A Fine-Grained Performance Model of Cloud Computing Centers

Hamzeh Khazaei, *Student Member, IEEE,* Jelena Mišić, *Senior Member, IEEE,* and Vojislav B. Mišić, *Senior Member, IEEE*

Abstract—Accurate performance evaluation of cloud computing resources is a necessary prerequisite for ensuring that quality of service (QoS) parameters remain within agreed limits. In this paper, we employ both the analytical and simulation modeling to addresses the complexity of cloud computing systems. Analytical model is comprised of distinct functional sub-models, the results of which are combined in an iterative manner to obtain the solution with required accuracy. Our models incorporate the important features of cloud centers such as batch arrival of user requests, resource virtualization and realistic servicing steps, to obtain important performance metrics such as task blocking probability and total waiting time incurred on user requests. Also, our results reveal important insights for capacity planning in order to control delay of servicing users requests.

Index Terms—cloud computing, performance analysis, response time, blocking probability, queuing theory, interacting Markov model, fixed-point iteration.

1 Introduction

Cloud Computing is a computing paradigm in which different computing resources such as infrastructure, platforms and software applications are made accessible over the Internet to remote user as services [1]. Infrastructure-as-a-Service (IaaS) cloud providers, such as Amazon EC2 [2], IBM Cloud [3], GoGrid [4], NephoScale [5], Rackspace [6] and others, deliver, on-demand, operating system (OS) instances provisioning computational resources in the form of virtual machines deployed in the cloud provider data center. A cloud service differs from traditional hosting in three principal aspects. First, it is provided on demand; second, it is elastic since users that use the service have as much or as little as they want at any given time (typically by the minute or the hour); and third, the service is fully managed by the provider [7], [8]. Due to dynamic nature of cloud environments, diversity of user requests, and time dependency of load, providing agreed quality of service (QoS) while avoiding over-provisioning is a difficult task [9].

Service availability and response time are two important quality measures in cloud's users prospective [10]. Quantifying and characterizing such performance measures requires appropriate modeling; the model ought to cover vast parameter space while it is still tractable. A monolithic model may suffer from intractability and poor scalability due to large number of parameters [11]. Instead, in this paper we develop and evaluate tractable functional sub-models and their interaction model while iteratively solve them. We construct separate sub-models for different servicing steps in a complex cloud center and the overall solution obtain by iteration over individual sub-model solutions [12]. We assume that the cloud center consists of a number of Physical Machines (PM) that are allocated to users in the order of task arrivals. More specifically, user requests may share a PM using virtualization technique. Since many of the large cloud centers employ virtualization to provide the required resources such as PMs [13], we consider PMs with a high degree of virtualization. Real cloud providers offer complex requests for their users. For instance, in Amazon EC2, the user is allowed to run up to 20 On-Demand or Reserved Instances, and up to 100 Spot Instances per region [2]. Such complex requests are referred to as *super-tasks*. Our proposed model embraces realistic aspects and provides deep insight into performance analysis of today's cloud centers. The main features of our analytical model can be listed as follows:

- Assumes Poisson arrival of user requests;
- Incorporates complex user requests by introducing super-tasks;
- Supports high degree of virtualization;
- Captures different delays imposed by cloud centers on user requests;
- Characterizes the service availability at cloud center;
- Provides information for capacity planning;
- Offers tractability and scalability;

The rest of the paper is organized as follows. In Section 2, we survey related work in cloud center performance analysis. Section 3 presents our model and the details of the analysis. Section 4 introduce the stochastic sub-models and their interactions. Section 5 presents the numerical results obtained from the analytical model. Finally, section

- *H. Khazaei is with the Department of Computer Science, University of Manitoba, Winnipeg, Manitoba. E-mail: hamzehk@cs.umanitoba.ca*
- *J. Mišić and V. B. Mišić are with the Department of Computer Science, Ryerson University, Toronto, Ontario. E-mails: {jmisic,vmisic}@scs.ryerson.ca*

6 summarizes our findings and concludes the paper.

2 Related Work

Cloud computing has attracted considerable research attention, but only a small portion of the work done so far has addressed performance issues by rigorous analytical models. Many research works carried out a measurement-based performance evaluation of the Amazon EC2 [2], IBM Blue Cloud [3] or other cloud providers in the context of scientific computing [14], [15], [16], [17], [18], [19], [20], [21]. Here we survey those that proposed a general analytical model for performance evaluation of cloud centers.

In [22], the authors studied the response time in terms of various metrics, such as the overhead of acquiring and realizing the virtual computing resources, and other virtualization and network communication overhead. To address these issues, they have designed and implemented C-Meter, a portable, extensible, and easy-to-use framework for generating and submitting test workloads to computing clouds. In [23], a cloud center is modeled as the classic open network with single arrival, from which the distribution of response time is obtained, assuming that both inter-arrival and service times are exponential. Using the distribution of response time, the relationship among the maximal number of tasks, the minimal service resources and the highest level of services was found.

In [24], the cloud center was modeled as an $M/M/m/m+r$ queuing system from which the distribution of response time was determined. The response time was broken down into waiting, service, and execution periods, assuming that all three periods are independent (which is unrealistic, according to authors' own argument). Our earlier work [25], [10], [26] presents monolithic analytical models which are much more restrictive than our current work in terms of extendability simplicity and cost of solution. Also, that work does not address the concept of virtualization as well as heterogeneous server pools and PMs.

The performance of cloud computing services for scientific computing workloads was examined in [14]. The authors quantified the presence of Many-Task Computing (MTC) users in real scientific computing workloads. Then, they performed an empirical evaluation of the performance of four commercial cloud computing services including Amazon EC2.

In [27], the authors quantified the resiliency of IaaS cloud with respect to changes in demand and available capacity. Using a stochastic reward net based model for provisioning and servicing requests in a IaaS cloud, they quantified the resiliency of IaaS cloud with respect to two key performance measures, namely, task rejection rate and total response delay. In this work, only, the effect of abrupt changes in arrival process and system capacity (i.e., physical machines) have been studied on mentioned performance metrics.

A hierarchical modeling approach for evaluating quality of experience in a cloud computing environment was proposed in [28]. Due to simplified assumptions, authors applied classical Erlang loss model and $M/M/m/K$ queuing system for outbound bandwidth and response time modeling respectively. Moreover, since services usually share the bandwidth, it seems that a processor sharing approach is more appropriate model than loss model to be used in the bandwidth model.

In [12], the authors proposed a general analytic model based approach for an end-to-end performance analysis of a cloud service. They illustrated their approach using IaaS cloud, where service availability and provisioning response delays are two key QoS metrics. The proposed approach reduces the complexity of performance analysis of cloud centers by dividing the overall model into sub-models and then obtaining the overall solution by iteration over individual sub-model solutions. The model is limited to single request arrivals, which is not quite realistic; cloud users may ask for multiple PMs or VMs by submitting in a single request. In addition, the effect of virtualization has not been reflected explicitly in their results.

