance of complex and distributed systems, namely measurement, simulation, and analytic modeling [15]–[19]. On-the-field measurement requires extensive experimentations with different workloads and system configurations, which may not be feasible in cloud systems due to the great number of parameters and the large scale of the problem space that should be investigated. Though modeling with simulation is flexible, but since it requires several runs to get an average result, it might be time consuming to gain dependable results. Furthermore, separate runs of the simulation model are needed in order to evaluate the impact of different input parameters, which makes the running times more severe. Using analytical modeling in this context, not only can help providers to attain good insights into the operation of the system including its power, performance- and dependability-related measures in different situations, but it can also be useful in terms of the budget and time limitations. In order to validate a newly proposed approach, based on each of above-mentioned evaluation methods, it suffices to compare the results gained from the new approach to the results obtained from another evaluation method.

Although state-space models are popular analytical models for distributed systems, the growth of the state space as more components or details of the system are taken into account is an important problem which is known as the largeness problem of Markov models [20]. Stochastic Reward Nets (SRNs) [21] can be used to model and evaluate the performability of cloud systems while tolerating the largeness problem, as they allow the automated generation of Markov Reward Models (MRMs). The SRN is an extension of Stochastic Petri Nets (SPNs) which has the advantage of specifying and evaluating real systems in a compact and intuitive way. However, solving large models is still too challenging [8], [16].

In this paper, we consider an IaaS cloud where Physical Machines (PMs) are organized into different groups. The PMs of each group are separated into two different pools based on power consumption and provisioning delays. The proposed method is presented in three steps: In the first step, we model a single group of PMs using SRNs and compute the mean response time, mean percentage of available PMs, and mean power consumption. Then, the SRN models of the individual groups of PMs are combined to develop a monolithic model for an entire IaaS cloud in the second step. Since the numbers of PMs and clusters in real cloud data centers are rather large, the monolithic model is not scalable due to the state explosion of the underlying Markov chain. To resolve this issue, in the third step, two approximate SRN models are proposed. The first approximate model uses folding technique [22], [23] to model an IaaS cloud while the second one uses decomposition approach and fixed-point iteration technique [21], [24] to solve the interacting sub-models. The application of these two techniques shows that the approximate SRN models considerably decrease the number of states in the underlying Markov chain of the SRNs without any significant loss of accuracy. When compared with the state-of-the-art, the proposed fixed-point approximate model provides higher scalability, making it appropriate for modeling large cloud systems. In the fixed-point approximate model, the existence of a fixed-point is proved by using Brouwer's fixed-point theorem. The validation of the proposed models is performed against the discrete-event simulation [25], [26] and the CloudSim framework [27]. Moreover, the sensitivity of output measures to the variation of input parameters is analyzed.

The aim of our modeling approach is to evaluate the performance and power consumption of an IaaS cloud while availability of the resources is taken into account. The innovative contributions of this paper are presenting a series of SRN models for computing the performance, availability, and power consumption of cloud data centers, and providing two approximate models that are scalable enough to assess the metrics of interest in cloud systems. The main advantage of the fixed-point approximate model proposed in this paper is its capability in modeling large clouds without showing any state space explosion.

The remainder of this paper is organized as follows. A literature review in the field of cloud performance and dependability evaluation with a glance on power consumption is given in Section 2. In Section 3, the system description and main assumptions about the reference architecture considered in this paper are introduced. Section 4 describes the proposed model for combined performance, availability, and power consumption analysis of a single cluster. Afterwards, the monolithic and approximate models for entire IaaS cloud are presented in Section 4. In Section 5, some metrics which can be computed by analyzing the proposed models are described, and then numerical results are provided in Section 6. The sensitivity analysis is presented in Section 7, and finally, Section 8 concludes the paper and outlines our future research directions.

2 Related Work

The evaluation of performance, availability, and power consumption through analytical modeling is an emerging topic in the field of cloud computing. In the following, some of the efforts in this area are introduced.

Bruneo et al. [28] have presented a modeling approach based on SRNs to investigate the strategies for managing a federation of clouds aiming to reduce the overall energy consumption. In [28], user requests that cannot be fulfilled locally are redirected to external clouds if a federation policy is active. On the other hand, if a VM consolidation policy is set, a consolidation action can be triggered to migrate executing VMs on the local data center to an available external cloud with the aim of powering off the local center. Different measures including the mean waiting time, mean downtime, mean migration probability, and mean probability that a center is off were evaluated in [28]. The authors do

not address the scalability issue which is the main focus of our work.

Ghosh et al. [6] have proposed multi-level interacting stochastic sub-models to evaluate the performance of request provisioning in large-scale IaaS clouds that offer tiered services by configuring physical machines into three different pools with different provisioning delays: *hot* (running), *warm* (turned on, but not ready), and *cold* (turned off). Dependencies among sub-models were resolved by fixed-point iteration method to cope with the state space explosion problem. Then, the impact of workload and system characteristics on job rejection probability and mean response delay were evaluated. However, the availability and power consumption of servers were not considered.

Ghosh et al. [29] have presented an analytical approach for end-to-end performability analysis of an IaaS cloud service. Authors used interacting stochastic models approach to design performance, availability, and performability sub-models. In pure performance analysis, performance measures are computed under the assumption that there is no failure in resources. For this purpose, three models based on Continuous Timed Markov Chain (CTMC) were created: resource provisioning decision model, VM provisioning model, and run-time model. In pure availability analysis, failures of PMs in different pools have been taken into consideration. Assigning pure performance measures as reward rates to the states of the pure availability model leads to performability analysis. To address scalability issue, the cyclic dependency among sub-models was resolved via fixed-point iteration. Finally, authors evaluated service availability and provisioning response delays as two key QoS metrics in cloud environments. In contrast, for cloud performability analysis, we choose SRN-based models to facilitate automated generation of Markov chains. Furthermore, although joint analysis of performance and availability is addressed in this work, the authors do not deal with power consumption of physical machines. Bruneo [8] has proposed an analytic model based on SRNs to provide a platform for applying and evaluating cloud strategies and policies. VM multiplexing was modeled considering degradation of multiplexed VMs. Moreover, availability and quality metrics were considered in a federated cloud scenario. Afterwards, several metrics including utilization, availability, waiting time, service time, and responsiveness were defined on the proposed model and evaluated. Though the model presented in [8] captures a high-level view of IaaS clouds, it does not encompass details of such cloud systems including VM provisioning and failure/repair behavior of servers. Moreover, power consumption was not considered.

Khazaei et al. [30] studied the steps of servicing an IaaS request and the corresponding delays. Physical machines have been assumed to be categorized into three pools. Like Ghosh et al. [29], the authors exploited CTMCs to model different modules of the system, including resource assignment and virtual machine provisioning modules (one for each pool). Interdependency among sub-models was resolved via fixed-point iteration to decrease the number of states in the resulted state space. They used the term $supertask$ to refer to a set of requests simultaneously submitted by one user each of which requesting a single VM. Effects of various parameters such as arrival rate of supertasks, task service time, and virtualization degree on the supertask rejection probability and response time were studied in [30]. Availability and power consumption were not analyzed. However, the concept of supertask presented in this work can be used to complement our work. In [31], a stochastic model-driven approach have been presented to quantify the availability of an IaaS cloud, where failures were mitigated through migration of PMs among three pools. Since the monolithic model presented in [31] is not scalable for large clouds, the authors proposed an interacting Markov chain-based approach to reduce the complexity of analysis and the solution time. Dependencies among interacting sub-models were resolved using fixed-point iteration technique for which existence of a solution was proved. Furthermore, the results obtained from the numerical analysis of the monolithic and proposed approximate models were compared with each other in terms of the number of states and non-zero entries of the underlying Markov chains, solution time, downtime, and the number of PMs in each pool. Unlike our approach proposed, the requests arrival and servicing process were not dealt with in [31], and the authors only focused on the cloud availability. Ghosh et al. [32] have provided cost analysis by proposing stochastic models for availability and performance of an IaaS cloud system. Physical machines are grouped into three different pools. The failure, repair, and migrations of PMs from one pool to another were considered in the proposed models. The authors studied two cost minimization problems to address the capacity planning in such cloud system. Several cost components were considered such as repair, downtime, and rejection costs. The cost-capacity trade-offs were then translated into mathematical optimization problems. Then, the optimization problems were solved using a stochastic search algorithm, named simulated annealing. In contrast, optimization of output measures is not a concern in our work.

Khazaei et al. [33] have complemented their own previous work [30] by modeling the pool management module which is responsible for moving PMs among pools. Furthermore, they considered the normalized power consumption as one of the evaluation metrics. Availability of resources was not considered in this work, too. In [17] and [34], layered analytical frameworks, based on SRNs, have been proposed to model a green IaaS cloud. The frameworks consist of virtual and physical layers. A sub-model for representing the behavior of each layer was proposed, and the interactions between the sub-models were specified. As a use case, scattering and saturation allocation policies were modeled and compared in terms of several metrics including utilization, power consumption, block probability, and performance degradation. A validation of the proposed model against the CloudSim framework was also presented in [17]. In contrast to our approach, failure/repair behavior of cloud servers was not considered in [17] and [34]. Moreover, the scalability issue was not handled in these studies.

Castiglione et al. [35] have proposed a stochastic modeling approach based on Markovian Agents and Mean Field Analysis to allow the effective description of different concurrent Big Data applications on a same multi-site cloud infrastructure. The information obtained from the proposed stochastic approach can be used to manage and plan the

power distribution within the cloud by exploiting load fluctuations for adapting the energy consumption to the current load. Although all the techniques presented in [35] are approximation techniques, they can provide acceptable solution with orders of magnitude faster computation as the population increases. Details of the system under study in [35] differs from the system we study in this paper. Moreover, the availability of the system is not a concern in [35]. Barbierato et al. [36] have proposed a modeling approach to evaluate the combined effects of cloud elasticity and data management layer on global scale cloud applications. The formalism used in [36] is based on a state space modeling technique, namely Markovian Agents, which uses continuous approximation of Markovian processes. The approach proposed in [36] was applied to a realistic scenario that encompasses both in-memory and in-storage based Big Data applications. In contrast to our work, the focus of this work is on the evaluation of the reliability of the system under study. Ciardo et al. [37] have presented a Petri net based model, named lumped Generalized Stochastic Petri Net (GSPN), to evaluate the power requirements of different allocation strategies for VMs in a data center when computing resources are heterogeneous PMs with various power consumption. The model accounts for allocation/deallocation of VMs on nodes, and uses lumping technique to scale up to the significant data center sizes. In contrast to our work, performance and availability are not considered in this study.

Although there have been proposed several mathematical models exploiting approximation techniques to cope with scalability problem in other distributed systems than clouds [16], [22], [23], the closest work to this study is the work done by Entezari-Maleki et al. [16] in which the performability of grid environments is evaluated by SRNs. To this end, the authors have modeled a single grid resource considering different scheduling schemes to dispatch grid and local tasks to the processors of the resource. Afterwards, the models for individual resources were combined to capture an entire grid environment, and the approximation techniques were applied to decrease the number of states in the underlying Markov chain of the proposed SRN. Even though the work presented in [16] and our work are similar in the formalism and the approximation techniques, there are key differences between them. Most importantly, the contexts of the two works are different distributed systems, so details of the proposed models are thoroughly dissimilar and disjoint. The models presented in [16] were calibrated for a grid resource/environment, and they cannot be used for a cloud system since the requirements and characteristics of the systems are different. Moreover, the input parameters of the models and the measures computed by the models proposed in [16] are specific to grid environments to assess the performability, ignoring the evaluation of the availability and power consumption, which are our main concerns in the proposed models in this paper.

3 System Description

In real cloud systems comprising several thousands of computing servers, centralized management solutions are subject to critical design limitations, including lack of scalability and expensive monitoring costs, and cannot effectively manage available resources [38]. Here, we consider an architecture in which an IaaS cloud provider uses two hierarchical levels of management layers for handling user requests to control a large number of physical servers in an optimized manner. Fig. 1 presents a graphical representation of our reference architecture, where the important components are shown. A similar architecture was exploited in other research proposals [39], [40], [41].

