Effective Modeling Approach for IaaS Data Center Performance Analysis under Heterogeneous Workload

Xiaolin Chang, Ruofan Xia, Jogesh K. Muppala, Kishor S. Trivedi, Jiqiang Liu

Abstract—Heterogeneity prevails not only among physical machines but also among workloads in real IaaS Cloud data centers (CDCs). The heterogeneity makes performance modeling of large and complex IaaS CDCs even more challenging. This paper considers the scenario where the number of virtual CPUs requested by each customer job may be different. We propose a hierarchical stochastic modeling approach applicable to IaaS CDC performance analysis under such a heterogeneous workload. Numerical results obtained from the proposed analytic model are verified through discrete-event simulations under various system parameter settings.

Index Terms—Markov chain; IaaS; Cloud computing; Performance analysis; Heterogeneous

1 Introduction

INFRASTRUCTURE as a service (IaaS) cloud services are being commercially offered, in which IaaS Cloud service providers (CSPs) allocate virtual computing resources to customers in the form of virtual machines (VMs) deployed on Physical Machines (PMs). The worldwide cloud service market is to see a fast growth of IaaS with a five-year Compound Annual Growth Rate of 31.7% [1]. With the ever-increasing dependence on cloud services from customers, an important issue facing CSPs is the evaluation of cloud service performance so as to lower various management costs without violating Quality of Service (QoS) requirements, which are commonly specified via Service Level Agreements (SLAs). Analytic modeling is an effective performance evaluation approach. Stochastic models (see [7]-[10] and the references therein) have been proposed for evaluating the IaaS CDC performance under the assumption that all VMs are homogeneous. In actual CDCs, the capacities and/or types of physical and/or virtual resources requested by customers may be different [2], leading to VM heterogeneity.

Fig. 1 describes the process of handling a customer request in a typical IaaS CDC, including a PM Search&Allocation server (PMSA) and a pool of PMs. PMSA is responsible for locating a PM to allocate to a customer job. A PM can run multiple VMs concurrently and each job is for a VM. The main objective of this paper is to explore an effective analytic modeling approach for performance analysis of the CDC described in Fig.1 under heterogeneous workload. We consider the simple workload heterogeneity, i.e. only the number of virtual CPUs (vCPUs) requested by each customer job may be different. The type and amount of other resources requested by each job are the same. Thus, VMs considered in this paper are heterogeneous in terms of their allocated vCPUs. We use *VM size* to denote the number of vCPUs occupied by a VM in the rest of the paper.

The main contributions are summarized as follows:

1) A monolithic model for a PM under heterogeneous workload is developed. To our knowledge, we are the first to allow the number of vCPUs requested by a customer job to follow a general distribution. The models proposed in [16] and [17] considered heterogeneous workload but they both assumed the uniform distribution of *VM size*. The workload analysis of some real CDCs [18] indicated the incorrectness of this assumption. Our literature investigation also shows that we are the first to consider the PM scenario where more than one heterogeneous job in the PM waiting queue goes into service concurrently after a job departure. Both [16] and [17] considered the scenario where only one job enters into service after a job departure. Our PM modeling method integrates three kinds of information into determining the state transitions due to a job departure and into the calculation of state transition rates. Namely, (i) the number of vCPUs released by a departing job; (ii) the number of jobs starting VM service after a job departure and (iii) the vCPU number requested by a job going into service.
2) A novel hierarchical modeling approach is proposed for the CDC in Fig.1. Different from the existing interacting models for CDCs proposed in [7] and [10], we investigate a more realistic scenario where PMSA assigns an

- *Xiaolin Chang and Jiqiang Liu are with Beijing Jiaotong University, P.R.China. E-mail: {xlchang, jqliu} @ bjtu.edu.cn.*
- *Ruofan Xia and Kishor S. Trivedi are with Duke University, Durham, NC 27708, USA. Email: {ruofan.xia,kst}@duke.edu.*
- *Jogesh K. Muppala is with Hong Kong University of Science and Technology. E-mail: muppala@cse.ust.hk.*

xxxx-xxxx/0x/$xx.00 © 200x IEEE Published by the IEEE Computer Society

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2560158, IEEE Transactions on Cloud Computing

arriving job only to those PMs whose waiting queues are not full. Such PMs are denoted as *nonFULL* PMs in the rest of the paper. To the best of our knowledge, we are the first to model this scenario. An iterative algorithm is proposed to achieve the job request allocation only to *nonFULL* PMs in a pool. Note that an iterative-type algorithm was used in both [7] and [10] but for assigning jobs among three pools. When there is only one pool in the CDC, their iterative algorithm is not necessary.