As a result, the lack of scalable and tractable model that embraces important features of computing cloud centers such as batch arrival of requests, virtualization and complex servicing steps motivated us to develop a comprehensive performance model that meets above mentioned aspects.

3 Analytical Model

In IaaS cloud, when a request is processed, a pre-built or customized disk image is used to create one or more VM instances [27]. In this work we assume that pre-built images fulfill all user requests. We assume that Physical Machines (PMs) are categorized into three server pools: hot (i.e., with running VMs), warm (i.e., turned on but without running VM) and cold (i.e., turned off) [12]. Such categorization reduces operational costs and offers the capability for disaster recovery [2], [29]. Each PM has up to two instantiation units that configure and deploy VMs on a PM. Instantiation of a VM from an image and provisioning it on hot PMs has the minimum delay compared to warm and cold PMs. Warm PMs require more time to be ready for provisioning and cold PMs require additional time to be started up before a VM instantiation. In this work we allow users to request more than one VM by submitting a *super-task* [26]. So, a super-task may contain more than one task, each of which requires one VM. The size of super-task is assumed to be geometrically distributed. However, for practical reason we set a maximum size of super-tasks (MSS) to conduct the numerical analysis (i.e., truncated geometric distribution). Therefore the probability of having a super-task with size i is

$$p_i = \begin{cases} (1-p)^{i-1}p, & \text{if } i < MSS \\ (1-p)^{MSS-1}, & \text{if } i = MSS \end{cases} \quad (1)$$

in which p is the probability of success in geometric distribution.

Due to high interaction among tasks within a super-task, each super-task will be provisioned on the same PM.

However, the number of tasks (VMs) requested by a single super-task cannot exceed the maximum number of VMs that can run on a single PM. In this manner the inter-tasks communication overhead among tasks within a given super-task will be decreased significantly. In this work, we assume that all PMs and VMs are homogeneous and adopt *total acceptance policy* [30]. Under this policy the system tries to service all tasks within a super-task at the same time; provided that partial admission of a super-task is not considered. User requests (super-tasks) are submitted to a global finite queue and then processed on the first-in, first-out basis (FIFO).

Resource Assigning Module (RAM) processes the super-task at the head of global queue as follows: first, it tries to provision the super-task on a PM machine in the hot pool. If the process is not successful then RAM tries the warm pool and finally the cold pool. Ultimately, RAM either assigns a PM in one of the pools to a super-task or the super-task gets rejected. As a result, user requests (super-tasks) may get rejected either due to lack of space in global input queue or insufficient resource at the cloud center. When a running task finishes, the capacity used by that VM is released and becomes available for servicing the next task.

Table 1 describes the symbols and acronyms that we use in next sections.

TABLE 1
Symbols and corresponding descriptions

Symbol	Description
RAM	Resource Assigning Module
VMPM	Virtual Machine Provisioning Module
RASM	Resource Allocation Sub-Model
VMPSM	Virtual Machine Provisioning Sub-Model
MSS	Maximum Super-task Size
p_i	Probability of having a super-task with size i
λ_{st}	Mean arrival rate of super-tasks
L_q	Size of global queue
P_h, P_w, P_c	Prob. of success in hot, warm and cold pool
N_h, N_w, N_c	Number of PMs in hot, warm and cold pool
ϕ_h, ϕ_w, ϕ_c	Instantiation rate of VMs in hot, warm or cold pool
$\lambda_h, \lambda_w, \lambda_c$	Arrival rate to a PM in hot, warm and cold pool
$\alpha_h, \alpha_w, \alpha_c$	Look up rate in hot, warm and cold pool
BP_q	Prob. of blocking due to lack of room in global queue
BP_r	Prob. of blocking due to lack of capacity
P_{reject}	Total probability of blocking ($BP_q + BP_r$)
$\overline{wt}$	Mean waiting time in global queue
$\overline{lut}$	Mean look-up delay among pools
$\overline{wt_{PM}}$	Mean waiting time in a PM queue
$\overline{pt}$	Mean VM provisioning time
$\overline{td}$	Mean total delay for a task before getting into service

We identify four main delays imposed on user requests (Fig. 1): queuing delay in the global queue; look-up time at the RAM; queuing delay at the chosen PM; and VM provisioning delay. The response time is the sum of the sum of these delays and the actual service time. In order to model each module precisely, we design two stochastic sub-models, captures the details of the RAM and VMPM modules. We then connect these sub-models into an overall model to compute the cloud performance metrics: task rejection probability and mean response delay. Finally, we solve these sub-models and show how performance metrics are affected by variations in workload (super-task arrival rate, super-task size, task mean service time, number of VMs per PM) and system capacity (i.e., number of PMs in each pool). We describe our analysis in the following sections.

4 Integrated Stochastic Models

As mentioned in section 2, due to high complexity of cloud centers, the proposed singular models were hard to analyze and extend. By introducing interacting analytical sub models, the complexity of system is divided into sub-models so that they can be analyzed accurately in order of processing of task components. Moreover, the sub-models are scalable enough to capture the flexibility (on-demand services) of the cloud computing paradigm. In this paper, we implement the sub-models using interactive Continuous Time Markov Chain (CTMC). The sub-models are interactive such that the output of one sub-model is input to the other ones and vice versa.

4.1 Resource Allocation Sub-Model

The resource allocation process is described in the resource allocation sub-model (RASM) shown in Fig. 2. RASM is a two dimensional CTMC in which we take care of number of super-task in the queue as well as the current pool on which provisioning is taking place. Since the number of potential users is high, but each user submits super-tasks one at a time with low probability, the super-task arrival can be modeled as a Poisson process [31] with rate λ_{st}. Super-tasks are lined up in the global finite queue to be processed in a first-in, first-out (FIFO) basis. Each state of Markov chain is labeled as (i, j), where i indicates the number of super-tasks in queue and j denotes the pool on which the leading super-task is under provisioning. State $(0, 0)$ indicates that system is empty which means there is no request under provisioning or in the queue. Index j can be h, w or c that indicate current super-task is undergoing provisioning on hot, warm or cold pool respectively. The global queue size is L_q and one more super-task can be at the deployment unit for provisioning so the total system capacity is $L_q + 1$.