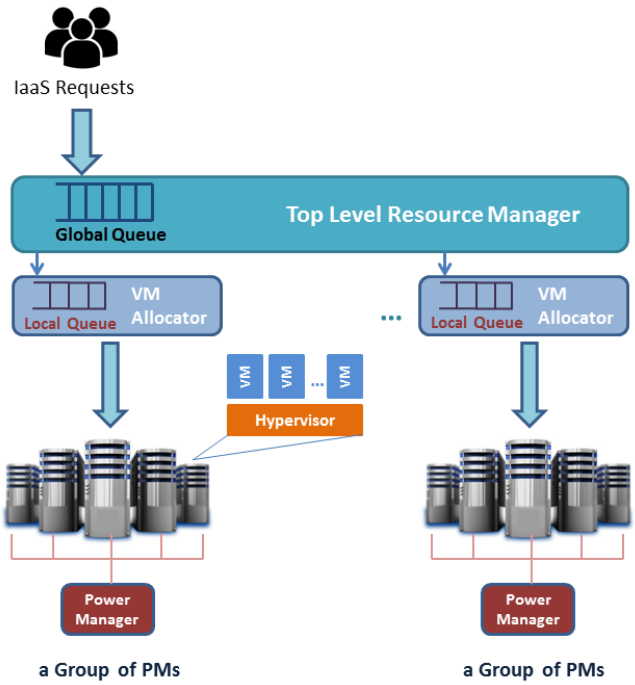

Fig. 1. An abstract architecture for an IaaS cloud

IaaS requests are submitted to the cloud provider in the form of Virtual Machine (VM) requests. VMs are on-demand operating system instances to provide users with computational resources. In the following, the term *request* is used to refer to a user request for provisioning VMs on top of Physical Machines (PMs). As shown in Fig. 1, the *top-level resource manager* acts as the entry point of user requests. Requests are submitted to the *global queue* of the top-level management layer which processes requests according to a specific scheduling algorithm typically in a First Come First Served (FCFS) order. According to a certain policy, the top-level resource manager chooses a group of PMs on top of them the new VM could be instantiated. In this context, a group of PMs can be considered as a cluster of servers or a whole data center itself, which is managed by an independent management layer, named *bottom-level resource manager* or *VM allocator*. In the following, we use the term *cluster* to refer to such group of physical servers.

After finding a suitable cluster to be allocated to a request, the top-level resource manager sends the request to the VM allocator of the selected cluster. Different resource allocation policies can be implemented in the top-level resource manager of a cloud provider such as random, best-fit, and worst-fit dispatch [17]. The VM allocator decides to which PM the request should be sent, and then, it sends the

request to the hypervisor of the destination PM. For the sake of simplicity, we assume that each request requires only a single VM instantiation to be processed, and all VM requests are homogeneous demanding a fixed amount of computational power, memory, and storage [8], [32]. Furthermore, it is assumed that each PM is allocated to only a single VM [6], [29] though this constraint can be relaxed as described in Section 4.1. PMs are assumed to be categorized into two separate pools: *hot* (running) and *cold* (turned off), resulting in different energy costs and provisioning delays [29]–[33]. A VM can be provisioned on hot PMs with minimum delay whereas PMs in the cold pool need additional time to be powered on before a VM instance can be deployed.

A VM allocator provisions a request based on a specific scheduling algorithm (typically the request at the head of its *local queue*) on a hot server if the pool of idle hot PMs is not empty. In this case, the hosting PM becomes busy and cannot host any additional VM until it is released. If no idle hot PM is available, a PM from cold pool must be transferred to the hot pool to be used for provisioning the requested VM. This action can be performed, e.g., using the Wake on LAN (WoL) feature implemented on Network Interface Cards (NICs) of servers [42]. If both hot and cold pools are empty, the request has to wait. Since the VM assigned to a request is released after finishing the user's tasks, the busy PM allocated to that VM is also turned back to the hot pool of idle PMs. In order to save power, idle hot PMs are moved to cold pool if no request is waiting in the queue of VM allocator. The movements between hot and cold pools are handled by the *power manager*. The details of these two types of movements are explained in Section 4.1.

Since failures in cloud data centers are very common [11], [43], a comprehensive architecture should consider the possibility of PM failures. We suppose that PM failure can happen in each state even when a PM is in a cold pool. It turns out that the failure rate of a busy hot PM is larger than that of an idle hot PM, and consequently, the failure rate of an idle hot PM is greater than that of a cold PM. Herein, it is assumed that the applications running on VMs are stateless, or their state is managed out of the VMs through a database or cloud data store. Alternatively, applications state can be reliably managed by middleware layers, e.g. IBM Websphere [44], that replicate objects on multiple server instances transparently to the end user, so that the state of applications can be restored if an instance becomes unavailable. Moreover, the concept of stateless VMs, considered in some literature [45]–[48], is adopted by Google Compute Engine (GCE). Hence, when a busy hot PM fails, the VM running on the failed PM can be restarted on another PM available for hosting VMs. The VM request is returned to the waiting queue of VM allocator to be processed later.

4 The Proposed SRN Models

In order to model and evaluate the performance, availability, and power consumption of an IaaS provider according to the architecture shown in Section 3, an SRN for modeling a single cluster is presented in Section 4.1. Afterwards, using the SRN model presented in the first step, a general model is presented in Section 4.2.1 which captures the behavior of the whole data center. To alleviate the scalability issues of such a monolithic SRN model, two approximate models are presented. The first one which is based on folding technique is presented in Section 4.2.2, and the second one which is based on fixed-point iteration method is introduced in Section 4.2.3. Moreover, the proof of existence of the fixed-point is given in Section 4.2.4. We do not present the concept of SRNs here due to space limitations. More information on Petri nets (PNs), timed and stochastic extensions of PNs, and SRNs can be found in [21], [25], [49]–[51].

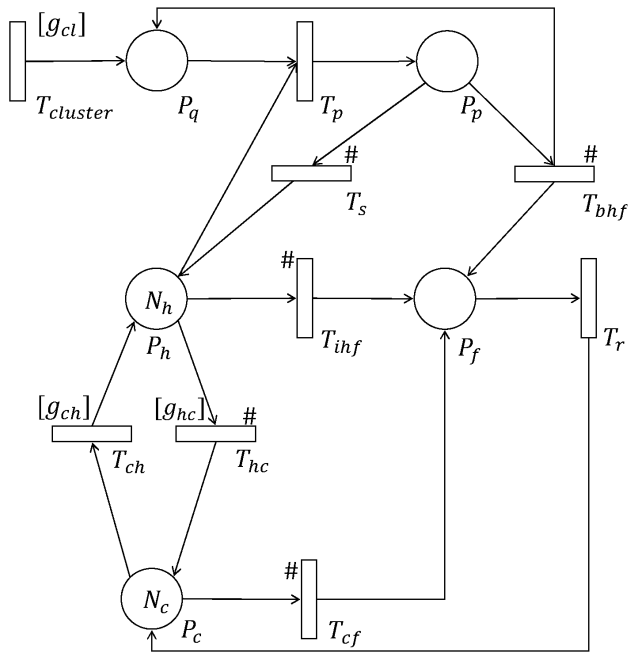

Fig. 2. The proposed SRN model for a cluster

4.1 SRN Model for a Cluster

The SRN model proposed for analyzing a cluster in IaaS cloud data center is presented in Fig. 2, and the description of all elements of this SRN is given in Table 1. It should be mentioned that, as in many other approaches, the times assigned to all timed transitions are assumed to be exponentially distributed [8], [16], [17], [19], [29]–[32], [34]. Specifically, it is assumed that the service duration of a running VM is a random variable following an exponential distribution with a constant rate, denoted by λ_s [6], [8], [17], [19], [28]–[30], [32]–[34]. Hence, we consider neither the case where the user specifies service duration of his/her VM, nor the case in which the provider or user can terminate the VM at any moment.

Input parameters of this model are: (1) arrival rate of requests (λ_{cl}), (2) the capacity of input queue (Q_c), (3) initial number of PMs in cold and hot pools (N_c and N_h, respectively), (4) mean time of VM provisioning $(1/\lambda_p)$, (5) service rate of each VM (λ_s), (6) mean time to move a cold PM to the hot pool $(1/\lambda_{ch})$, (7) mean time to move a hot PM to the cold pool $(1/\lambda_{hc})$, (8) failure rates of cold, idle, and busy PMs represented by λ_{cf}, λ_{ihf}, and λ_{bhf}, respectively, where $\lambda_{cf} < \lambda_{ihf} < \lambda_{bhf}$, and (9) repair rate of a failed PM (λ_r). As shown in Fig. 2, request arrival is modeled by timed transition $T_{cluster}$. Once transition $T_{cluster}$ fires, a token is

TABLE 1
Elements of the SRN model shown in Fig. 2

Name	Description	Rate or Initial number of tokens
P_q	Queue of input requests	0
P_p	Provisioned PMs	0
P_c	Cold pool of PMs	N_c
P_h	Hot pool of PMs	N_h
P_f	Failed PMs	0
$T_{cluster}$	Arrival of requests	λ_{cl}
T_p	Provisioning VM for request	$Min(\#P_q, \#P_h).\lambda_p$
T_s	Servicing requests	$\#P_p.\lambda_s$
T_{ch}	Powering on a cold PM	λ_{ch}
T_{hc}	Powering off a hot PM	$\#P_h.\lambda_{hc}$
T_{cf}	Failure of cold PMs	$\#P_c.\lambda_{cf}$
T_{ihf}	Failure of idle hot PMs	$\#P_h.\lambda_{ihf}$
T_{bhf}	Failure of busy hot PMs	$\#P_p.\lambda_{bhf}$
T_r	Repair of failed PMs	λ_r

TABLE 2
Guard functions of the SRN model shown in Fig. 2

Guard Function	Value
g_{cl}	1 if $[\#P_q] < Q_c$ 0 otherwise
g_{ch}	1 if $[\#P_q - \#P_h] > 0$ 0 otherwise
g_{hc}	1 if $[\#P_q] = 0$ 0 otherwise

put into place P_q to show that a request has been submitted to the cluster and it should be processed later. The guard function g_{cl} is associated with transition $T_{cluster}$ to take care of the size of the input queue. When the number of tokens in place P_q reaches the given threshold Q_c, the guard function g_{cl}, shown in Table 2, prevents transition $T_{cluster}$ from firing. If we consider a queue with zero capacity, then an arriving request should only check the availability of a PM to be assigned to that request. If there is such a PM in the cluster, the request is accepted; otherwise, it is rejected from the system.

Places P_c and P_h model *cold* and *hot* pools of the system, respectively. The number of tokens inside places P_c and P_h show the number of operational PMs in cold and hot pools, respectively. If there is at least one token in both places P_q and P_h, transition T_p is enabled, and it can fire. Upon firing of this transition, one token is removed from both places P_q and P_h, and one token is added to place P_p. The actual firing rate of T_p is $Min(\#P_q, \#P_h) \cdot \lambda_p$ where λ_p is the provisioning rate of a single VM and $Min(\#P_q, \#P_h)$ is the minimum number of tokens in P_q and P_h. If there is a token in place P_q and no token in place P_h, the existence of a token in place P_c is checked. If there is at least one token in P_c, it shows that there is a cold PM which can be moved to the hot pool to be allocated to the waiting request. In this case, transition T_{ch} fires and moves a token from P_c to P_h. The guard function of transition T_{ch}, called g_{ch}, is presented in Table 2. Otherwise, the newly received request has to wait because there are no idle hot or cold PMs, which indicates that all PMs have been already allocated to the requests or failed. In the case of a queue with zero capacity, the guard function $[g_{cl}]$ should check the condition $[\#P_h] + [\#P_c] > 0$, which checks the existence of an available PM in the cluster.