3) Extensive numerical experiments and simulations are carried out to evaluate the accuracy of the PM monolithic model and the capability of the hierarchical model in capturing the CDC behavior. Note that the proposed hierarchical modeling approach solves only one model for all the PMs in the CDC and then aggregates the obtained results to approximate the CDC behavior. Therefore, some errors are introduced. The experiment results in Section 5 indicate that such error is insignificant and the proposed hierarchical modeling approach is effective in terms of accuracy and scalability.

The rest of the paper is organized as follows. In Section 2, we present related work. Section 3 describes the system and presents the hierarchical model. The details of the PM model are presented in Section 4. Experimental results are presented in Section 5. Section 6 describes how to extend the PMSA model described in Section 3 to the situation of a more complex (non-Poisson) job arrival process at CDC. Finally, we conclude this paper in Section 7.

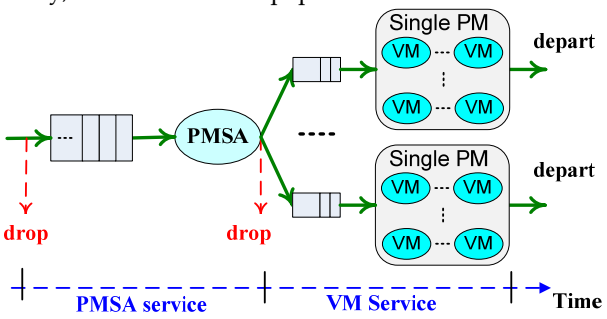


Fig.1 Job processing steps in a typical IaaS CDC system

2 Related Work

The increasing prominence of IaaS clouds has prompted research efforts into their performance analysis. Analytic modeling can be applied to effectively evaluate the CDC performance. State-space analytic modeling approaches (see [7], [8], [9] and the references therein) have been proposed to evaluate the performance of IaaS CDCs by assuming all VMs/servers are homogeneous. But heterogeneity prevails not only among physical machines but also among workloads in real IaaS CDCs:

- *Physical Machine Heterogeneity*. Production CDCs are often composed of multiple types of physical machines. These machines have heterogeneous processing capabilities, different hardware features, and varying memory and disk sizes. Even if production CDC started with homogeneous PMs, PMs may be upgraded or replaced and the PM pool typically becomes heterogeneous over time.
- *Workload Heterogeneity*. Production IaaS CDCs typically receive vast numbers of heterogeneous requests with diverse resource demands, durations, priorities, and performance objectives [2].

These heterogeneities result in IaaS cloud service heterogeneity. Ghosh *et al.* [7] discussed a scalable modeling approach for the scenario where a PM may be in an active state, in a sleep state or in an idle state. This paper assumes that all the PMs are active. We leave the modeling for the pool with different state PMs for future work.

In production CDCs, IaaS CSPs have offered various VMs in order to meet the customers' diverse service requests. Amazon EC2, for example, currently offers at least twelve types of VM instances [3]. Azure offers at least five types [4] and Rackspace offers at least seven types [5]. Recently, Khazaei *et al.* [16] and Wang *et al.* [17] considered VM heterogeneity in terms of *VM size*. There are two major differences between their work and ours:

- *The number of waiting jobs that go into service after a job completes its service and departs from the system*. Khazaei and Wang assumed there are two phases in a PM: VM provisioning phase and VM service phase. In addition, they assumed that there was only one server in VM provisioning phase. Thus, at most one job in the PM waiting queue goes into VM provisioning phase when a job departs from the system. As in [8] and [15], this paper ignores VM provisioning delay, which can significantly reduce the state space size but brings challenges to PM modeling. The reason is that more than one job may go into VM service after a job departure. It should be noted that when the VM provisioning phase is considered and there is more than one server in this phase, it is possible for more than one job to begin VM provisioning when a job departs from the system. The state-transition rules proposed in this paper can be applied to this situation.
- *The distribution of the number of vCPUs released by a departing job*. Determining the exact number of the vCPUs released by a departing job is difficult, if not impossible. Khazaei and Wang proposed to first find out the probabilities of all the numbers of vCPUs (denoted as c_{rel}) released by a departing job, then apply the probabilities to obtain the possible state transition rates. This paper follows this approach. One major challenge in this approach is how to calculate the probability of releasing c_{rel} vCPUs. Khazaei assumed that c_{rel} follows the same distribution as the number of vCPUs (denoted as c_{req}) requested by an arriving job. Wang pointed out this as-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2560158, IEEE Transactions on Cloud Computing

sumption was wrong and derived the formula for computing the probability that a departing job releases c_{rel} vCPUs, but restricted to the situation where c_{req} is uniformly distributed. This paper allows c_{req} to follow a general distribution and derives the formula for computing the probability of releasing c_{rel} vCPUs. Note that although this formula is similar to that in [17], the proof is different.