Let P_h, P_w and P_c be the success probabilities of finding a PM that can accept the current super-task in hot, warm and cold pool respectively. We assume that $1/\alpha_h$, $1/\alpha_w$ and $1/\alpha_c$ are the mean look-up delays for finding an appropriate PM in hot, warm and cold pool respectively. Upon arrival of first super-task, system moves to state $(0, h)$ which means the super-task will be provisioned immediately in the hot pool. Afterwards, depending on the upcoming event, three possible transitions can occur:

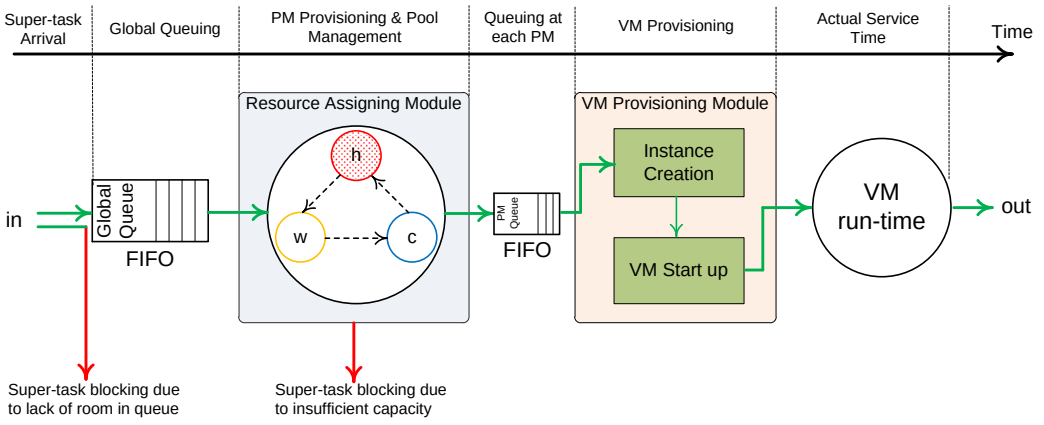


Fig. 1. The steps of servicing and corresponding delays.

(a) Another super-task has arrived and system transits to state $(1, h)$ with rate λ_{st}.
(b) A PM in hot pool accepts the super-task so that system moves back to state $(0, 0)$ with rate $P_h\alpha_h$.
(c) None of PMs in hot pool has enough capacity to accept the super-task, so the system will examine the warm pool (i.e., transit to state $(0, w)$) with rate $(1 - P_h)\alpha_h$.

On state $(0, w)$, RASM tries to provision the super-task on warm pool; if one of the PMs in warm pool can accommodate the super-task, the system will get back to $(0, 0)$, otherwise, RASM examines the cold pool (i.e., transition from state $(0, w)$ to $(0, c)$). If none of the PMs in cold pool can provision the super-task as well, the system moves back from $(0, c)$ to $(0, 0)$ with rate $(1-P_c)\alpha_c$ which means the user request (super-task) will get rejected due to insufficient resources in the cloud center. During the provisioning in cold pool, another super-task may arrive and takes the system to state $(1, c)$ which means there is one super-task in deployment unit and one awaiting in global queue. Finally, the super-task under provisioning decision leaves the deployment unit, once it receives a decision from RASM and the next super-task at the head of global queue will go under provisioning. In this sub-model, arrival rate of super-task (λ_{st}) and look-up delays ($1/\alpha_h$, $1/\alpha_w$ and $1/\alpha_c$) are exogenous parameters and success probabilities (P_h, P_w and P_c) are calculated from the VM provisioning sub-model. The VM provisioning sub-model will be discussed in the next section.

Using steady-state probabilities $\pi_{(i,j)}$, some performance metrics such as blocking probability and probability of immediate service can be calculated. Two types of blocking may happen to a given super-task:

(a) Blocking due to a full global queue occurs with the probability of

$$BP_q = \pi_{(L_q,h)} + \pi_{(L_q,w)} + \pi_{(L_q,c)} \tag{2}$$

(b) Blocking due to insufficient resources (PMs) at pools

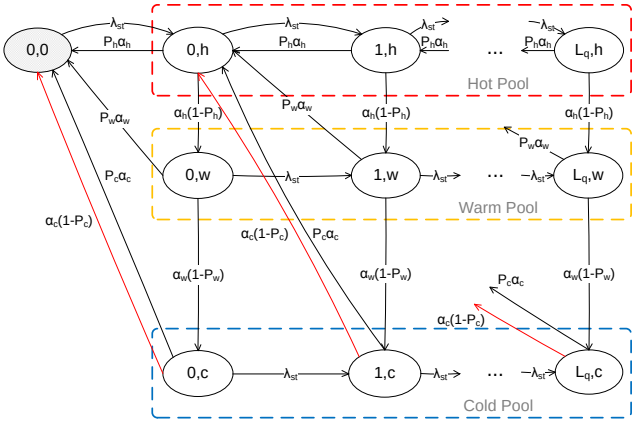


Fig. 2. Resource allocation sub-model

[32], with the probability of

$$BP_r = \sum_{i=0}^{L_q} \frac{\alpha_c(1 - P_c)}{\alpha_c + \lambda_{st}} \pi_{(i,c)} \tag{3}$$

The probability of reject (P_{reject}) is, then, $P_{reject} = BP_q + BP_r$. In order to calculate the mean waiting time in queue, we first establish the probability generating function (PGF) for the number of super-tasks in the queue [33],

$$Q(z) = \pi_{(0,0)} + \sum_{i=0}^{L_q} (\pi_{(i,h)} + \pi_{(i,w)} + \pi_{(i,c)}) z^i \tag{4}$$

The mean number of super-tasks in queue ($\overline{q}$) is the first derivative of $Q(z)$ at $z = 1$ [30],

$$\overline{q} = Q'(1) \tag{5}$$

Applying Little's law [34], the mean waiting time in queue ($\overline{wt}$) is given by:

$$\overline{wt} = \frac{\overline{q}}{\lambda_{st}(1 - P_{reject})} \tag{6}$$

Look-up time among pools can be considered as a Coxian distribution with 3 steps (Fig. 3). So it can be calculated

as [34], [35],

$$\overline{lut} = \frac{1/\alpha_h + (1-P_h)((1/\alpha_w) + (1-P_w)(1/\alpha_c))}{1 - BP_q} \quad (7)$$

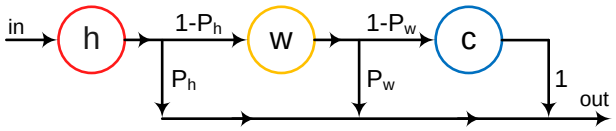


Fig. 3. Three steps of look-up delays among pools.

4.2 VM Provisioning Sub-Model

Virtual Machine Provisioning Sub-Model (VMPSM) captures the instantiation, deployment and provisioning of VMs on a PM. VMPSM also incorporate the actual servicing of each task (VM) on a PM. Note that each task within super-task is provisioned with an individual VM. However RASM makes sure that all tasks within a super-task get provisioned at the same PM. In this model, we assume homogeneous PMs, VMs and tasks. As can be seen from Fig. 3, the look-up time for finding a proper PM among pools has Coxian distribution and the look-up time is a weighted sum of an exponentially distributed random variable with the probability of P_h, a 2-stage Erlang distributed random variable with the probability of $(1-P_h)P_w$, and a 3-stage Erlang distributed random variable with the probability of $(1-P_h)(1-P_w)P_c$. Hence, the Laplace-Stieltjes Transform (LST) of look-up time can be calculated as:

$$B^*(s) = P_h(\frac{\alpha_h}{s+\alpha_h}) + P_w(1-P_h)(\frac{\alpha_h}{s+\alpha_h})(\frac{\alpha_w}{s+\alpha_w}) + (1-P_h)(1-P_w)(\frac{\alpha_h}{s+\alpha_h})(\frac{\alpha_w}{s+\alpha_w})(\frac{\alpha_c}{s+\alpha_c}) \quad (8)$$

which allows us to calculate the mean, standard deviation and coefficient of variation (CoV) of the look-up time at the PM pools. As RASM is not an M/M/1 queuing system, the output process (which is also the arrival process to the PMs) is not Poisson [36] so, strictly speaking, the VMPSM cannot be modeled with a CTMC [34]. However, if the CoV of the arrival process to the PMs is not significantly lower than 1, we can approximate it with a Poisson process. This assumption, which will be justified in Section 5, allows us to model the process with sufficient accuracy.