The existence of a token in place P_p shows that a VM is running on a PM. The servicing process is modeled by timed transition T_s. The firing rate of this transition is $(\#P_p) \cdot \lambda_s$ where $(\#P_p)$ denotes the number of tokens inside place P_p. So, the firing rate of transition T_s is marking dependent, which is shown by symbol # on the arc connecting place P_p to transition T_s. Upon firing of this transition, a token is removed from place P_p, and a token is deposited into place P_h to show that a user request has been completed, and its corresponding PM was turned back to the hot pool. Timed transition T_{hc} models the movement of a hot PM to the cold pool. A guard function, called g_{hc}, is associated with this transition to check the existence of a token in place P_q. If there is no token in place P_q, and at least one token in place P_h, transition T_{hc} fires. Using this mechanism, we shut down idle hot PMs to save power when there is no waiting request in the cluster. The guard function g_{hc} is shown in Table 2. As shown in Table 1, the firing rate of transition T_{hc} is $(\#P_h) \cdot \lambda_{hc}$ where $(\#P_h)$ is the number of tokens in place P_h and $(1/\lambda_{hc})$ is the mean time to move a hot PM to the cold pool.

As mentioned in Section 3, all PMs in a cloud system are prone to fail. We model these failures using three different timed transitions, named T_{bhf}, T_{ihf}, and T_{cf}, which model failure processes of *busy hot PMs*, *idle hot PMs*, and *cold PMs*, respectively. Upon firing of transition T_{bhf}, a token is removed from place P_p, and a token is deposited into both places P_q and P_f. Using this mechanism, the request associated to a failed PM is returned to the waiting queue to be allocated to another PM later. If the capacity of the input queue is zero, the output arc from transition T_{bhf} to place P_q should be removed. In this case, the rejection probability of the cluster can be computed by counting the number of tokens rejected by both timed transitions $T_{cluster}$ and T_{bhf}. Tokens in place P_f represent failed PMs waiting to be repaired. Timed transition T_r models the repair process of failed PMs. When a failed PM is repaired through transition T_r, one token is put into place P_c, which indicates that the repaired PM is available for submitted requests as a cold PM. Transition T_r is not marking dependent since each cluster is assumed to be equipped with only one repair facility. Firing rates of the transitions related to the failure/repair behavior of PMs are shown in Table 1.

It is worth mentioning that the proposed model shown in Fig. 2 can be easily modified to handle VM multiplexing by setting the multiplicity of the arcs connecting transition T_{ch} to place P_h, place P_h to transition T_{hc}, transition T_{cf} to place P_f, and place P_f to transition T_r to the number L, where L is the maximum number of VMs which can be concurrently provisioned on a single PM. However, to avoid the state space explosion which occurs in the monolithic model proposed for a cloud data center in Section 4.2, we

ignore this extension.

4.2 SRN Models for Entire Cloud Data Center

In this section, the SRN model presented for a single cluster is used to model a more realistic IaaS cloud including many clusters which are managed hierarchically. As described in Section 3, the hierarchical structure considered here includes two management levels though it can even be extended to more levels. As a two-level hierarchical structure, the top-level entity can be considered as a network of data centers while the low-level could be a data center itself as proposed in [52]. Alternatively, the top-level element can be considered as a data center while the bottom-level could model a cluster [38], [53]. Each level has its own management part in such a way that the top-level management layer receives an IaaS request and decides to which low-level element this request should be dispatched. Afterwards, the management part of the bottom-level element decides how to service the request using its own available PMs in the local cold and hot pools.

4.2.1 The Monolithic Model

Using the SRN model shown in Fig. 2, a model for the entire cloud data center is presented in Fig. 3 that obeys the reference architecture of Fig. 1. The proposed SRN contains N clusters at bottom-level which are homogeneous with respect to their structure, and the number and capacity of their PMs.

In this model, there is a single entry point for user requests at top-level management layer, which is modeled by timed transition T_{input} with firing rate λ_{input}. The arriving requests are buffered into a single queue with limited capacity modeled by place P_{input}. The transition T_{input} is enabled only if the number of tokens in place P_{input} is less than the given threshold Q_{in}. This condition is realized by guard function g_{in} shown in Table 3. The explanation given about the local queue with zero capacity in bottom-level resource manager can be generalized to the global queue of the top-level resource manager by checking the existence of an available PM in one of the underlying clusters upon arriving a request to the global queue. The arriving requests to the top-level resource manager are dispatched to the clusters existing in the data center. The dispatching process is modeled by timed transitions $T_{cluster_i}, 1 \leq i \leq N$. The selection of the cluster to host the incoming request is done according to a load distribution policy which checks the number of waiting requests inside each cluster. If a cluster has free resources, it is selected to host the new request. In this paper, we adopt a random load distribution policy, i.e., if there are more than one cluster capable to host a request, one of them is selected randomly. If the local queues of all clusters are full, requests have to wait in the global queue till one of the clusters becomes available. After dispatching an incoming request, the VM allocator of the receiving cluster queues the request in its local buffer shown by P_{q_i} in Fig. 3. The capacity of this queue for each cluster i is Q_c. The remaining process proceeds according to the description given in Section 4.1. All guard functions of SRN model presented in Fig. 3 are described in Table 3.

The SRN model shown in Fig. 3 can be used to model a cloud data center with any number of clusters with different

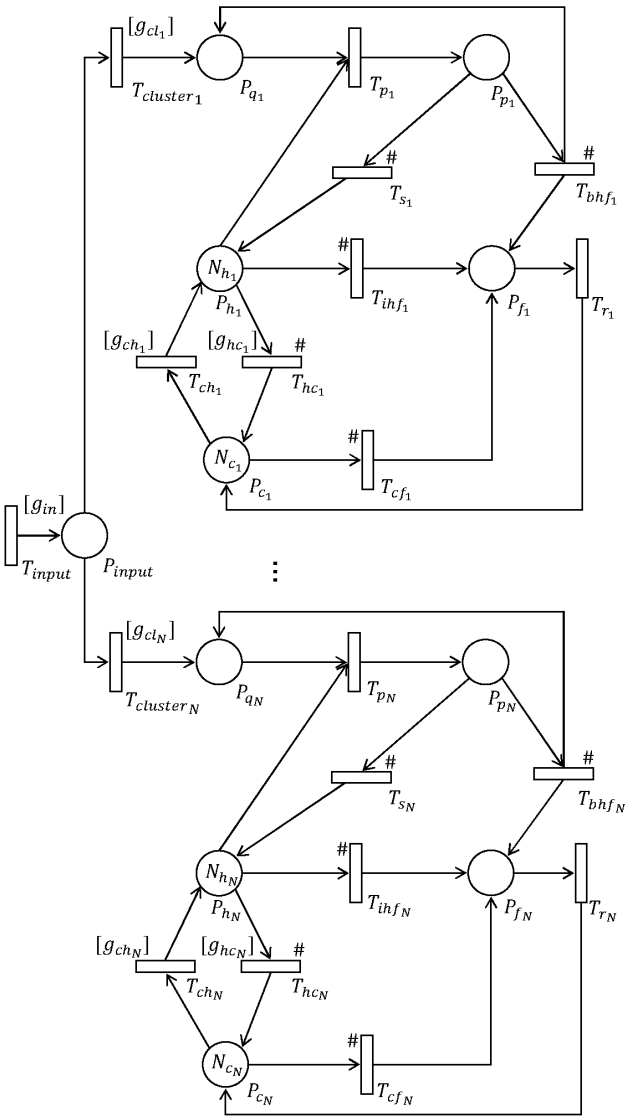

Fig. 3. Monolithic SRN model of a cloud data center

TABLE 3
Guard functions of the SRN model shown in Fig. 3

Guard Function	Value
g_{in}	1 if $[\#P_{input}] < Q_{in}$ 0 otherwise
g_{cl_i}	1 if $[\#P_{q_i}] < Q_c$ 0 otherwise
g_{ch_i}	1 if $[\#P_{q_i} - \#P_{h_i}] > 0$ 0 otherwise
g_{hc_i}	1 if $[\#P_{q_i}] = 0$ 0 otherwise

configurations. However, when it is applied to model a real cloud system even with a small number of clusters and servers, it encounters the state space explosion problem since the number of states of the underlying Markov chain of the SRN grows exponentially and cannot be solved by existing packages and tools. The number of states increases exponentially as the number of clusters, the number of servers in each cluster, and the capacity of global and local queues increase. Numerical results showing the intractability of the monolithic model are given in Section 6.

4.2.2 The Folded Approximate Model

Since it is assumed that the structure and configuration of all clusters of the SRN model presented in Fig. 3 are the same, one possible approximate model consists of an arbitrarily selected *Tagged* cluster (e.g. cluster 1) and a *Folded* subnetwork for the remaining $(N-1)$ ones grouped together to form a single powerful cluster while leaving the top-level management layer intact [16], [22]. In such model, places P_q, P_p, P_h, P_c, and P_f in the *Folded* part correspond to places P_{q_i}, P_{p_i}, P_{h_i}, P_{c_i}, and P_{f_i}, $2 \leq i \leq N$, of the monolithic SRN model shown in Fig. 3, respectively.

In this approximate model, the initial number of tokens inside places P_c and P_h are $N_c = (N-1) \cdot N_{c_1}$ and $N_h = (N-1) \cdot N_{h_1}$, respectively, where N_{c_1} and N_{h_1} are the initial number of tokens in places P_{c_1} and P_{h_1} of the *Tagged* cluster, respectively. Furthermore, the capacity of the local queue of the *Folded* cluster is $Q_c = (N-1) \cdot Q_{c_1}$, where Q_{c_1} is the capacity of the local queue of the *Tagged* cluster. The request arrival rate (λ_{cl}), transfer rate from the cold pool to the hot pool (λ_{ch}), and repair rate of PMs (λ_r) in the *Folded* subnet are $\lambda_{cl} = (N-1) \cdot \lambda_{cl_1}$, $\lambda_{ch} = (N-1) \cdot \lambda_{ch_1}$, and $\lambda_r = (N-1) \cdot \lambda_{r_1}$, respectively, where λ_{cl_1}, λ_{ch_1}, and λ_{r_1} are the corresponding parameters in the *Tagged* cluster.

Since the rate of PM transmission from the hot pool to the cold pool, provision and service rates of VMs, and failure rates of cold and hot PMs are marking dependent, these values are not multiplied by $(N-1)$. This is because the initial number of cold and hot PMs and the input rate of requests to the *Folded* subnet have been already multiplied by $(N-1)$. The guard functions associated with timed transitions of the *Folded* subnet of the model are the same as the guard functions described in Table 2 except for g_{cl} in which Q_c has to be replaced by $(N-1) \cdot Q_c$. It is worth to mention that multiplying the parameters of a single cluster by $(N-1)$ and associating them to the corresponding parameters of the *Folded* subnet allow only to approximate the required metrics. However, exact estimation of the number of tokens and the rate of transitions is very difficult since there are many factors that come to play.

As it is shown in Section 6, although the folding approximation method partially overcomes the scalability problem of the monolithic model, it is not still appropriate for modeling cloud systems with large number of clusters or servers. Nevertheless, it provides a good improvement over the monolithic model regarding the number of states and non-zero entries in the underlying Markov chain while keeping errors in a tolerable range.

4.2.3 The Fixed-point Approximate Model

In order to cope with the state space explosion problem that still affects the folded model, and to refine the model further to solve larger cloud instances, another approximate model based on fixed-point iteration [22], [24], [54] is presented. The fixed-point iteration technique is known as a good solution for analyzing a system of interdependent components [16], [31]. In this method, each component is analyzed with the remaining components represented in a simplified manner (i.e., a delay). The method acts iteratively, and the parameters of the simplified complement of a component are modified after the other components are analyzed. Each component is then re-analyzed with new input parameters to produce updated inputs for other components. This process is iterated until the difference of interested values in two successive iterations becomes lower than a given threshold.