Large scale is a fact of life in IaaS CDCs. Some of CDCs' unique characteristics, such as the large scale and many system parameters, make it difficult for a monolithic model to capture the CDC behavior. Recently, Ghosh *et al.* [7] and Khazaei *et al.*[10] proposed scalable modeling approaches, namely using interacting stochastic models, to overcome the complexity caused by the monolithic model for the whole large-scale CDC. This kind of approach decomposes the monolithic model into sub-models and a fixed-point based iterative algorithm was applied to obtain the final results. Ghosh and Khazaei both assumed that PMSA looked for a candidate PM for a job request in the whole pool. That is, each PM in a pool can be a candidate even when its waiting queue is full. If a job is assigned to a PM with a full queue, the job is dropped even if there exist some PMs whose waiting queues are not full. This paper considers a more realistic scenario, where PMSA picks up a PM candidate randomly from the *nonFULL* PMs. Thus, a job request is dropped only when the waiting queues of all PMs are full or PMSA waiting queue is full. Both Ghosh *et al.* [7] and Khazaei *et al.*[10] applied an iteration-based algorithm to allocate jobs among three pools. Thus, this iterative algorithm is not used when there is only one pool. But the iterative algorithm proposed in this paper aims to find out *nonFull* PMs, working in a pool.

Besides stochastic modeling, discrete-event simulation and measurement-based approaches are being adopted for performance evaluation of cloud services [7][8]. Furthermore, our paper is not only related to cloud resource management under heterogeneous workload [11] and/or heterogeneous hardware [12], but also related to various tools, such as in [22][23], which have been developed for monitoring and evaluating cloud service's behavior in order to improve cloud service performance. It should be noted that our modeling is complementary to these studies and can be used to make analysis of the reasons behind the results. Hardware and software are prone to failures in IaaS CDCs [13] and availability models [14] to capture this behavior have been proposed. Our performance model can be coupled with these availability models for performability analysis.

3 Hierarchical Model

This section first presents system description and then the hierarchical model.

3.1 System description

In an IaaS Cloud, when a tenant submits a request, a VM instance is first created according to the tenant's resource demand. Then a PM is determined for hosting the VM instance. When the VM instance is working, this tenant is granted access to this VM [35]. When the tenant finishes the work, the VM instance is released. This paper abstracts these actions into a two-step service, shown in Fig. 1. PMSA service is meant for creating VM instance and determining a PM. The remaining work is denoted as VM service.

TABLE 1 gives variable definitions be used in the rest of the paper. As in [7], [8], [12] and [19], the customer job arrivals at PMSA are assumed to follow a Poisson process and searching times are assumed to be exponentially distributed random variables with rate μ_C. An arriving customer job that finds the PMSA queue full will be rejected immediately. Once the job is admitted to the PMSA queue, it must wait until the PMSA processes it on a First-come, First-served (FCFS) basis. All PMs are homogeneous and each PM has a waiting buffer/queue, containing at most $(KC-1)$ jobs. If a PM with free buffer spaces is found, the job is put into the PM waiting queue for further service. If all PMs' waiting buffers are full, then the job is dropped. Thus, a customer job may be assigned to a PM, rejected because all PMs' waiting queues are full, or rejected due to insufficient PMSA buffer space. Each PM behaves independently from its peers within the pool, i.e., each *nonFULL* PM statistically handles $1/N_{nonFULL}$ of the jobs arriving at the pool.

We assume that the maximum number of vCPUs (denoted as m) from all co-located VMs at a PM is pre-defined. Fixed vCPU-to-core ratio is helpful for CSPs to provide guaranteed QoS to customers. Google Compute Engine (GCE) implements a single vCPU as a single hardware hyper-thread on the host core [6], i.e. the ratio in GCE is fixed. We also assume that the number of vCPUs requested by a new job follows a general distribution and the maximum number of vCPUs requested is denoted by MC, which is less than m. The inter-arrival times between successive arriving jobs are independent and exponentially distributed random variables with rate $1/\lambda$. If the waiting queue is full, a job arriving at this PM is rejected immediately. If the PM waiting queue is empty and the number of the free vCPUs is no less than c_{req} of the arriving job, the job goes into service right away. Otherwise, the job goes into the PM waiting queue. Jobs are processed FCFS on each PM. Servers (VMs) on a PM work independently. The service times follow an exponential distribution with rate $1/\mu$. When a busy VM/server finishes a job, it immediately releases the occupied vCPUs. It starts serving the next job waiting in the PM queue if the number of free vCPUs is not less than c_{req} of this job. ρ is defined as the vCPU utilization at a PM and $\rho = \left(\overline{P_{arrival}^{c_{req}}} \cdot \lambda\right) \Big/ (m \cdot \mu)$, where $\overline{P_{arrival}^{c_{req}}}$ is the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2560158, IEEE Transactions on Cloud Computing

expected value of $P_{arrival}^{c_{req}}$.