Fig. 4 shows the VMPSM (an approximated CTMC) for a PM in hot pool. A PM in warm or cold pool can be modeled with the same VMPSM, though, with different arrival and instantiation rate. Consequently, each pool (hot, warm and cold) can be modeled as a set of VMPSM with the same arrival and instantiation rate. Each state in Fig. 4 is labeled by (i,j,k) in which i indicates the number of tasks in PM's queue, j denotes the number of task that is under provisioning and k is the number of VM that are already deployed on the PM. Note that we set the queue size at each PM equal to the maximum number of VMs that can be deployed on a PM. Let ϕ_h be the rate at which a VM can be deployed on a PM at hot pool and μ be the service rate of each VM. So, the total service rate for each PM is the product of number of running VMs by μ. State $(0,0,0)$ indicates that the PM is empty and there is no task either in the queue or in the instantiation unit. Depending on the size of arriving super-task, model transits to one of the states in $\{(0,1,0), (1,1,0), \cdots, (MSS-1,1,0)\}$, in which MSS is the maximum possible size of super-tasks. Since all tasks within a super-task are to be provisioned at the same PM, the maximum number of VMs on a PM must be at least equal to MSS. The arrival rate to each PM in the hot pool is given by:

$$\lambda_h = \frac{\lambda_{st}(1-BP_q)}{N_h} \quad (9)$$

in which N_h is the number of PMs in the hot pool. Note that BP_q, used in (2), is obtained from RASM. In Fig. 4, $\{\lambda_i, i = 1..MSS\}$ is given by $\lambda_i = \lambda_h p_i$; here p_i is the probability of having a super-task of size i. In this work, the size of super-task can be generally distributed, but we use truncated geometric distribution for numerical experiment. The state transition in VMPSM can occur due to super-task arrival, task instantiation or service completion. From state $(0,0,0)$, system can move to state $(1,1,0)$ with rate λ_2 (i.e., super-task with size 2). From $(1,1,0)$, system can transit to $(0,1,1)$ with rate ϕ_h (i.e., instantiation rate) or upon arriving a super-task with size k, moves to state $(k+1,1,0)$. From $(0,1,1)$, system can move to $(0,0,2)$ with rate ϕ_h, transits to $(0,1,0)$ with rate μ (i.e., service completion rate), or again upon arriving a super-task with size k, system can move to state $(k,1,1)$ with rate λ_k. A PM can not accept a super-task, if there is no enough room to accommodate all tasks within super-task. Suppose that $\pi^h_{(i,j,k)}$ is the steady-state probability for the hot PM model (Fig. 4) to be in the state (i,j,k). Using steady-state probabilities, we can obtain the probability that at least one PM in hot pool can accept the super-task for provisioning. First we need to compute the probability that a hot PM cannot admit a super-task for provisioning (P^h_{na}). Let m be the maximum possible number of VMs in a PM and F_h be the free capacity at each state:

$$F_h(i,j,k) = m - (i+j+k)$$

P^h_{na} is then given by:

$$P^h_{na} = \sum_{x \in \xi} \sum_{t=F_h+1}^{MSS} \pi^h_x p_t \quad (10)$$

where p_t is the probability of having a super-task of size t and ξ is a subset of VMPSM states with the following definition:

$$\xi = \{(i,j,k) |\ F_h(i,j,k) < MSS\}$$

Therefore, probability of success provisioning (P_h) in the hot pool can be obtained as

$$P_h = 1 - (P^h_{na})^{N_h} \quad (11)$$

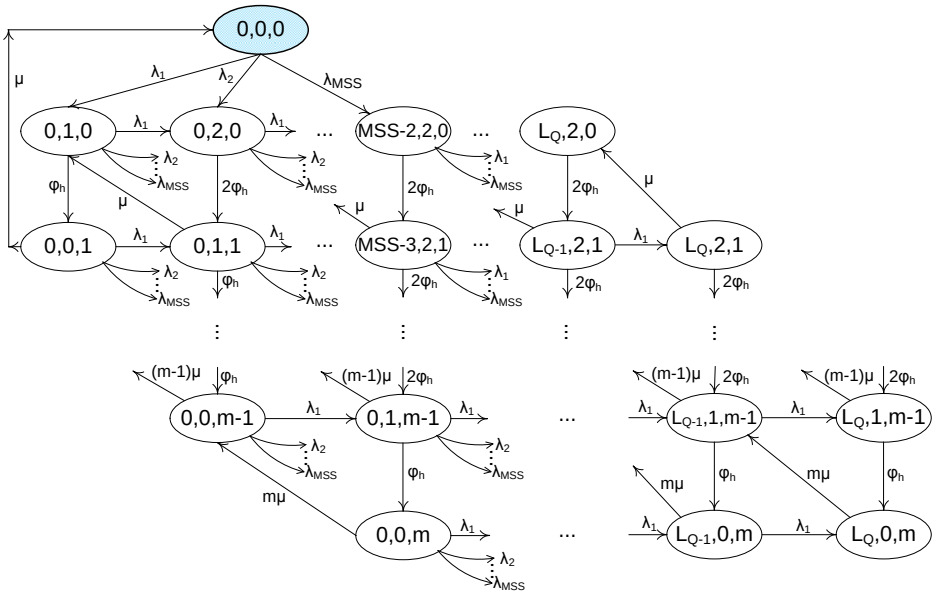


Fig. 4. Virtual machine provisioning sub-model for a PM in the hot pool.

Note that P_h is used as an input parameter in the resource allocation sub-model (Fig. 2). The provisioning model for warm PM is the same with the one for hot PM, though, there are some differences in parameters:

(a) The arrival rate to each PM in warm pool is

$$\lambda_w = \frac{\lambda_{st}(1 - BP_q)(1 - P_h)}{N_w} \tag{12}$$

where N_w is the number of PMs in warm pool.

(b) Every PM in warm pool requires extra time to be ready (hot) for first instantiation. This time is assumed to be exponentially distributed with mean value of γ_w.