To apply fixed-point technique, the monolithic SRN model shown in Fig. 3 is divided into two sub-models: the first sub-model contains a tagged cluster together with the top-level cloud management elements, and the second sub-model contains remaining $(N-1)$ clusters. Since the remaining $(N-1)$ clusters act as a delay to the tagged cluster, they are replaced by a timed transition in the first sub-model. Fig. 4 shows the first sub-model of the fixed-point iteration method. In this figure, transition T_{delay} represents the delay associated with other $(N-1)$ clusters.

The number of tokens, the rate of transitions, and guard functions defined in the SRN model shown in Fig. 4 are the same as their corresponding values in the *Tagged* subnet of the model described in Section 4.2.2. To evaluate the performance measures of interest, this model should be solved with an appropriate firing rate for timed transition T_{delay}. Since the rate of this transition cannot be determined a priori, the SRN model shown in Fig. 4 cannot be solved directly. Instead, the rate of T_{delay} can be approximated using the iterative approach of fixed-point by exploiting the second sub-model shown in Fig. 5.

In Fig. 5, place P_{input} contains M tokens where $1 \leq M \leq Q_{in}$, and Q_{in} is the capacity of the global queue of top-level resource manager, which is equal to the maximum number of tokens that can be placed in P_{input} represented in SRN model of Fig. 4. The initial number of tokens in places P_c and P_h are $N_c = (N-1) \cdot N_{c_1}$ and $N_h = (N-1) \cdot N_{h_1}$, respectively, where N_{c_1} and N_{h_1} are the corresponding parameters of the tagged cluster defined in the SRN model of Fig. 4. The capacity of the local queue of the second sub-model is $Q_c = (N-1) \cdot Q_{c_1}$, where Q_{c_1} is the capacity of the local queue of the first sub-model. Furthermore, the rate of request arrival to the system (λ_{cl}), the rate of moving PMs from the cold pool to the hot pool (λ_{ch}), and the repair rate of failed PMs (λ_r) in the model of Fig. 5 are $\lambda_{cl} = (N-1) \cdot \lambda_{cl_1}$, $\lambda_{ch} = (N-1) \cdot \lambda_{ch_1}$, and $\lambda_r = (N-1) \cdot \lambda_{r_1}$, respectively, where λ_{cl_1}, λ_{ch_1}, and λ_{r_1} are the corresponding parameters of the first sub-model. Other values for firing rate of transitions and the initial number of tokens in places are the same as those of the model shown in Fig. 4.

As shown in Fig. 5, a new place P_{end} is added to the model to trap all requests submitted to the system after their successful execution. Therefore, if the initial number

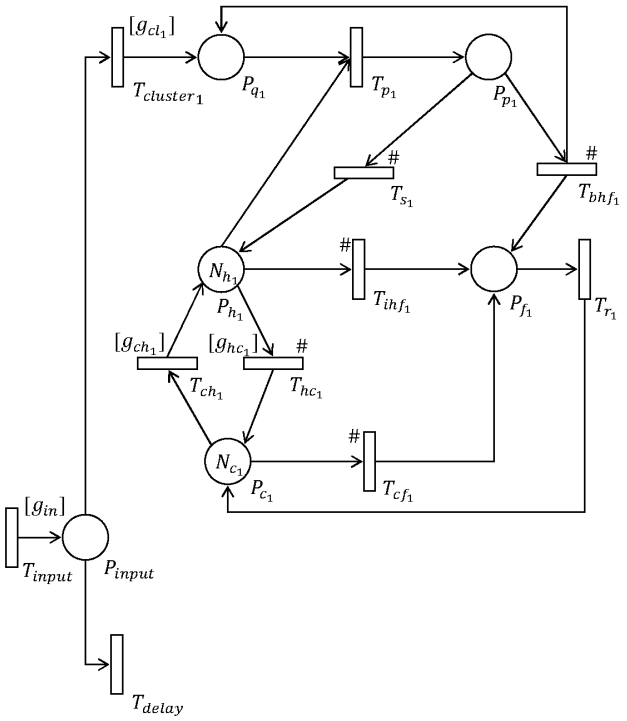

Fig. 4. The fixed-point approximate model for entire cloud data center (the first sub-model)

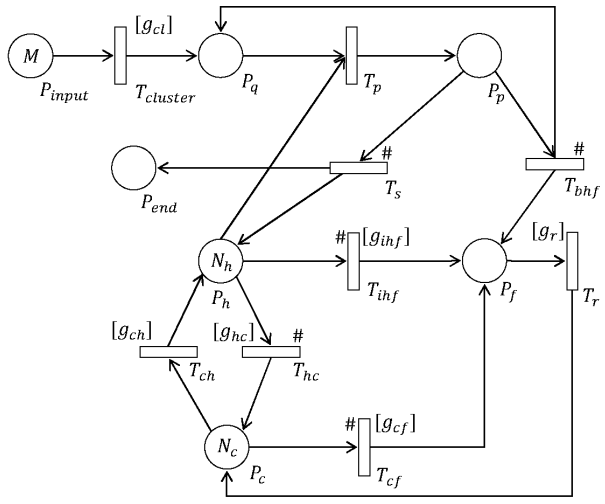

Fig. 5. The fixed-point approximate model for entire cloud data center (the second sub-model)

TABLE 4
Guard functions of the SRN model shown in Fig. 5

Guard Function	Value
g_{ihf}, g_{cf}, and g_r	1 if $[\#P_{end}] < M$ 0 otherwise

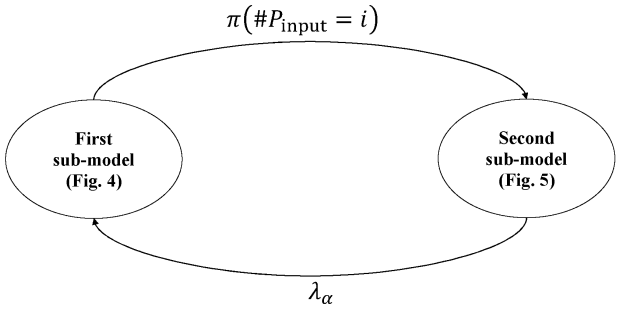

Fig. 6. Import graph of interacting sub-models in the fixed-point approximate technique

of tokens inside place P_{input} is M, after a specific amount of time, all the tokens will be moved to the place P_{end}. If suitable guard functions are defined for transitions of this model, the underlying Markov chain will be an absorbing Markov chain and the Mean Time To Absorption (MTTA) of the model can be calculated. Table 4 describes the newly added guard functions to the second sub-model. Guard functions g_{ihf}, g_{cf}, and g_r are associated with timed transitions T_{ihf}, T_{cf}, and T_r, respectively. Other guard functions of the SRN model shown in Fig. 5 are the same as those of the *Folded* subnet of the SRN model proposed in Section 4.2.2.

Let $MTTA_i$ represent the mean time to absorption of the SRN model shown in Fig. 5 when M is equal to i. Also, let $\pi(\#P_{input} = i)$ denote the steady-state probability of there being i tokens in place P_{input} of the same model, which can be computed by steady-state analysis of the SRN model shown in Fig. 4. Then, the MTTA of the SRN model of Fig. 5 is computed as (1).

$$MTTA = \sum_{i=1}^{Q_{in}} MTTA_i \cdot \pi(\#P_{input} = i) \tag{1}$$

Once MTTA value is computed, the firing rate of timed transition T_{delay} is computed as (2).

$$\lambda_\alpha = \frac{1}{MTTA} \tag{2}$$

The value of λ_α is used for solving the first sub-model at the next iteration. Solving this model at the next iteration results in new steady-state probabilities of there being i tokens in place P_{input}. These values are then applied to equations (1) and (2) to determine the new value of λ_α. This process continues until the difference of values of λ_α in two successive iterations becomes lower than a specific threshold. The import graph describing interaction between two sub-models of the fixed-point approximate model (SRN models shown in Fig. 4 and Fig. 5) is represented in Fig. 6.

4.2.4 Existence of a Fixed-point

In this section, we demonstrate that a fixed-point always exists when the decomposition illustrated above is performed. To achieve this, the following proof is presented.

Let $y_i = \pi(\#P_{input} = i)$, where $\pi(\#P_{input} = i)$ is the steady-state probability of there being i tokens in place

P_{input} and $1 \leq i \leq Q_{in}$. Considering the SRN model presented in Fig. 4, for some function h, we can write $y_i = h(\lambda_\alpha)$ where λ_α is the firing rate of timed transition T_{delay}. On the other hand, this rate is obtained by computing MTTA in SRN shown in Fig. 5 for i tokens inside place P_{input} where $1 \leq i \leq Q_{in}$. Using (1) and (2), we can write $\frac{1}{\lambda_\alpha} = \sum_{i=1}^{Q_{in}} MTTA_i \cdot \pi(\#P_{input} = i)$ where $MTTA_i$ is the mean time to absorption of SRN shown in Fig. 5 when the number of the tokens inside place P_{input} is i. Let vector $\vec{y} = (y_1, \ldots, y_{Q_{in}})$, so for some function f we can derive $\vec{y} = f(\vec{y})$.

Brouwer's fixed-point theorem [24] is used to show that $f(\vec{y}) = \vec{y}$ has a solution. This theorem states that if there exists a *compact convex* set $\overline{C} \subset R^n$ and there is a continuous function f such that $f(\vec{y}) \in \overline{C}$ for all $\vec{y} \in \overline{C}$, then there exists a solution for equation $f(\vec{y}) = \vec{y}$.

Since $\vec{y} = (y_1, y_2, \ldots, y_{Q_{in}})$, and each $y_i = \pi(\#P_{input} = i)$ representing the steady-state probability of there being i tokens in place P_{input} is bounded below by 0 and above by 1, so each y_i is bounded in range $[0, 1]$. Hence, $\overline{C}$ is defined as a set of points $(y_1, y_2, \ldots, y_{Q_{in}})$ where each $y_i \in [0, 1]$. Consider the function f over $\overline{C}$ by defining $y_i = \pi(\#P_{input} = i)$ and $y_i = f(y_i)$, $1 \leq i \leq Q_{in}$. Since the probability of being in a subset of a Markov chain is always bounded below by 0 and above by 1, therefore $y_i \in [0, 1]$ for all i. Now $\overline{C}$ will be shown to be a convex set. A set $\overline{C} \subset R^n$ is convex if $\lambda\vec{x} + (1 - \lambda)\vec{y} \in \overline{C}$ whenever $\vec{x}$ and $\vec{y}$ are n-vectors $\in \overline{C}$ and $\lambda \in [0, 1]$. Let us consider one equation: $z_i = \lambda x_i + (1 - \lambda)y_i$. It turns out that $z_i \geq 0$ since $\lambda \geq 0$, $x_i \geq 0$, $(1 - \lambda) \geq 0$, and $y_i \geq 0$. Since z_i is maximized when $x_i = y_i = 1$ (since $\lambda \geq 0$ and $(1 - \lambda) \geq 0$) to its maximum value 1, $z_i \leq 1$.

Finally, we show that f is continuous over $\overline{C}$. Function $f(\vec{y})$ is continuous if for each point $\hat{y} \in \overline{C}$, $\lim_{\vec{y} \to \hat{y}} f(\vec{y}) = f(\hat{y})$. As $f(\vec{y})$ is a vector valued function, this is equivalent to say $\lim_{\vec{y} \to \hat{y}} f_k(\vec{y}) = f_k(\hat{y})$ for $k \in \{1, 2, \ldots, Q_{in}\}$ and $\hat{y} \in \overline{C}$. By defining $\lim_{\vec{y} \to \hat{y}} f_k(\vec{y}) = \lim_{\vec{y} \to \hat{y}} \big[[MTTA_1 \cdot \pi(\#P_{in} = 1)] + [MTTA_2 \cdot \pi(\#P_{input} = 2)] + \ldots + [MTTA_{Q_{in}} \cdot \pi(\#P_{input} = Q_{in})]\big]$, we can derive $\lim_{\vec{y} \to \hat{y}} f_k(\vec{y}) = \lim_{\vec{y} \to \hat{y}} \big[[MTTA_1 \cdot y_1] + \big[[MTTA_2 \cdot y_2] + \ldots + [MTTA_{Q_{in}} \cdot y_{Q_{in}}]\big]$. Since $y_i = \pi(\#P_{input} = i)$ is the probability of being in a subset of the states of a Markov chain, $\lim_{y_i \to \hat{y_i}} y_i = \hat{y_i}$. Due to the fact that each term of the summation converges to its (finite) value at $\hat{y}$, $\lim_{\vec{y} \to \hat{y}} f_k(\vec{y}) = f_k(\hat{y})$, and therefore, $\lim_{\vec{y} \to \hat{y}} f(\vec{y}) = f(\hat{y})$.