TABLE 1 DEFINITION OF VARIABLES

Notation	Description
KC	PMSA waiting buffer size plus one
λ_{PMSA}	Job arrival rate at PMSA
μ_C	Searching rate at PMSA
N	The number of total physical machines in the CDC
$N_{nonFULL}$	The number of physical machines, whose waiting queue is not full
P_{block}	The blocking probability at PMSA due to the PMSA waiting queue full
P_{drop}	The dropping probability at a PM due to PM queue full
i	The number of jobs in the PM waiting queue
j	The number of busy vCPUs in the PM
k	The number of busy VMs or jobs in service. $1 \le k \le j$
MC	The maximum number of vCPUs requested by a job
c_{req}	The number of vCPUs requested by an arriving job. $1 \le c_{req} \le MC$
c_{rel}	The number of vCPUs released by a departing job. $1 \le c_{rel} \le MC$
m	The maximum number of vCPUs from all co-located VMs in a PM
K	PM waiting buffer size
$P_{arrival}^{c_{req}}$	The probability that an arriving job requests c_{req} vCPUs. $\sum_{c_{req}=1}^{MC} P_{arrival}^{c_{req}} = 1$
$P_{leave}^{(j,k,c_{rel})}$	The probability that a departing job releases c_{rel} vCPUs. j and k, respectively, denote the number of busy vCPUs and the number of jobs in service on a PM. Here, $1 \le c_{rel} \le j-k+1$.

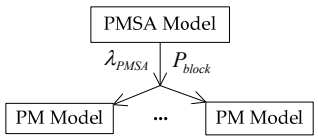


Fig.2 **Two-level hierarchical model**

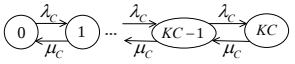


Fig.3 **PMSA model**

3.2 Hierarchical model

Our approach uses a two-level hierarchical model (Fig.2) for the performance analysis of the IaaS service. The top model is the PMSA model, capturing the PMSA service step in Fig.1. PMSA model processes jobs one by one without considering c_{req} of each customer job. Thus, it is an $M/M/1/K$ queuing model and Fig.3 gives the state transition diagram, in which a state is described by the number of jobs in the PMSA waiting queue plus any in service. The state transition rate vector in PMSA does not depend on any output of the bottom level model. But the PMSA output P_{block} is to be used as input to the bottom level model.

The bottom level model is the PM pool model, capturing the second step, denoted as VM service in Fig.1. We construct a three-dimensional CTMC model for each individual PM (denoted as PM model in the rest of the paper and described in Section 4) and then the overall PM pool model consists of these individual PM models. Each PM model requires as its input the job arrival rate at it, which is $1/N_{nonFULL}$ of the job arrival rate at the pool. **Algorithm 1** describes how to use the output of PMSA model and PM model to calculate $N_{nonFULL}$ and a set of steady-state performance measures (defined in Section 4.2) in the scenario, where a job is dropped only due to no *nonFULL* PMs if it is not rejected due to PMSA queue being full. For numerical experiments of the proposed hierarchical model, the PM model in **Algorithm 1** is iterated until the difference between the previous and current values of dropping probability (i.e., P_{drop}) on a PM is less than 10^{-15}.

Algorithm 1: Iterative algorithm for the steady state solutions of PM model

Input: λ_{PMSA}, P_{block} and ε

Output: P_{drop}

1. P_{drop} =0; P_{drop_new} =1.0; $N_{nonFULL} = N$
2. while $\left|P_{drop} - P_{drop\_new}\right| > \varepsilon$
3. $P_{drop\_new} = P_{drop}$
4. $\lambda = \dfrac{\lambda_{PMSA} \cdot (1 - P_{block})}{N_{nonFULL}}$
5. $P_{drop} \leftarrow$ solve PM model with λ ;
6. $N_{nonFULL} = N \cdot \left(1 - P_{drop}\right)$
7. endwhile

4 PM MODEL UNDER HETEROGENEOUS WORKLOAD

This section first describes the PM model in Section 4.1. Then Section 4.2 presents PM performance measure formulas.