Instantiation rate in warm pool is the same as in hot pool (ϕ_h). Like VMPSM for a hot PM (Fig. 4), the model for a warm PM is solved and the steady-states probabilities ($\pi^w_{(i,j,k)}$), are obtained. The success probability for provisioning a super-task in warm pool is:

$$P_w = 1 - (P^w_{na})^{N_w} \tag{13}$$

A PM in the cold pool can be modeled as in the warm and hot pool but with different arrival rate and start up time. The effective arrival rate at each PM in the cold pool is given by:

$$\lambda_c = \frac{\lambda_{st}(1 - BP_q)(1 - P_h)(1 - P_w)}{N_c} \tag{14}$$

in which N_c is the number of PMs in the cold pool. A cold PM (off machine), requires longer time than a warm PM to be ready (hot) for the first instantiation. We assume that the start up time is also exponentially distributed with mean value of γ_c. The success probability for provisioning a super-task in the cold pool is given by:

$$P_c = 1 - (P^c_{na})^{N_c} \tag{15}$$

From VMPSM, we can also obtain the mean waiting time at PM queue ($\overline{wt_{PM}}$) and mean provisioning time ($\overline{pt}$) by using the same approach as the one that led to Eq. (6). As a result, the total delay before starting of actual service time, is given by:

$$\overline{td} = \overline{wt} + \overline{lut} + \overline{wt_{PM}} + \overline{pt} \tag{16}$$

Note that all success probabilities (i.e., P_h, P_w and P_c) are dependent on the arrangement (i.e., number of hot, warm and cold PMs) of pools.

4.3 Interaction among sub-models

The interactions among sub-models is depicted in Fig. 5. VM provisioning sub-models (VMPSM) compute the steady state probabilities (P_h, P_w and P_c) that at least one PM in a pool (hot, warm and cold, respectively) can accept a super-task for provisioning. These probabilities are used as input parameters to the resource allocation sub-model (RASM). Hot PM sub-model (VMPSM_hot) computes P_h which is the input parameter for both warm and cold sub-models and warm PM sub-model (VMPSM_warm) computes P_w which is the input parameter for the cold sub-model (VMPSM_cold). The resource allocation sub-model compute the blocking probability, BP_q, which is the input parameter to VM provisioning sub-models. As can be seen, there is an inter-dependency among sub-models. This cyclic dependency is resolved via fixed-point iterative method [37] using a modified version of successive substitution approach (Algorithm 1).

5 Numerical validation

The resulting sub-models have been implemented and solved using Maple 15 from Maplesoft, Inc. [38]. The successive substitution method (Algorithm 1) is continued until the difference between the previous and current value of blocking probability in global queue (i.e., BP_q) is less

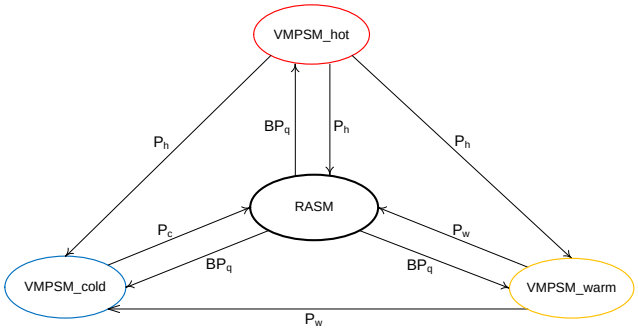


Fig. 5. Interaction diagram among sub-models.

than 10^{-6}. Usually the integrated model converges to the solution in less than 15 iterations. To validate the analytical solution, we have also built a discrete event simulator of the cloud center using the Artifex engine [39].

Algorithm 1 Successive Substitution Method

Input: Initial success probabilities in pools: P_{h0}, P_{w0}, P_{c0}
Output: Blocking probability in Global Queue: BP_q

```
counter ← 0; max ← 30; diff ← 1
BP_q0 ← RASM (P_h0, P_w0, P_c0)
while diff ≥ 10^-6 do
  counter ← counter + 1
  P_h1 ← VMPSM_hot (BP_q0,P_h1)
  P_w1 ← VMPSM_warm (BP_q0,P_h1)
  P_c1 ← VMPSM_cold (BP_q0,P_h1,P_w1)
  BP_q1 ← RASM (P_h1, P_w1, P_c1)
  P_h0 ← P_h1; P_w0 ← P_w1; P_c0 ← P_c1
  diff ← |(BP_q1 − BP_q0)|
  BP_q0 ← BP_q1
  if counter = max then
    break
  end if
end while
if counter = max then
  return -1
else
  return BP_q0
end if
```

Under different configurations and parameter settings, we obtain two important performance metrics, namely, rejection probability of super-tasks (STs) and total response delay. Mean service time of each task within STs ranges form 20 to 140 minutes. However, it should be noted that, beyond a certain degree of virtualization (i.e., number of deployed VMs), the actual service times will increase due to the overhead required to run several VMs concurrently. To model this effect, we have used publicly available data about server performance [40]. The continuous line in Fig. 6 shows the VMmark performance for a family of advanced servers under varying degree of virtualization [41], including both computation and communication time; from this data, we have extracted the normalized response time, shown by circles, which has subsequently been used in our experiments. Mean service time represents both

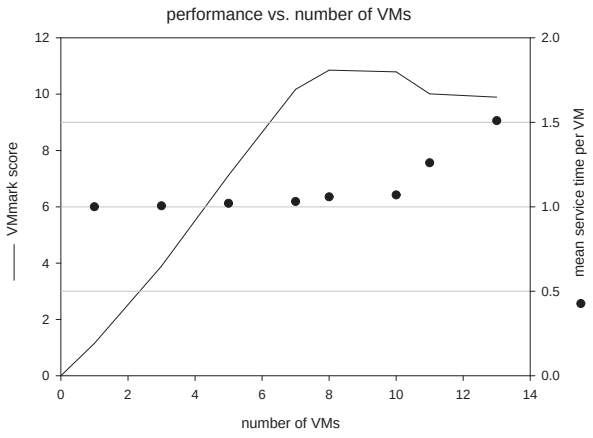


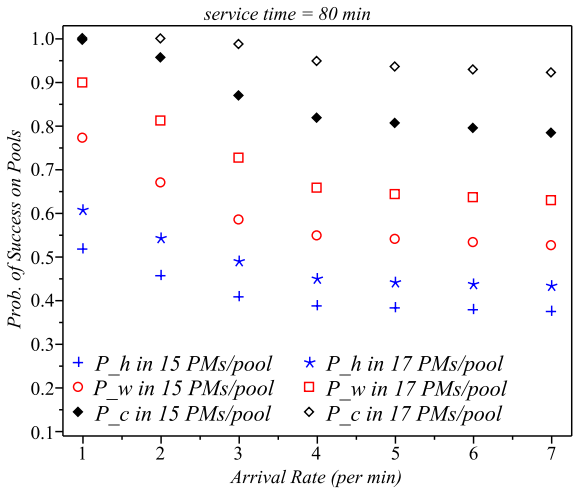
Fig. 6. VMmark results and normalized mean service time.

computation and communication time for a task within super-tasks. Identifying higher moments of service time requires more data or benchmarks from cloud providers [42].