5 Performance Measures of Interest

In this section, some performance measures which can be obtained by solving the SRN models given in Section 4 at steady-state are introduced. These measures can be computed using the Markov reward approach [55]. In this approach, appropriate reward rates are assigned to each feasible marking of an SRN model, and then, expected reward rates in the steady-state are computed. Let r denote the reward function that associates a reward rate r_i to each marking i of an SRN model. If π_i denotes the probability for the SRN to be in marking i at steady-state, then the expected reward at steady-state, $E[r]$, can be calculated by $\sum_i r_i \pi_i$. In the following, some of the interesting measures which can be computed using our proposed models are described. These measures are defined and evaluated for a single cluster (i.e., the tagged cluster) in the cloud data center.

Mean response time: It is the mean time from when a request is submitted by the top-level resource manager to a VM allocator at cluster level until the time it completes its service. In other words, this is the mean time from the instant a token is deposited into P_q until it leaves place P_p by firing transition T_s. Using Little's law [20], the steady-state mean response time is computed as (3).

$$E[RT] = \frac{E[r^q] + E[r^p]}{\lambda_{cl}^{eff}} \tag{3}$$

where r^q and r^p are reward functions counting the number of tokens inside places P_q and P_p, respectively, and λ_{cl}^{eff} is the effective rate of request arrival into the cluster queue. In other words, it is the expected throughput of transition $T_{cluster}$ and is computed as (4).

$$\lambda_{cl}^{eff} = (1 - p_b)\lambda_{cl} \tag{4}$$

where p_b is the blocking probability of arriving requests at cluster level and is computed by assigning the reward function (5) to SRN models.

$$r^b = \begin{cases} 1, & [\#P_q] \geq Q_c \\ 0, & otherwise \end{cases} \tag{5}$$

Mean percentage of available PMs: It is the mean percentage of non-failed PMs, which can be derived as (6).

$$E[AV] = (1 - \frac{E[r^f]}{N_c + N_h}) \times 100 \tag{6}$$

where r^f is the reward function that counts the number of failed PMs by counting the number of tokens inside place P_f. Moreover, N_c and N_h are initial numbers of PMs in cold and hot pools, respectively as described in Section 4.1.

Mean power consumption: It is the mean value of power consumed by PMs inside a cluster and is computed as (7).

$$E[PC] = \big(c_{bh}E[r^p] + c_{ih}E[r^h] + c_cE[r^c]\big) \cdot p_{max} \tag{7}$$

where r^p , r^h , and r^c are the reward functions that count the number of tokens inside places P_p, P_h, and P_c, respectively. Moreover, p_{max} is the maximum power consumption of a PM, and c_{bh}, c_{ih}, and c_c are fractions of maximum power consumed by a PM in its busy, idle, and turned off modes, respectively. $(0 \leq c_{bh}, c_{ih}, c_c \leq 1)$.

6 Performance Evaluation

In order to compare the results obtained by analytically solving three SRN models presented in Section 4.2, a two-level cloud data center with different number of clusters is considered in this section. The models have been solved for a wide range of input parameter values, but for the sake of brevity, we only report some interesting results in this paper. Most of the values considered as input parameters of the cloud system investigated in this paper, are according to other literature proposals [8], [16], [17], [29]–[31], [56]. These values are shown in Table 5. As mentioned in Section 4.1, the value of L, the multiplexing factor, can be set to any arbitrary number, such as 2, 4, or 8 which are typical in

TABLE 5
Configuration of the cloud data center considered in this paper

Parameter	Value or Range
Requests arrival rate (λ_{input})	$[30, 50]$ *requests/hour*
Capacity of global queue (Q_{in})	5 *requests*
Capacity of local queue (Q_{c_i})	2 *requests*
Initial number of cold PMs in each cluster (N_{c_i})	3 *PMs*
Initial number of hot PMs in each cluster (N_{h_i})	0 *PMs*
Requests arrival rate to each cluster (λ_{cl_i})	1000 *requests/hour*
Provisioning rate (λ_{p_i})	30 *VMs/hour*
Service rate of each VM (λ_{s_i})	3 *VMs/hour*
Transmission rate of a cold PM to the hot pool (λ_{ch_i})	20 *PMs/hour*
Transmission rate of a hot PM to the cold pool (λ_{hc_i})	60 *PMs/hour*
Failure rate of a cold PM (λ_{cf_i})	0.001 *PMs/hour*
Failure rate of an idle PM (λ_{ihf_i})	0.01 *PMs/hour*
Failure rate of a busy PM (λ_{bhf_i})	0.015 *PMs/hour*
Repair rate of a failed PM (λ_{r_i})	0.33 *PMs/hour*
Maximum power consumption of a PM (p_{max})	400 *watts*
Fraction of maximum power consumed in cold mode (c_c)	0.1
Fraction of maximum power consumed in idle mode (c_{ih})	0.675
Fraction of maximum power consumed in busy mode (c_{bh})	1

TABLE 6
Steady-state mean response time ($E[RT]$) resulted from monolithic, folded, and fixed-point models (in *seconds*)

N	$\lambda_{\mathbf{input}}$	SRN Models				
		Monolithic	Folded		Fixed-point	
		Value	Value	PE (%)	Value	PE (%)
2	**30**	2118.834	2118.834	0.000	2123.208	0.206
	35	2121.728	2121.728	0.000	2123.298	0.074
	40	2122.693	2122.693	0.000	2123.327	0.030
	45	2123.057	2123.057	0.000	2123.341	0.013
	50	2123.208	2123.208	0.000	2123.345	0.006
3	**30**	2075.602	2065.414	0.491	2123.194	2.293
	35	2104.855	2101.090	0.179	2123.294	0.876
	40	2115.634	2114.147	0.070	2123.327	0.364
	45	2119.853	2119.216	0.030	2123.338	0.164
	50	2121.642	2121.343	0.014	2123.345	0.080

TABLE 7
Steady-state percentage of available PMs ($E[AV]$) resulted from monolithic, folded, and fixed-point models

N	$\lambda_{\mathbf{input}}$	SRN Models				
		Monolithic	Folded		Fixed-point	
		Value	Value	PE (%)	Value	PE (%)
2	**30**	95.432	95.432	0.000	95.429	0.003
	35	95.430	95.430	0.000	95.429	0.001
	40	95.429	95.429	0.000	95.429	0.000
	45	95.429	95.429	0.000	95.429	0.000
	50	95.429	95.429	0.000	95.429	0.000
3	**30**	95.488	95.514	0.028	95.429	0.061
	35	95.445	95.452	0.007	95.429	0.017
	40	95.434	95.436	0.002	95.429	0.005
	45	95.431	95.431	0.001	95.429	0.002
	50	95.430	95.430	0.000	95.429	0.001

real world systems [57]–[59], without encountering any state space explosion in the model of Fig. 2. However, if it is set to a larger number in the model of Fig. 3, representing the monolithic model of a cloud data center, the model cannot be solved and it encounters largeness problem. Since the main aim of both folded and fixed-point approximate models is to approximate the monolithic model, we cannot compare the results obtained from the approximate models with the results of the monolithic model if VM multiplexing is taken into account.

We use the SPNP software package [60] to solve the SRN models presented in this paper. The comparison of the steady-state mean response time of IaaS requests among monolithic, folded approximate, and fixed-point approximate models are shown in Table 6. Here, Percent Error (PE) expresses the difference between an approximate or simulated value and an exact or known one, computed from the monolithic model, as a percentage of the exact value. The steady-state percentage of available PMs resulted from the monolithic and two approximate models are presented in Table 7. Moreover, Table 8 shows the steady-state mean power consumption of PMs obtained from above-mentioned models. In all of these tables, N represents the number of clusters within a data center. It is worth mentioning that running the monolithic model for just *four* clusters produces so many states and leads to state space explosion. Therefore, we only compare the measures of the models with *two* and *three* clusters.

As can be seen in Tables 6, 7, and 8, the steady-state mean values of the interested measures obtained from all models are very close to each other. For example, the maximum percent errors of the results obtained by (folded, fixed-point) approximate models in Tables 6, 7, and 8 are $(0.491\%, 2.293\%)$, $(0.028\%, 0.061\%)$, and $(0.552\%, 1.225\%)$, respectively. This indicates that our proposed approximate models can appropriately estimate the steady-state mean response time, available PMs, and power consumption of the monolithic model.

In order to cross validate the results obtained from the monolithic model and to compare them with the results achieved by solving approximate models, we simulate the IaaS cloud mentioned above using an ad-hoc discrete-event simulation method we developed in Java. Furthermore, to demonstrate the accuracy of the proposed models, we simulate the sample cloud systems with the CloudSim framework. CloudSim is a Java-based development platform to support modeling and simulation of large-scale cloud computing environments [27]. We extend the CloudSim framework to adapt it to the requirements of the proposed models

TABLE 8
Steady-state power consumption $(E[PC])$ resulted from monolithic, folded, and fixed-point models (in *watts*)

N	λ_{input}	SRN Models				
		Monolithic	Folded		Fixed-point	
		Value	Value	PE (%)	Value	PE (%)
2	30	1109.541	1109.541	0.000	1110.285	0.067
	35	1110.095	1110.095	0.000	1110.295	0.018
	40	1110.235	1110.235	0.000	1110.298	0.006
	45	1110.276	1110.276	0.000	1110.299	0.002
	50	1110.290	1110.290	0.000	1110.299	0.001
3	30	1096.847	1090.789	0.552	1110.284	1.225
	35	1106.547	1104.928	0.146	1110.295	0.339
	40	1109.126	1108.650	0.043	1110.298	0.106
	45	1109.886	1109.729	0.014	1110.299	0.037
	50	1110.138	1110.080	0.005	1110.299	0.015

by defining the global and local queues, and PM powering on/off, failure, and repair events. The results obtained by the discrete-event simulation and the CloudSim framework are reported in Table 9 and Table 10, respectively. Each row of these tables is the average obtained from 50 independent simulation runs for which the Standard Deviation (SD) is also computed. According to Table 9, the maximum percent errors of the results obtained by discrete-event simulation and the monolithic model for response time, availability, and power consumption are 0.052%, 0.020%, and 0.036%, respectively. These errors for the results achieved by the CloudSim framework are 1.863%, 0.180%, and 0.324%, respectively. Both sets of errors indicate that the simulation approaches are able to validate the results obtained from the proposed monolithic model. Moreover, the results obtained by simulation of the cloud systems with a larger number of clusters verify the correctness of the proposed approximate models.

The numbers of states and non-zero entries of the underlying Markov chain matrix generated by three monolithic, folded and fixed-point models are shown in Table 11 and Table 12, respectively. It should be mentioned that the numbers reported for fixed-point approximate model are only the numbers generated by the second sub-model (Fig. 5) for the maximum value of M, since the numbers of states and non-zero entries of the underlying Markov chain matrix of the first sub-model (Fig. 4) are fixed numbers for all values of N and λ_{input}. As shown in Table 11 and Table 12, numbers reported for fixed-point approximate model are much less than those for two other models. Hence, it can be concluded that using fixed-point technique and folding method to approximate complex models of cloud systems can be helpful in reducing the number of states and avoiding state space explosion.