4.1 State transition rules of PM model

Fig.4 shows PM model, in which a state is described by a 3-tuple (i, j, k). Definitions of *i*, *j* and *k* are given in TABLE 1. Two kinds of events lead to a state transition: *job arrival* and *job departure*. We describe the state transitions due to each event in Section 4.1.1 and 4.1.2, respectively.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2560158, IEEE Transactions on Cloud Computing

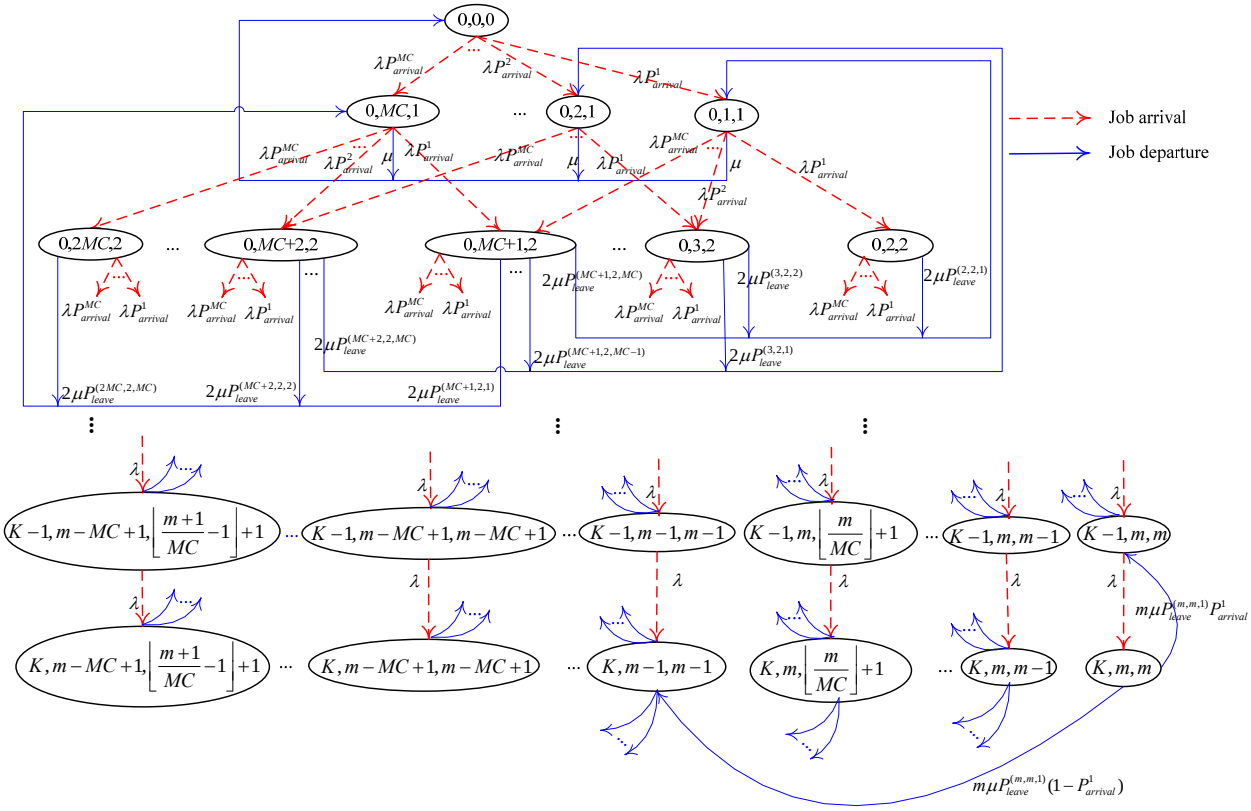


Fig.4 State transitions in a PM model

4.1.1 State transitions due to job arrival

All the state transitions upon a new job arrival can be divided into three cases: ***Case_A1***, ***Case_A2*** and ***Case_A3*** as follows according to the request demand and the amount of available resources.

Case_A1. $i=0$, $k \le j \le m$ and $c_{req} \in [1, \min\{MC, m-j\}]$

In this case, the arriving job can go into service immediately because c_{req} is no larger than the number of free vCPUs $(m-j)$. Equations (1) describes all the possible state transitions of this case.

$$(0, j, k) \to (0, j + c_{req}, k+1) \tag{1}$$

Here, $c_{req} \in [1, \min\{MC, m-j\}]$ and the corresponding transition rate is $\left(\lambda_C \cdot P_{arrival}^{c_{req}}\right)$.