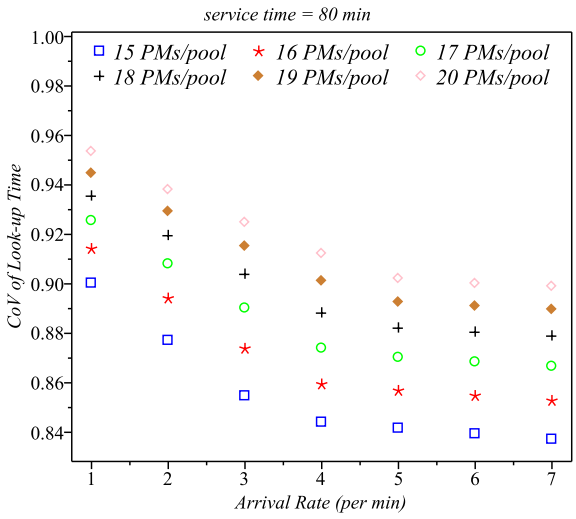
Arrival rate ranges from 1 to 7 STs per minute and global queue size is set to 50 STs. We assume 15 to 20 PMs in each pool and each PM can run up to 10 VMs simultaneously. The look-up time for finding a proper PM is assumed to be independent of the number PMs in the pool and the type of pool (i.e., hot, warm and cold). Look-up rate is set to 10 searches per minute. Mean preparing delay for a warm and cold PM to become a hot PM (i.e., be ready for first VM instantiation) are assumed to be 1 to 3 and 5 to 10 minutes, respectively. Mean time to deploy a VM on a hot PM is assumed to be between 1 and 10 minutes. First VM instantiation on a warm and cold PM are to be 5 to 20 and 15 to 30 minutes, respectively. After first VM instantiation on a warm or cold PM, it has already become a hot PM so that the next VM can be deployed just like a hot PM. Note that in all plots, analytical model and simulation results are shown by lines and symbols respectively.

First, we characterize the look-up time in pools by presenting the effect of arrival rate on the success probabilities for two configurations (15 and 17 PMs per pool). As can be seen in Fig. 7(a), the success probabilities decrease smoothly by increasing the arrival rate. Also, it can be noticed that the curves for P_h, P_w and P_c are equidistant which reveals a linear dependency among them with respect to offered load and pool capacity. Fig. 7(b) shows that by increasing the arrival rate, the CoV of look-up time is slightly below one which corresponds to exponential distribution. More specifically, under high traffic intensity, super-tasks have a lower chance to get provisioned in the hot pool so that RAM needs to check other pools more frequently. Under heavy load, distribution of look-up time will approach Coxian distribution with CoV between 0.84 and 0.90 while under low load it is close to exponential distribution. Overall, these results justify the

Poisson approximation of the arrival process to the PMs, as does the degree of match between analytical and simulation results in Figs. 8, 9 and 10.



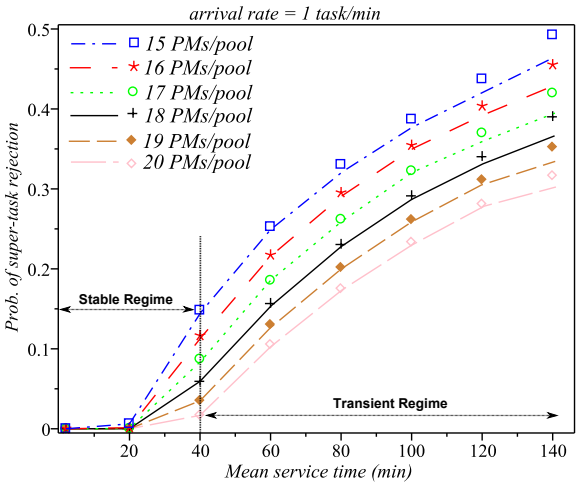
(a) Success probabilities on the pools.



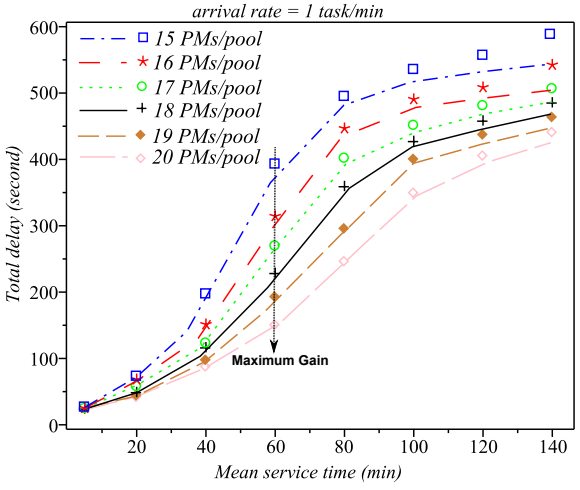
(b) CoV of look-up time on the pools.

Fig. 7. Look-up time characteristics.

We also analyze the effects of task service time on rejection probability and total delay imposed by the cloud center on STs before getting into service. In the first experiment, the results of which is shown in Fig. 8, STs have one task and PMs are permitted to run only one VM. Fig. 8(a) shows, at a fixed arrival rate (1 STs/min) and different numbers of PMs in each pool, increasing mean service time, raising the rejection probability linearly. Increasing the system capacity (i.e., the number of PMs in each pool) reduces the rejection probability. Also, it can be seen that for tasks shorter than 40 min, extending the capacity of system has negligible effect on rejection probability, while for tasks of longer duration (i.e., from 40 to 140 min) rejection probability drops almost linearly. The maximum gain is obtained at 140 minutes. Fig 8(b) shows that longer service times will result in longer imposed



(a) Task Rejection probability.

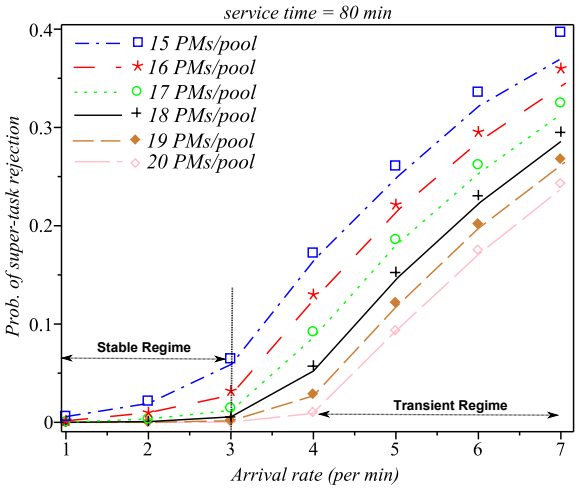


(b) Total delay for a super-task.