Among state-of-the-art approaches proposed in the same area, the work presented in [17] provides an analytical framework based on SRNs that allows cloud service provider to decide about the resource allocation policies to be enforced. Though it lacks the details existing in real cloud systems including VM provisioning and failure/repair of PMs, power-aware resource allocation is one of the main concerns investigated in [17]. Hence, we compare our proposed fixed-point approximate model and the model presented in [17] (the saturation strategy), based on the number of states in the underlying Markov chains and the number of the non-zero entries in the underlying Markov chain matrices. The results are shown in Table 13 and Table 14. As it can be observed from Table 13 and Table 14, the numbers reported for our fixed-point approximate model are always significantly lower than those for the model presented in [17], which proves that our fixed-point approximate model is more scalable. However, it should be noted that the aim of our proposed model and the model presented in [17], and the details of the systems considered in both models are different, and we only compared the models in terms of the resulting state space.

7 Sensitivity Analysis

In this section, the sensitivity of output measures to the variation of input parameters is studied. To achieve this, each input parameter is varied in a valid range, and then the sensitivity of the results to the variation of that parameter is analyzed. There are many input parameters in the proposed SRNs. For the sake of brevity, we only report the sensitivity of output measures to the variation of four important input parameters by considering four different scenarios. The input parameters varied in these scenarios are: (1) the initial number of cold PMs in each cluster (N_c), (2) the failure rate of a busy hot PM (λ_{bhf}), (3) the failure rate of an idle hot PM (λ_{ihf}), and (4) the failure rate of a cold PM (λ_{cf}). In all scenarios, the number of clusters, N, and the request arrival rate, λ_{input}, are set to 3 *clusters* and 40 *requests/hour*, respectively. Unless differently stated, the values of other input parameters are the ones reported in Table 5.

In the first scenario, the impact of the variation of the initial number of cold PMs in each cluster on output measures of interest is studied. To this end, the initial number of cold PMs per each cluster, N_c, is changed from 1 to 3. Tables 15, 16, and 17 show the steady-state mean response time, percentage of available PMs, and power consumption, respectively, obtained by solving the monolithic, folded, and fixed-point models. The maximum percent errors of the results obtained by (folded, fixed-point) approximate models in Tables 15, 16, and 17 are $(0.070\%, 0.364\%)$, $(0.002\%, 0.005\%)$, and $(0.043\%, 0.106\%)$, respectively, which indicate that the values of output measures obtained from all models are very close to each other. It can be further observed that all output measures are sensitive to the variation of the number of PMs. As the initial number of cold PMs inside each cluster increases, the response time and the percentage of available PMs decrease whereas the power consumption increases.

In the second scenario, the impact of the variation of failure rate of busy hot PMs on output measures of interest is studied. For this purpose, the failure rate of a busy hot PM, λ_{bhf}, is changed from 0.015 to 0.05 *PMs/hour* with step 0.005. Tables 18, 19, and 20 show the steady-state mean response time, percentage of available PMs, and power consumption, respectively, obtained by solving the monolithic, folded, and fixed-point models. The maximum percent errors of the results obtained by (folded, fixed-point) approximate models in Tables 18, 19, and 20 are $(0.085\%, 0.364\%)$, $(0.007\%, 0.013\%)$, and $(0.043\%, 0.106\%)$,

TABLE 9
Results obtained by the discrete-event simulation

N	λ_{input}	Measures								
		Response Time (in *seconds*)			Percentage of Available PMs			Power Consumption (in *watts*)		
		Value	PE (%)	SD	Value	PE (%)	SD	Value	PE (%)	SD
2	30	2118.586	0.012	1.394	95.437	0.005	0.045	1109.538	0.000	0.141
	35	2121.458	0.013	1.173	95.438	0.008	0.058	1110.158	0.006	0.239
	40	2122.675	0.001	0.920	95.435	0.006	0.063	1110.284	0.004	0.498
	45	2122.592	0.022	1.819	95.448	0.020	0.090	1110.481	0.018	0.884
	50	2123.352	0.007	0.872	95.426	0.003	0.038	1110.242	0.004	0.210
3	30	2076.674	0.052	3.990	95.494	0.006	0.049	1096.452	0.036	0.919
	35	2105.035	0.009	0.831	95.442	0.003	0.126	1106.314	0.021	0.922
	40	2116.033	0.019	1.619	95.421	0.014	0.094	1108.899	0.020	1.016
	45	2119.802	0.002	0.351	95.436	0.005	0.039	1109.910	0.002	0.118
	50	2121.880	0.011	2.071	95.443	0.014	0.055	1110.277	0.013	1.179

TABLE 10
Results obtained by the CloudSim framework

N	λ_{input}	Measures								
		Response Time (in *seconds*)			Percentage of Available PMs			Power Consumption (in *watts*)		
		Value	PE (%)	SD	Value	PE (%)	SD	Value	PE (%)	SD
2	30	2154.661	1.691	132.761	95.533	0.106	0.541	1106.190	0.302	14.864
	35	2158.603	1.738	134.095	95.528	0.103	0.625	1106.664	0.309	12.907
	40	2158.585	1.691	140.310	95.541	0.117	0.410	1106.914	0.299	12.352
	45	2157.768	1.635	126.432	95.586	0.165	0.437	1107.445	0.255	10.913
	50	2157.170	1.600	120.178	95.601	0.180	0.650	1107.647	0.238	9.535
3	30	2114.262	1.863	139.473	95.591	0.108	0.409	1095.501	0.123	11.316
	35	2140.610	1.699	131.743	95.520	0.079	0.536	1103.277	0.296	12.382
	40	2148.574	1.557	125.269	95.560	0.132	0.515	1106.061	0.276	15.657
	45	2153.200	1.573	142.551	95.519	0.092	0.789	1106.294	0.324	14.045
	50	2155.608	1.601	123.066	95.523	0.097	0.382	1106.576	0.321	13.582

TABLE 11
The number of states in the underlying Markov chain of the monolithic, folded, and fixed-point models

N	SRN Models		
	Monolithic	Folded	Fixed-point
2	66150	66150	314
3	6945750	502740	881
4	*Memory shortage*	1905750	1469
5		5159700	2057
6		*Memory shortage*	2645
10			4997
20			10877
30			16757
40			22637
50			28517

TABLE 12
The number of the non-zero entries in the underlying Markov chain matrix of the monolithic, folded, and fixed-point models

N	SRN Models		
	Monolithic	Folded	Fixed-point
2	538125	538125	1125
3	81860625	4688670	4078
4	*Memory shortage*	18905975	7223
5		53024790	10343
6		*Memory shortage*	13463
10			25943
20			57143
30			88343
40			119543
50			150743

respectively, which show that the values of the output measures obtained from all models are very close to each other. It can be seen that by increasing the failure rate of a busy hot PM, the response time increases whereas the percentage of available PMs and power consumption decrease.

In the third scenario, the impact of variation of failure

TABLE 13
Comparison of the number of states in the underlying Markov chain of our proposed fixed-point model with that of the model presented in [17]

Number of PMs	Our proposed fixed-point model	Model presented in [17]
30	4997	453292
60	10877	2598132
90	16757	6735092
120	22637	*Memory shortage*
150	28517	
300	57917	
600	116717	

TABLE 14
Comparison of the number of non-zero entries in the underlying Markov chain matrix of our proposed fixed-point model with that of the model presented in [17]

Number of PMs	Our proposed fixed-point model	Model presented in [17]
30	25943	2864362
60	57143	16401800
90	88343	42362070
120	119543	*Memory shortage*
150	150743	
300	306743	
600	618743	

TABLE 15
Steady-state mean response time $(E[RT])$ resulted from the monolithic, folded, and fixed-point models when N_c varies (in *seconds*)

$\mathbf{N_c}$	SRN Models				
	Monolithic	Folded		Fixed-point	
	Value	Value	PE (%)	Value	PE (%)
1	4011.815	4011.815	0.000	4011.826	0.000
2	2585.772	2585.668	0.004	2586.506	0.028
3	2115.634	2114.147	0.070	2123.327	0.364

TABLE 16
Steady-state percentage of available PMs $(E[AV])$ resulted from the monolithic, folded, and fixed-point models when N_c varies

$\mathbf{N_c}$	SRN Models				
	Monolithic	Folded		Fixed-point	
	Value	Value	PE (%)	Value	PE (%)
1	95.782	95.782	0.000	95.782	0.000
2	95.612	95.612	0.000	95.612	0.000
3	95.434	95.436	0.002	95.429	0.005

TABLE 17
Steady-state power consumption $(E[PC])$ resulted from the monolithic, folded, and fixed-point models when N_c varies (in *watts*)

$\mathbf{N_c}$	SRN Models				
	Monolithic	Folded		Fixed-point	
	Value	Value	PE (%)	Value	PE (%)
1	371.513	371.513	0.000	371.513	0.000
2	741.685	741.678	0.001	741.706	0.003
3	1109.126	1108.650	0.043	1110.298	0.106

TABLE 18
Steady-state mean response time $(E[RT])$ resulted from the monolithic, folded, and fixed-point models when λ_{bhf} varies (in *seconds*)

$\lambda_{\mathbf{bhf}}$	SRN Models				
	Monolithic	Folded		Fixed-point	
	Value	Value	PE (%)	Value	PE (%)
0.015	2115.634	2114.147	0.070	2123.327	0.364
0.020	2132.356	2130.764	0.075	2139.628	0.341
0.025	2149.884	2148.206	0.078	2156.756	0.320
0.030	2168.233	2166.484	0.081	2174.720	0.299
0.035	2187.407	2185.600	0.083	2193.530	0.280
0.040	2207.408	2205.554	0.084	2213.186	0.262
0.045	2228.238	2226.355	0.085	2233.688	0.245
0.050	2249.892	2247.988	0.085	2255.033	0.228

TABLE 19
Steady-state percentage of available PMs $(E[AV])$ resulted from the monolithic, folded, and fixed-point models when λ_{bhf} varies

$\lambda_{\mathbf{bhf}}$	SRN Models				
	Monolithic	Folded		Fixed-point	
	Value	Value	PE (%)	Value	PE (%)
0.015	95.434	95.436	0.002	95.429	0.005
0.020	93.955	93.958	0.003	93.949	0.006
0.025	92.461	92.464	0.003	92.453	0.009
0.030	90.954	90.958	0.004	90.945	0.010
0.035	89.441	89.446	0.006	89.431	0.011
0.040	87.924	87.930	0.007	87.914	0.011
0.045	86.410	86.416	0.007	86.399	0.013
0.050	84.900	84.906	0.007	84.889	0.013

TABLE 20
Steady-state power consumption $(E[PC])$ resulted from the monolithic, folded, and fixed-point models when λ_{bhf} varies (in *watts*)

$\lambda_{\mathbf{bhf}}$	SRN Models				
	Monolithic	Folded		Fixed-point	
	Value	Value	PE (%)	Value	PE (%)
0.015	1109.126	1108.650	0.043	1110.298	0.106
0.020	1091.755	1091.295	0.042	1092.806	0.096
0.025	1074.195	1073.752	0.041	1075.136	0.088
0.030	1056.502	1056.078	0.040	1057.342	0.080
0.035	1038.729	1038.327	0.039	1039.479	0.072
0.040	1020.927	1020.547	0.037	1021.595	0.065
0.045	1003.144	1002.786	0.036	1003.738	0.059
0.050	985.422	985.088	0.034	985.950	0.054

TABLE 21
Steady-state mean response time $(E[RT])$ resulted from the monolithic, folded, and fixed-point models when λ_{ihf} varies (in *seconds*)

$\lambda_{\mathbf{ihf}}$	SRN Models				
	Monolithic	Folded		Fixed-point	
	Value	Value	PE (%)	Value	PE (%)
0.01	2115.634	2114.147	0.070	2123.327	0.364
0.02	2118.701	2117.196	0.071	2126.290	0.358
0.03	2121.793	2120.274	0.072	2129.285	0.353
0.04	2124.914	2123.377	0.072	2132.302	0.348
0.05	2128.061	2126.506	0.073	2135.351	0.343
0.06	2131.236	2129.663	0.074	2138.425	0.337
0.07	2134.433	2132.849	0.074	2141.525	0.332
0.08	2137.658	2136.056	0.075	2144.653	0.327
0.09	2140.909	2139.296	0.075	2147.810	0.322
0.10	2144.185	2142.558	0.076	2150.993	0.318