Case_A2. $i=0$, $k \le j \le m$ and $c_{req} \in [\min\{MC, m-j\}+1, MC]$

In this case, although the waiting queue is empty when a job arrives, c_{req} of the job is larger than the number of free vCPUs $(m-j)$. Thus, the arriving job gets into the PM waiting queue. Equation (2) describes the state transitions and the transition rate is $\left(\lambda_C \cdot \sum_{c_{req}=\min\{MC, m-j\}+1}^{MC} \left(P_{arrival}^{c_{req}}\right)\right)$.

$$(0, j, k) \to (1, j, k) \tag{2}$$

Case_A3. $0 < i \le K$ and $k \le j \le m$

When a job arrives in this case, the waiting queue is not empty and then the arriving job gets into the waiting queue. Equation (3) describes the state transitions and the transition rate is λ_C.

$$(i, j, k) \to (i+1, j, k) \tag{3}$$

4.1.2 State transitions due to job departure

When a job departs the system, there may be more than one job in the waiting queue entering into service. Thus, there are the following two challenges in constructing state transition rules for a job departure event:

1) How to integrate the number of vCPUs, released by a departing job, into the calculation of state transition rates.

2) How to integrate the number of jobs starting VM service after a job departure and each requested vCPU number into the calculation of state transition rates.

We tackle each challenge by first computing the probability of each occurrence and then integrate the probabilities into the rate computation. Before describing the state transition rules, we first define some variables and then

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2560158, IEEE Transactions on Cloud Computing

discuss how to calculate $P_{leave}^{(j,k,c_{rel})}$. Given the number of busy vCPUs j and the number of jobs k in service, $\mathbb{N}[j,k]$ is defined to denote the number of ways (permutations) of k distinct jobs/VMs sharing j vCPUs. We assume that in the ℓ^{th} ($\ell \in [1,\mathbb{N}[j,k]]$) way, the 1st job, the 2nd job, … and the k^{th} job occupy $x_{\ell 1}$, $x_{\ell 2}$ …, and $x_{\ell k}$ vCPUs, respectively. Here, $\sum_{\gamma=1}^{k} x_{\ell\gamma} = j$ for any ℓ and $\max\{1, j-(k-1)\cdot MC\} \le x_{\ell\gamma} \le \min\{MC, j-k+1\}$. See an example of $j=5$, $k=3$ and $MC=4$. The possible way of the three jobs sharing 5 vCPUs can be $(1,1,3)$, $(1,3,1)$, $(1,2,2)$, $(2,1,2)$, $(2,2,1)$, or $(3,1,1)$. That is, $\mathbb{N}[5,3]=6$. ℓ=3 means $x_{\ell 1}=1$, $x_{\ell 2}=2$, and $x_{\ell 3}=2$.

Algorithm 2: Calculate $\Omega_{spec}[c_{rel}][j,k]$ and $\Omega_{all}[j,k]$

Input: *m*, *MC*
Output: Ω_{all} and Ω_{spec}
1. set all $\Omega_{spec}[c_{rel}][j,k]$ and all $\Omega_{all}[j,k]$ to zero.
2. for γ=1 to *MC* do
3. $\Omega_{all}[\gamma,1] = P_{arrival}^{\gamma}$
4. $\Omega_{spec}[\gamma][\gamma,1] = P_{arrival}^{\gamma}$
5. endfor
6. for ℓ_1 from 1 to m do
7. for ℓ_2 from 2 to m do
8. for ℓ_3 from $\max\{1, \ell_1-(\ell_2-1)\cdot MC\}$ to $\min\{MC, \ell_1-\ell_2+1\}$ do
9. $\Omega_{all}[\ell_1,\ell_2] = \Omega_{all}[\ell_1,\ell_2] + P_{arrival}^{\ell_3}\cdot\Omega_{all}[\ell_1-\ell_3,\ell_2-1]$;
10. for ℓ_4 from 1 to *MC* do
11. $\Omega_{spec}[\ell_4][\ell_1,\ell_2] = \Omega_{spec}[\ell_4][\ell_1,\ell_2] + P_{arrival}^{\ell_3}\cdot\Omega_{spec}[\ell_4][\ell_1-\ell_3,\ell_2-1]$
12. endfor
13. $\Omega_{spec}[\ell_3][\ell_1,\ell_2] = \Omega_{spec}[\ell_3][\ell_1,\ell_2] + P_{arrival}^{\ell_3}\cdot\Omega_{all}[\ell_1-\ell_3,\ell_2-1]$
14. endfor
15. endfor
16.endfor