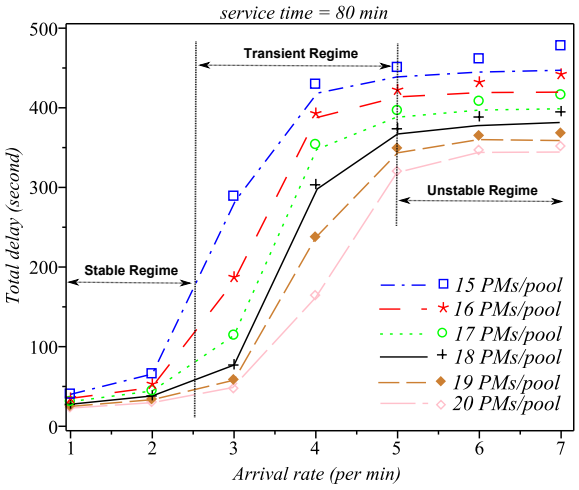
Fig. 8. Single VM deployed on each PM.

delays on STs. It can be seen that for any capacity, there is a turning point beyond which total delay increases sharply. For instance when the capacity is 45 PMs (i.e., 15 PMs in each pool), the turning point is 20 minutes while for a capacity of 60 PMs (i.e., 20 PMs in each pool), the turning point is 40 minutes. Cloud providers should operate in the stable regime (i.e., before turning points). They can be alerted by such turning points to add more resources and avoid entering transient regime. In addition, it can be observed that for the given configuration, adding five PMs to each pool (i.e., comparing 15 PMs with 20 PMs at 60 min) can reduce the delay by up to a half.

We also calculate the two performance metrics under different arrival rates while fixing the average service time at 80 minutes (Fig. 9). One turning point can be noticed in rejection probability curves (Fig. 9(a)) at which rejection trend increases suddenly. Such points (e.g., Fig. 9(a), 4 STs/min for 20 PMs in each pool) may suggest the appropriate time to upgrade or inject more resources (i.e., PMs) so that prevent of an undesired rejection rate. In case

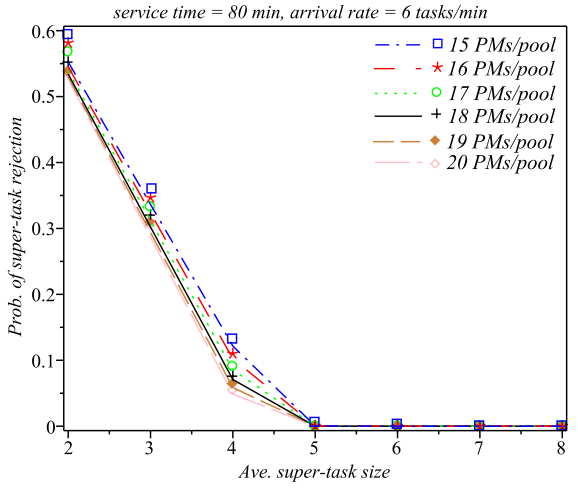


(a) Task Rejection probability.

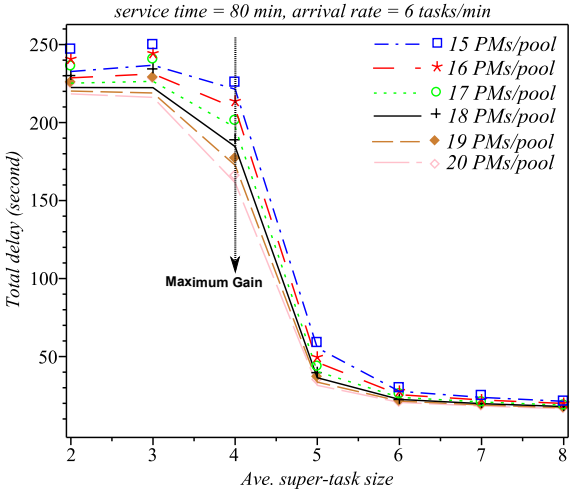


(b) Total delay for a super-task.

Fig. 9. Single VM on each PM.



(a) Task Rejection probability.



(b) Total delay for a super-task.

Fig. 10. Ten VMs are allowed on each PM.

of total delay, two turning points can be identified (Fig. 9(b)): after the first one , total delay grows sharply while at the second point increasing the arrival rate almost has no effect (i.e., unstable regime). The transient regime, the range between two turning points, is attributed to the size of global queue. The longer the global queue is, the larger the transient range is going to be which is expected.

In the last experiment, we examine the effect of super-task size on task rejection probability and total delay. Each PM can run up to ten VMs simultaneously and a super-task may request up to all of the available VMs in a PM. Note that the aggregate arrival rate is set to 6 tasks/hr, however the effective arrival rate of super-tasks is various for different super-task size. For instance, if the average super-task size was 3, then the effective arrival rate of super-tasks would be 2 STs/hr. As can be seen by increasing the size of super-tasks the rejection probability and total delay reduce sharply. The bigger super-task size will result in the lower number of arrivals as well as lower deviation of super-task size. In other words, in case of large super-tasks, they are more likely to be almost in the same size which improves the utilization of the cloud center. Notice that, in this work we adopted total acceptance policy for super-task servicing. Maximum gain is occurred at arrival rate of 4 STs/min for different pool arrangements. Also, it can be seen that the results from analytical model and the simulation are in a good agreement.

6 Conclusions

In this paper we presented a performance model suitable for analyzing the service quality of large sized IaaS clouds, using interacting stochastic models. We examined the effects of various parameters including ST arrival rate, task service time, the virtualization degree, and ST size on task rejection probability and total response delay. The stable, transient and unstable regimes of operation for given configurations have been identified so that capacity planning is going to be a less challenging task for cloud providers. Validation of analytical results through extensive simulation has shown

that our analytical model is sufficiently detailed to capture all realistic aspects of resource allocation process, instance creation and instantiation delays of a modern cloud center, while maintaining a good tradeoff between accuracy and tractability. Consequently, cloud providers can obtain a reliable estimation of response time and blocking probability that will result in avoidance of SLA violation.

REFERENCES

[1] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, 2009.

[2] An amazon.com company, "Amazon Elastic Compute Cloud, Amazon EC2," Website, 2012, http://aws.amazon.com/ec2.

[3] IBM, "IBM Cloud Computing," Website, 2012, http://www.ibm.com/ibm/cloud/.

[4] GOGRID, "GoGrid Cloud," Website, July 2012, http://www.gogrid.com.

[5] NephoScale, "The NephoScale Cloud Servers," Website, July 2012, http://www.nephoscale.com/nephoscale-cloud-servers.

[6] Rackspace, "The Rackspace Cloud," Website, July 2012, http://www.rackspace.com/cloud/.

[7] M. Martinello, M. Kaniche, and K. Kanoun, "Web service availability–impact of error recovery and traffic model," *Reliability Engineering System Safety*, vol. 89, no. 1, p. 616, 2005.

[8] N. Sato and K. S. Trivedi, "Stochastic modeling of composite web services for closed-form analysis of their performance and reliability bottlenecks," *Service Oriented Computing ICSOC 2007*, pp. 107–118, 2007.

[9] S. Ferretti, V. Ghini, F. Panzieri, M. Pellegrini, and E. Turrini, "QoS-aware clouds," in *2010 IEEE 3rd International Conference on Cloud Computing (CLOUD)*, Jul. 2010, pp. 321–328.