TABLE 22
Steady-state percentage of available PMs $(E[AV])$ resulted from the monolithic, folded, and fixed-point models when λ_{ihf} varies

$\lambda_{\mathbf{ihf}}$	SRN Models				
	Monolithic	Folded		Fixed-point	
	Value	Value	PE (%)	Value	PE (%)
0.01	95.434	95.436	0.002	95.429	0.005
0.02	95.136	95.138	0.002	95.132	0.004
0.03	94.838	94.840	0.002	94.834	0.004
0.04	94.539	94.541	0.002	94.535	0.004
0.05	94.239	94.241	0.002	94.236	0.003
0.06	93.938	93.940	0.002	93.936	0.002
0.07	93.637	93.639	0.002	93.635	0.002
0.08	93.335	93.337	0.002	93.334	0.001
0.09	93.033	93.034	0.001	93.032	0.001
0.10	92.730	92.731	0.001	92.729	0.001

TABLE 23
Steady-state power consumption $(E[PC])$ resulted from the monolithic, folded, and fixed-point models when λ_{ihf} varies (in *watts*)

$\lambda_{\mathbf{ihf}}$	SRN Models				
	Monolithic	Folded		Fixed-point	
	Value	Value	PE (%)	Value	PE (%)
0.01	1109.126	1108.650	0.043	1110.298	0.106
0.02	1105.641	1105.166	0.043	1106.796	0.104
0.03	1102.147	1101.673	0.043	1103.285	0.103
0.04	1098.644	1098.171	0.043	1099.765	0.102
0.05	1095.132	1094.661	0.043	1096.236	0.101
0.06	1091.613	1091.142	0.043	1092.701	0.100
0.07	1088.086	1087.616	0.043	1089.157	0.098
0.08	1084.552	1084.084	0.043	1085.607	0.097
0.09	1081.011	1080.544	0.043	1082.051	0.096
0.10	1077.464	1076.999	0.043	1078.488	0.095

TABLE 24
Steady-state mean response time $(E[RT])$ resulted from the monolithic, folded, and fixed-point models when λ_{cf} varies (in *seconds*)

$\lambda_{\mathbf{cf}}$	SRN Models				
	Monolithic	Folded		Fixed-point	
	Value	Value	PE (%)	Value	PE (%)
0.001	2115.634	2114.147	0.070	2123.327	0.364
0.002	2115.641	2114.154	0.070	2123.330	0.363
0.003	2115.644	2114.161	0.070	2123.330	0.363
0.004	2115.652	2114.172	0.070	2123.334	0.363
0.005	2115.659	2114.179	0.070	2123.334	0.363
0.006	2115.662	2114.186	0.070	2123.338	0.363
0.007	2115.670	2114.194	0.070	2123.341	0.363
0.008	2115.677	2114.201	0.070	2123.341	0.362
0.009	2115.680	2114.208	0.070	2123.345	0.362
0.010	2115.688	2114.215	0.070	2123.348	0.362

rate of idle hot PMs on output measures is investigated. In this scenario, the failure rate of an idle hot PM, λ_{ihf}, is changed from 0.01 to 0.1 *PMs/hour* with step 0.01. Tables 21, 22, and 23 show the steady-state mean response time, percentage of available PMs, and power consumption, respectively, obtained by solving the monolithic, folded, and fixed-point models. The maximum percent errors of the results obtained by (folded, fixed-point) approximate models in Tables 21, 22, and 23 are $(0.076\%, 0.364\%)$, $(0.002\%, 0.005\%)$, and $(0.043\%, 0.106\%)$, respectively. It can be concluded from these results that, by increasing the failure rate of an idle hot PM, the mean response time increases whereas the percentage of available PMs and power consumption decrease.

In the fourth scenario, the impact of variation of failure rate of cold PMs on output measures is investigated by varying the failure rate of a cold PM, λ_{cf}, from 0.001 to 0.01 *PMs/hour* with step 0.001. Tables 24, 25, and 26 show the steady-state mean response time, percentage of available PMs, and power consumption, respectively. The maximum percent errors of the results obtained by (folded, fixed-point) approximate models in Tables 24, 25, and 26 are $(0.070\%, 0.364\%)$, $(0.002\%, 0.005\%)$, and $(0.044\%, 0.109\%)$, respectively. By increasing the failure rate of a cold PM, the mean response time increases, but the percentage of available PMs and power consumption decrease. However, the impact of variation of failure rate of a cold PM on output measures is negligible. It is worth to mention that since the mean number of tokens in place P_c is around 5×10^{-3}, output measures are not very sensitive to the variation of the failure rate of a cold PM as shown in this scenario.

8 Conclusion and Future Work

In this paper, we aimed to simultaneously analyze the performance, availability, and power consumption of IaaS cloud systems. To achieve this, an SRN model was proposed for a cluster of PMs within a cloud data center. The proposed model captures several realistic aspects of an IaaS cloud including different pools of PMs, provisioning delays, servicing process, and failure/repair behavior of PMs. The model also exploits a strategy for moving PMs between cold and hot pools in order to minimize power consumption of the cluster. The SRN model presented for a single cluster was then used to model a more realistic IaaS cloud system

TABLE 25
Steady-state percentage of available PMs $(E[AV])$ resulted from the monolithic, folded, and fixed-point models when λ_{cf} varies

λ_{cf}	SRN Models				
	Monolithic	Folded		Fixed-point	
	Value	Value	PE (%)	Value	PE (%)
0.001	95.434	95.436	0.002	95.429	0.005
0.002	95.433	95.435	0.002	95.429	0.004
0.003	95.433	95.435	0.002	95.428	0.005
0.004	95.432	95.434	0.002	95.428	0.004
0.005	95.432	95.433	0.001	95.428	0.004
0.006	95.431	95.432	0.001	95.428	0.003
0.007	95.430	95.432	0.002	95.427	0.003
0.008	95.430	95.431	0.001	95.427	0.003
0.009	95.429	95.430	0.001	95.427	0.002
0.010	95.429	95.429	0.000	95.427	0.002

TABLE 26
Steady-state power consumption $(E[PC])$ resulted from the monolithic, folded, and fixed-point models when λ_{cf} varies (in *watts*)

λ_{cf}	SRN Models				
	Monolithic	Folded		Fixed-point	
	Value	Value	PE (%)	Value	PE (%)
0.001	1109.126	1108.650	0.043	1110.298	0.106
0.002	1109.119	1108.642	0.043	1110.295	0.106
0.003	1109.112	1108.634	0.043	1110.292	0.106
0.004	1109.106	1108.625	0.043	1110.289	0.107
0.005	1109.099	1108.617	0.043	1110.287	0.107
0.006	1109.092	1108.609	0.044	1110.284	0.107
0.007	1109.086	1108.601	0.044	1110.281	0.108
0.008	1109.079	1108.592	0.044	1110.278	0.108
0.009	1109.072	1108.584	0.044	1110.276	0.109
0.010	1109.065	1108.576	0.044	1110.273	0.109

containing many clusters in a hierarchical form. Since the monolithic SRN model encounters the largeness problem, two approximate models called folded and fixed-point were proposed to solve this problem. Comparing the results obtained from monolithic model with the results obtained from our proposed approximate models showed that the accuracy of measures is not compromised when exploiting approximation techniques. Therefore, cloud providers can benefit from the proposed modeling approaches during design, development, and maintenance of an IaaS cloud.

Following are some ideas and issues which can be used for further research in this area:

- The design and analysis of more complicated strategies and policies at top-level management layer of the reference cloud architecture is an interesting work. For example, if the performance, availability, and power consumption measures of the individual clusters are available at the top-level resource manager, it can dispatch the requests more efficiently applying smart load distribution techniques. The top-level resource manager may try to minimize the number of active clusters and reduce the overall power consumption by switching inactive clusters to low-power mode.
- By consolidating requested VMs on fewer PMs, both the utilization of switched-on servers and the energy efficiency of the cloud data center are improved. The reduction of total power consumption, when consolidation is adopted, is due to the non energy-proportional characteristics of real servers. One future research direction is to extend our proposed models to support the VM consolidation by designing proper scheduling policies at the bottom-level management layer, i.e., cluster level. The VM migration, which is an important part of the consolidation, is another concern that can be considered in the future work
- Extending the models presented in this paper to support heterogeneous PMs and VMs is another interesting research line which can be developed. Considering heterogeneous PMs, we can model different classes of PMs with different processing speeds, storages, memory capacities, failure/repair rates, and power consumptions. Moreover, heterogeneous VMs can be modeled with different requirements. One way to support the heterogeneity is to add some auxiliary places and transitions to the proposed models representing different classes of VMs and PMs and their associated events like arrival, servicing, and failure. The penalty of such an approach is fast growth of the state space when several classes of VMs or PMs need to be modeled. Another way to support the heterogeneity is to use Colored Petri Nets (CPNs) which is an extension to PNs that allows tokens to have data types, making it possible for the modeler to differentiate between tokens representing different classes of VMs and PMs.
- Exploiting the proposed models of a cluster and a data center, and increasing the number of management layers of the reference cloud architecture, geo-distributed cloud system containing many different data centers can be modeled. The fixed-point approximate model seems promising to alleviate the largeness problem of very large-scale IaaS cloud systems.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] 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.

[3] F. Machida, D. S. Kim, and K. S. Trivedi, "Modeling and analysis of software rejuvenation in a server virtualized system with live VM migration," *Performance Evaluation*, vol. 70, no. 3, pp. 212–230, 2013.

[4] Y. Han, J. Chan, T. Alpcan, and C. Leckie, "Using virtual machine allocation policies to defend against co-resident attacks in cloud computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 1, pp. 95–108, 2017.

[5] J. Subirats and J. Guitart, "Assessing and forecasting energy efficiency on cloud computing platforms," *Future Generation Computer Systems*, vol. 45, pp. 70–94, 2015.

[6] R. Ghosh, F. Longo, V. K. Naik, and K. S. Trivedi, "Modeling and performance analysis of large scale IaaS clouds," *Future Generation Computer Systems*, vol. 29, no. 5, pp. 1216–1234, 2013.

[7] L. Zhao, S. Sakr, and A. Liu, "A framework for consumer-centric SLA management of cloud-hosted databases," *IEEE Transactions on Services Computing*, vol. 8, no. 4, pp. 534–549, 2015.

[8] D. Bruneo, "A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 560–569, 2014.

[9] K. Bilal, S. U. R. Malik, S. U. Khan, and A. Y. Zomaya, "Trends and challenges in cloud datacenters," *IEEE Cloud Computing*, vol. 1, no. 1, pp. 10–20, 2014.

[10] "Downtime, outages and failures - understanding their true costs," http://urlm.in/sjhk, accessed: July 2017.

[11] K. V. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability," in *The 1st ACM symposium on Cloud Computing*, Indianapolis, IN, US, June 2010, pp. 193–204.

[12] J. Koomey, "Growth in data center electricity use 2005 to 2010," *A report by Analytical Press, completed at the request of The New York Times*, 2011.

[13] T. Mastelic, A. Oleksiak, H. Claussen, I. Brandic, J.-M. Pierson, and A. V. Vasilakos, "Cloud computing: Survey on energy efficiency," *ACM Computing Surveys*, vol. 47, no. 2, pp. 1–35, 2015.

[14] T. Mastelic and I. Brandic, "Recent trends in energy-efficient cloud computing," *IEEE Cloud Computing*, vol. 2, no. 1, pp. 40–47, 2015.

[15] X. Chang, R. Xia, J. K. Muppala, K. S. Trivedi, and J. Liu, "Effective modeling approach for IaaS data center performance analysis under heterogeneous workload," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, 2016.

[16] R. Entezari-Maleki, K. S. Trivedi, and A. Movaghar, "Performability evaluation of grid environments using stochastic reward nets," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 2, pp. 204–216, 2015.