[10] H. Khazaei, J. Mišić, and V. B. Mišić, "Performance analysis of cloud computing centers using $M/G/m/m+r$ queueing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 5, p. 1, 2012.

[11] F. Longo, R. Ghosh, V. K. Naik, and K. S. Trivedi, "A scalable availability model for infrastructure-as-a-service cloud," *Dependable Systems and Networks, International Conference on*, pp. 335–346, 2011.

[12] 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 *Proceedings of the 2010 IEEE 16th Pacific Rim International Symposium on Dependable Computing*, ser. PRDC '10, Washington, DC, USA, 2010, pp. 125–132.

[13] J. Fu, W. Hao, M. Tu, B. Ma, J. Baldwin, and F. Bastani, "Virtual services in cloud computing," in *IEEE 2010 6th World Congress on Services*, Miami, FL, 2010, pp. 467–472.

[14] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 6, pp. 931–945, Jun. 2011.

[15] L. Youseff, R. Wolski, B. Gorda, and R. Krintz, "Paravirtualization for hpc systems," in *In Proc. Workshop on Xen in High-Performance Cluster and Grid Computing*. Springer, 2006, pp. 474–486.

[16] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: The montage example," *2008 SC International Conference for High Performance Computing Networking Storage and Analysis*, vol. C, no. Nov., pp. 1–12, 2008.

[17] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon S3 for science grids: a viable solution?" *Proceedings of the 2008 international workshop on Dataaware distributed computing DADC 08*, pp. 55–64, 2008.

[18] E. Walker, "Benchmarking Amazon EC2 for high-performance scientific computing," *LOGIN*, vol. 33, no. 5, pp. 18–23, 2008.

[19] L. Wang, J. Zhan, W. Shi, Y. Liang, and L. Yuan, "In cloud, do mtc or htc service providers benefit from the economies of scale?" *Proceedings of the 2nd Workshop on ManyTask Computing on Grids and Supercomputers MTAGS 09*, vol. 2, pp. 1–10, 2010.

[20] S. Saini, D. Talcott, D. Jespersen, J. Djomehri, H. Jin, and R. Biswas, "Scientific application-based performance comparison of sgi altix 4700, ibm power5+, and sgi ice 8200 supercomputers," *2008 SC International Conference for High Performance Computing Networking Storage and Analysis*, pp. 1–12, 2008.

[21] S. Alam, R. Barrett, M. Bast, M. R. Fahey, J. Kuehn, C. McCurdy, J. Rogers, P. Roth, R. Sankaran, and J. S. Vetter, "Early evaluation of IBM BlueGene," *2008 SC International Conference for High Performance Computing Networking Storage and Analysis*, pp. 1–12, 2008.

[22] N. Yigitbasi, A. Iosup, D. Epema, and S. Ostermann, "C-meter: A framework for performance analysis of computing clouds," in *CCGRID09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009, pp. 472–477.

[23] K. Xiong and H. Perros, "Service performance and analysis in cloud computing," in *IEEE 2009 World Conference on Services*, Los Angeles, CA, 2009, pp. 693–700.

[24] B. Yang, F. Tan, Y. Dai, and S. Guo, "Performance evaluation of cloud service considering fault recovery," in *First Int'l Conference on Cloud Computing (CloudCom) 2009*, Beijing, China, Dec. 2009, pp. 571–576.

[25] H. Khazaei, J. Mišić, and V. B. Mišić, "Modeling of cloud computing centers using $M/G/m$ queues," in *The First International Workshop on Data Center Performance*, Minneapolis, MN, Mar. 2011.

[26] ——, "Performance analysis of cloud centers under burst arrivals and total rejection policy," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, Dec. 2011, pp. 1–6.

[27] R. Ghosh, F. Longo, V. K. Naik, and K. S. Trivedi, "Quantifying resiliency of IaaS cloud," *Reliable Distributed Systems, IEEE Symposium on*, vol. 0, pp. 343–347, 2010.

[28] H. Qian, D. Medhi, and K. S. Trivedi, "A hierarchical model to evaluate quality of experience of online services hosted by cloud computing," in *Integrated Network Management (IM), IFIP/IEEE International Symposium on*, May. 2011, pp. 105–112.

[29] S. Kieffer, W. Spencer, A. Schmidt, and S. Lyszyk, "Planning a Data Center," white paper, February 2003, http://www.nsai.net/White_Paper-Planning_A_Data_Center.pdf.

[30] H. Takagi, *Queueing Analysis*. Amsterdam, The Netherlands: North-Holland, 1993, vol. 2: Finite Systems.

[31] G. Grimmett and D. Stirzaker, *Probability and Random Processes*, 3rd ed. Oxford University Press, Jul 2010.

[32] D. P. Heyman and M. J. Sobel, *Stochastic Models in Operations Research*. Dover, 2004, vol. 1.

[33] H. Takagi, *Queueing Analysis*. Amsterdam, The Netherlands: North-Holland, 1991, vol. 1: Vacation and Priority Systems.

[34] L. Kleinrock, *Queueing Systems, Volume 1, Theory*. Wiley-Interscience, 1975.

[35] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, 2nd ed. Wiley, 2001.

[36] D. G. Feitelson, *Workload Modeling for Computer Systems Performance Evaluation*. Jerusalem, Israel, June 2012, Version 0.36.

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

[38] Maplesoft, Inc., "Maple 15," Website, Mar. 2011, http://www.maplesoft.com.

[39] RSoft Design, *Artifex v.4.4.2*. San Jose, CA: RSoft Design Group, Inc., 2003.

[40] A. Baig, "UniCloud Virtualization Benchmark Report," white paper, March 2010, http://www.oracle.com/us/technologies/linux/intel-univa-virtualization-400164.pdf.

[41] VMware, Inc., "VMware VMmark 2.0 benchmark results," Website, May. 2012, http://www.vmware.com/a/vmmark/.

[42] J. Cao, W. Zhang, and W. Tan, "Dynamic control of data streaming and processing in a virtualized environment," *Automation Science and Engineering, IEEE Transactions on*, vol. 9, no. 2, pp. 365–376, April 2012.

Hamzeh Khazaei received his B.S. (2004) and M.S. (2008) degrees in computer science from Amirkabir University of Technology (Tehran Polytechnic), Iran. Currently he is a Ph.D candidate in Department of Computer Science, University of Manitoba, Canada. His research interests include cloud computing modeling and performance evaluation, stochastic processes and queuing theory.

Jelena Mišić (M91, SM08) received her PhD degree in Computer Engineering from University of Belgrade, Serbia, in 1993. She is currently Professor of Computer Science at the Ryerson University, Toronto, Canada. Her current research interests include wireless networks and security in wireless networks. She is a member of IEEE and ACM.

Vojislav B. Mišić received his PhD in Computer Science from University of Belgrade, Serbia, in 1993. He is currently Professor of Computer Science at Ryerson University in Toronto, Ontario, Canada. His research interests include wireless networks and systems and software engineering.