[17] D. Bruneo, A. Lhoas, F. Longo, and A. Puliafito, "Modeling and evaluation of energy policies in green clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 11, pp. 3052–3065, 2015.

[18] R. Entezari-Maleki, M. Bagheri, S. Mehri, and A. Movaghar, "Performance aware scheduling considering resource availability in grid computing," *Engineering with Computers*, vol. 33, no. 2, pp. 191–206, 2017.

[19] R. Entezari-Maleki, L. Sousa, and A. Movaghar, "Performance and power modeling and evaluation of virtualized servers in IaaS clouds," *Information Sciences*, vol. 394, pp. 106–122, 2017.

[20] K. S. Trivedi, *Probability and Statistics With Reliability, Queuing and Computer Science Applications*, 2nd ed. John Wiley and Sons, 2002.

[21] J. K. Muppala and K. S. Trivedi, "Composite performance and availability analysis using a hierarchy of stochastic reward nets," in *Computer Performance Evaluation, Modelling Techniques and Tools*, G. Balbo and G. Serazzi, Eds. Elsevier Science Publishers B.V. (North-Holland), 1992, pp. 335–349.

[22] H. Choi and K. S. Trivedi, "Approximate performance models of polling systems using stochastic petri nets," in *IEEE INFOCOM*, Florence, Italy, May 1992, pp. 2306–2314.

[23] O. C. Ibe, H. Choi, and K. S. Trivedi, "Performance evaluation of client-server systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 11, pp. 1217–1229, 1993.

[24] V. Mainkar and K. S. Trivedi, "Sufficient conditions for existence of a fixed point in stochastic reward net-based iterative models," *IEEE Transactions on Software Engineering*, vol. 22, no. 9, pp. 640–653, 1996.

[25] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modeling with Generalized Stochastic Petri Nets*, 1st ed. John Wiley and Sons, 1995.

[26] K. Kant and M. M. Srinivasan, *Introduction to Computer System Performance Evaluation*, 1st ed. McGraw-Hill, 1992.

[27] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[28] D. Bruneo, F. Longo, and A. Puliafito, "Evaluating energy consumption in a cloud infrastructure," in *The IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, Lucca, Italy, June 2011, pp. 1–6.

[29] R. Ghosh, K. S. Trivedi, V. K. Naik, and D. S. Kim, "End-to-end performability analysis for Infrastructure-as-a-Service cloud: An interacting stochastic models approach," in *The IEEE 16th Pacific Rim International Symposium on Dependable Computing*, Tokyo, Japan, December 2010, pp. 125–132.

[30] H. Khazaei, J. Misic, and V. B. Misic, "A fine-grained performance model of cloud computing centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 11, pp. 2138–2147, 2013.

[31] R. Ghosh, F. Longo, F. Frattini, S. Russo, and K. S. Trivedi, "Scalable analytics for IaaS cloud availability," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 57–70, 2014.

[32] R. Ghosh, F. Longo, R. Xia, V. K. Naik, and K. S. Trivedi, "Stochastic model driven capacity planning for an infrastructure-as-a-service cloud," *IEEE Transactions on Services Computing*, vol. 7, no. 4, pp. 667–680, 2014.

[33] H. Khazaei, J. Misic, V. B. Misic, and S. Rashwand, "Analysis of a pool management scheme for cloud computing centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 5, pp. 849–861, 2013.

[34] D. Bruneo, A. Lhoas, F. Longo, and A. Puliafito, "Analytical evaluation of resource allocation policies in green IaaS clouds," in *The 3rd International Conference on Cloud and Green Computing*, Karlsruhe, Germany, September-October 2013, pp. 84–91.

[35] A. Castiglione, M. Gribaudo, M. Iacono, and F. Palmieri, "Modeling performances of concurrent big data applications," *Software: Practice and Experience*, vol. 45, no. 8, pp. 1127–1144, 2015.

[36] E. Barbierato, M. Gribaudo, and M. Iacono, "Modeling and evaluating the effects of big data storage resource allocation in global scale cloud architectures," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 12, no. 2, pp. 1–20, 2016.

[37] G. Ciardo, M. Gribaudo, M. Iacono, A. Miner, and P. Piazzolla, "Power consumption analysis of replicated virtual applications in heterogeneous architectures," in *Digitally Supported Innovation*. Springer, 2016, pp. 285–297.

[38] B. Addis, D. Ardagna, B. Panicucci, M. S. Squillante, and L. Zhang, "A hierarchical approach for the resource management of very large cloud platforms," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 5, pp. 253–272, 2013.

[39] X. Qiu, Y. Dai, Y. Xiang, and L. Xing, "A hierarchical correlation model for evaluating reliability, performance, and power consumption of a cloud service," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 3, pp. 401–412, 2016.

[40] M. Sedaghat, F. Hernandez-Rodriguez, and E. Elmroth, "Decentralized cloud datacenter reconsolidation through emergent and topology-aware behavior," *Future Generation Computer Systems*, vol. 56, pp. 51–63, 2016.

[41] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, and H. Tenhunen, "Hierarchical VM management architecture for cloud data centers," in *The IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom)*, Singapore, Singapore, December 2014, pp. 306–311.

[42] B. C. Gunaratne, K. Christensen, and B. Nordman, "Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed," *International Journal of Network Management*, vol. 15, no. 5, pp. 297–310, 2005.

[43] B. Javadi, D. Kondo, A. Iosup, and D. Epema, "The failure trace archive: Enabling the comparison of failure measurements and models of distributed systems," *Journal of Parallel and Distributed Computing*, vol. 73, no. 8, pp. 1208–1223, 2013.

[44] "WebSphere application server," http://www-03.ibm.com/software/products/en/appserv-was, accessed: July 2017.

[45] M. Nabi, M. Toeroe, and F. Khendek, "Availability in the cloud: State of the art," *Journal of Network and Computer Applications*, vol. 60, pp. 54–67, 2016.

[46] Y. Geng, S. Chen, Y. Wu, R. Wu, G. Yang, and W. Zheng, "Location-aware MapReduce in virtual cloud," in *The International Conference on Parallel Processing (ICPP)*, Taipei City, Taiwan, September 2011, pp. 275–284.

[47] A. Kanso and Y. Lemieux, "Achieving high availability at the application level in the cloud," in *The IEEE 6th International Conference on Cloud Computing (CLOUD)*, Santa Clara, CA, June-July 2013, pp. 778–785.

[48] W. Li, A. Kanso, and A. Gherbi, "Leveraging linux containers to achieve high availability for cloud services," in *The IEEE Interna-*

tional Conference on Cloud Engineering (IC2E), Tempe, AZ, March 2015, pp. 76–83.
[49] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, 1st ed. Prentice Hall, 1981.
[50] M. A. Marsan, G. Conte, and G. Balbo, "A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems," *ACM Transactions on Computer Systems*, vol. 2, no. 2, pp. 93–122, 1984.
[51] F. Bause and P. S. Kritzinger, *Stochastic Petri Nets- An Introduction to the Theory*, 2nd ed. Vieweg Verlag, 2002.
[52] F. F. Moghaddam, "Carbon-profit-aware job scheduling and load balancing in geographically distributed cloud for HPC and web applications," Ph.D. dissertation, Ecole de Technologie Superieure, 2014.
[53] I. Hwang and M. Pedram, "Hierarchical virtual machine consolidation in a cloud computing system," in *The IEEE 6th International Conference on Cloud Computing*, Santa Clara, CA, June-July 2013, pp. 196–203.
[54] L. A. Tomek and K. S. Trivedi, "Fixed point iteration in availability modeling," in *The 5th International GI/ITG/GMA Conference on Fault-Tolerant Computing Systems, Tests, Diagnosis, Fault Treatment*, Nrnberg, Germany, September 1991, pp. 229–240.
[55] G. Ciardo, A. Blakemore, P. F. Chimento, J. K. Muppala, and K. S. Trivedi, "Automated generation and analysis of markov reward models using stochastic reward nets," in *Linear Algebra, Markov Chains, and Queueing Models*, ser. The IMA Volumes in Mathematics and its Application, C. D. Meyer and R. J. Plemmons, Eds., vol. 48. Springer, 1993, pp. 145–191.
[56] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: Eliminating server idle power," in *The 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, Washington, DC, March 2009, pp. 205–216.
[57] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *The 1st ACM Symposium on Cloud Computing*, Indianapolis, IN, June 2010, pp. 39–50.
[58] Q. Chen, P. Grosso, K. van der Veldt, C. de Laat, R. Hofman, and H. Bal, "Profiling energy consumption of VMs for green cloud computing," in *The IEEE 9th International Conference on Dependable, Autonomic and Secure Computing (DASC)*, Sydney, Australia, December 2011, pp. 768–775.
[59] J. Bi, Z. Zhu, R. Tian, and Q. Wang, "Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center," in *The IEEE 3rd International Conference on Cloud Computing (CLOUD)*, Miami, FL, July 2010, pp. 370–377.
[60] G. Ciardo, J. Muppala, and K. S. Trivedi, "SPNP: stochastic petri net package," in *The 3rd International Workshop on Petri Nets and Performance Models*, Kyoto, Japan, December 1989, pp. 142–151.

Ehsan Ataie is a Ph.D. candidate in Computer Engineering at the Department of Computer Engineering in the Sharif University of Technology, Tehran, Iran. He received his B.S. and M.S. degrees from the same university in 2002 and 2005, respectively. He visited Politecnico di Milano in 2016. His main research interests include cloud computing, green computing, and performance and dependability modeling and evaluation.

Reza Entezari-Maleki is a Post-Doctoral Researcher in the School of Computer Science at Institute for Research in Fundamental Sciences (IPM) in Tehran, Iran. He received his Ph.D. in Computer Engineering from the Sharif University of Technology, Tehran, Iran in 2014, and M.S. and B.S. degrees in Computer Engineering from the Iran University of Science and Technology, Tehran, Iran in 2009 and 2007, respectively. His main research interests are performance/dependability modeling and evaluation, distributed computing systems, cloud computing, and task scheduling algorithms.

Leila Rashidi is currently a Ph.D. candidate in Computer Engineering (Software discipline) at the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. She received the B.S. degree in Computer Engineering (Software discipline) from the University of Tehran (UT), Tehran, Iran, in 2014. Her main research interests are performance analysis, wireless networks, and mobility modeling.

Kishor S. Trivedi received the M.S. and Ph.D. degrees in computer science from the University of Illinois, Urbana. He holds the Hudson Chair with the Department of Electrical and Computer Engineering, Duke University, Durham, NC. Since 1975, he has been with the Duke faculty. He is the author of the well-known text entitled Probability and Statistics with Reliability, Queuing and Computer Science Applications (Prentice-Hall); a thoroughly revised second edition (including its Indian edition) of this book has been published by John Wiley. He is also the author of two other books, one entitled Performance and Reliability Analysis of Computer Systems (Kluwer) and the other entitled Queuing Networks and Markov Chains (John Wiley). He has published more than 500 papers and has supervised 45 Ph.D. dissertations. His research interests are reliability, availability, performance, performability, and survivability modeling of computer and communication systems. Dr. Trivedi is a Life Fellow of the IEEE and is a Golden Core Member of the IEEE Computer Society.

Danilo Ardagna is an Associate Professor at the Dipartimento di Elettronica Informazione and Bioingegneria at Politecnico di Milano, Milan, Italy. He received the Ph.D. degree in Computer Engineering from Politecnico di Milano in 2004. His work focuses on the design, prototype and evaluation of optimization algorithms for resource management and planning of cloud systems.

Ali Movaghar is a Professor in the Department of Computer Engineering at Sharif University of Technology in Tehran, Iran and has been on the Sharif faculty since 1993. He received his B.S. degree in Electrical Engineering from the University of Tehran in 1977, and M.S. and Ph.D. degrees in Computer, Information, and Control Engineering from the University of Michigan, Ann Arbor, in 1979 and 1985, respectively. His research interests include performance/dependability modeling and formal verification of wireless networks and distributed real-time systems. He is a senior member of the IEEE and the ACM.