Variables, $\Omega_{all}[j,k]$ and $\Omega_{spec}[c_{rel}][j,k]$, are defined in the context where just when a job (one of the k jobs) is completed, there are k jobs in service and each job can request at most *MC* vCPUs. $\Omega_{all}[j,k]$ denotes the probability that there are j busy vCPUs when one job is completed. Thus, $\Omega_{all}[j,k] = \sum_{\ell=1}^{\mathbb{N}[j,k]} P_{way}^{(j,k,\ell)}$. $\Omega_{spec}[c_{rel}][j,k]$ denotes the probability that a job occupying c_{rel} vCPUs is completed when there are there are j busy vCPUs. Thus,

$$\Omega_{spec}[c_{rel}][j,k] = \sum_{\ell=1}^{\mathbb{N}[j,k]} \left(P_{way}^{(j,k,\ell)} \cdot \xi[\ell][c_{rel}][j,k] \right)$$

Here $P_{way}^{(j,k,\ell)} = P_{arrival}^{x_{\ell 1}} \cdot P_{arrival}^{x_{\ell 2}} \cdot \cdots \cdot P_{arrival}^{x_{\ell k}}$ and $\xi[\ell][c_{rel}][j,k]$ denotes the number of jobs, each of which occupies c_{rel} vCPUs in the ℓ^{th} way. **Proposition 1** provides the formula for computing $P_{leave}^{(j,k,c_{rel})}$. The proof of **Proposition 1** is provided in the Appendix A of the supplementary file.

Proposition 1 Given the number of busy vCPUs j and the number of jobs k in service, the probability that a departing job releases c_{rel} vCPUs is

$$P_{leave}^{(j,k,c_{rel})} = \frac{\Omega_{spec}[c_{rel}][j,k]}{k\cdot\Omega_{all}[j,k]} \tag{4}$$

Equation (4) indicates that c_{rel} does not follow the distribution of c_{req}. State transitions due to a job departure depend on not only c_{rel} of the departing job, but also the number of jobs in VM service and the busy vCPU number. Given *m* and *MC*, **Algorithm 2** calculates all $\Omega_{spec}[c_{rel}][j,k]$ and $\Omega_{all}[j,k]$. That is, all $P_{leave}^{(j,k,c_{rel})}$ can be calculated by **Algorithm 2** with the knowledge of *m* and *MC*. The correctness of **Algorithm 2** is proved in **Proposition 2** of Appendix A in the supplementary file.

We now describe state transition rules upon a job departure in the following three cases: ***Case_D1***, ***Case_D2***, and ***Case_D3***.

Case_D1. $i=0$ and $k \le j \le \min\{MC\cdot k, m\}$

Equation (5) describes the possible state transitions and the transition rate is $k\cdot\mu\cdot P_{leave}^{(j,k,c_{rel})}$. Here, $c_{rel} \in \left[\max\{1, j-(k-1)\cdot MC\}, \min\{MC, j-k+1\}\right]$.

$$(0,j,k) \to (0, j-c_{rel}, k-1) \tag{5}$$

Case_D2. $i=1$ and $1 \le k \le j \le \min\{MC\cdot k, m\}$

The states reachable from state $(1,j,k)$ can be divided into two categories. The first are those where the waiting job cannot go into service, described in Equations (6) for $\forall c_{rel} \in \left[\max\{1, j-(k-1)\cdot MC\}, \min\{j-k+1, MC\}\right]$ and $\forall c_{req} \in \left[\min\{m-j+c_{rel}, MC\}+1, MC\right]$.

$$(1,j,k) \to (1, j-c_{rel}, k-1) \tag{6}$$

Here, c_{req} is the number of vCPUs requested by the waiting job. The corresponding transition rate denoted as $rate\_D21$ is $k\cdot\mu\cdot P_{leave}^{(j,k,c_{rel})} \cdot \left(\frac{\sum_{\gamma=\min\{m-j+c_{rel}, MC\}+1}^{MC} P_{arrival}^{\gamma}}{\sum_{\gamma=m-j+1}^{MC} P_{arrival}^{\gamma}} \right)$. The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2560158, IEEE Transactions on Cloud Computing

second are those where the waiting job enters into service, given by Equation (7) for $\forall c_{rel} \in [\max\{1, j-(k-1)\cdot MC\}, \min\{MC, j-k+1\}]$ and $\forall c_{req} \in [m-j+1, \min\{m-j+c_{rel}, MC\}]$.

$$(1, j, k) \to (0, j - c_{rel} + c_{req}, k) \quad (7)$$

The transition rate denoted as $rate\_D22$ is $k \cdot \mu \cdot P_{leave}^{(j,k,c_{rel})} \cdot \left(P_{arrival}^{c_{req}} \Big/ \sum_{\gamma=m-j+1}^{MC} P_{arrival}^{\gamma} \right)$. Now we explain them in detail. Before a job departure, the number of the available vCPUs is $(m-j)$. Thus, the number of vCPUs requested by the waiting job is in $[m-j+1, MC]$. Given j and k, the probability of this waiting job requesting c_{req} is $\left(P_{arrival}^{c_{req}} \Big/ \left(\sum_{\gamma=m-j+1}^{MC} P_{arrival}^{\gamma} \right) \right)$. After a job departure, there are $(m-j+c_{rel})$ free vCPUs. With these analyses, we now discuss the possible states which may be reached from state $(1, j, k)$ and the corresponding transition rates for each c_{rel} in the following three situations:

(i) $m-j+c_{rel} < MC$ and $c_{req} \in [m-j+c_{rel}+1, MC]$. In this case, the first waiting job in the queue cannot go into service after a job departure because there are no enough available vCPUs. Hence, given c_{rel}, j and k, the probability that the waiting job cannot be in service is $\left(\sum_{\gamma=\min\{m-j+c_{rel}, MC\}+1}^{MC} P_{arrival}^{\gamma} \Big/ \sum_{\gamma=m-j+1}^{MC} P_{arrival}^{\gamma} \right)$ and then the state transition rate from state $(1, j, k)$ to state $(1, j-c_{req}+c_{rel}, k)$ is $rate\_D21$. This is the state transition rate formula for Equation (6).

(ii) $m-j+c_{rel} < MC$ and $c_{req} \in [m-j+1, m-j+c_{rel}]$. In this case, the waiting job in the queue can go into service after a job departure. Thus, the state transition rate from state $(1, j, k)$ to state $(0, j-c_{rel}+c_{req}, k)$ is $rate\_D22$ for $\forall c_{req} \in [m-j+1, m-j+c_{rel}]$. Note that $[m-j+1, m-j+c_{rel}]$ can be re-written as $[m-j+1, \min\{m-j+c_{rel}, MC\}]$.

(iii) $m-j+c_{rel} \geq MC$. The waiting job definitely goes into service after a job departure. In addition, $[m-j+1, \min\{m-j+c_{rel}, MC\}]$ is equal to $[m-j+1, MC]$. Thus, state $(1, j, k)$ would move to state $(0, j-c_{rel}+c_{req}, k)$ with rate $rate\_D22$ for each $c_{req} \in [m-j+1, \min\{m-j+c_{rel}, MC\}]$.

Combining (ii) and (iii), we obtain the state transition rate formula for Equation (7).

Case_D3. $i > 1$ and $1 \leq k \leq j \leq m$

As in **Case_*D2***, the number of vCPUs requested by the first waiting job is in $[m-j+1, \min\{m-j+c_{rel}, MC\}]$. Define x as the number of waiting jobs, which go into service after a job departure. The states reachable from state (i, j, k) can be divided into the following four categories according to x:

(i) The first waiting job cannot be in service after a job departure.

In this situation, $x = 0$. Equations (8) describes all the possible state transitions.

$$(i, j, k) \to (i, j - c_{rel}, k-1) \quad (8)$$

The transitions rate is same as $rate\_D21$ and $c_{rel} \in [\max\{1, j-(k-1)\cdot MC\}, \min\{MC, j-k+1\}]$. The discussions of the formula of **Case_*D2*(i)** can be applied to explain Equation (8).

(ii) Only the first waiting job can go into service after a job departure.

In this situation, $x = 1$. Equation (9) describes state transition for $\forall c_{rel} \in [\max\{1, j-(k-1)\cdot MC\}, \min\{MC, j-k+1\}]$ and $\forall c_{req} \in [m-j+1, \min\{m-j+c_{rel}, MC\}]$.

$$(i, j, k) \to (i-1, j - c_{rel} + c_{req}, k) \quad (9)$$

Here, c_{req} denotes the number of vCPUs requested by the first waiting job and the transition rate denoted as $rate\_D32$ is as follows

$$k \cdot \mu \cdot P_{leave}^{(j,k,c_{rel})} \cdot \left(P_{arrival}^{c_{req}} \Big/ \sum_{\gamma=m-j+1}^{MC} P_{arrival}^{\gamma} \right) \cdot \left(\sum_{\gamma=m-j+c_{rel}-c_{req}+1}^{MC} P_{arrival}^{\gamma} \right)$$

The discussions in ***Case_D2*(ii, iii)** can be applied to explain $k \cdot \mu \cdot P_{leave}^{(j,k,c_{rel})} \cdot \left(P_{arrival}^{c_{req}} \Big/ \sum_{\gamma=m-j+1}^{MC} P_{arrival}^{\gamma} \right)$ in $rate\_D32$. Let's see $\left(\sum_{\gamma=m-j+c_{rel}-c_{req}+1}^{MC} P_{arrival}^{\gamma} \right)$ in $rate\_D32$. The number of vCPUs requested by the second waiting job in the PM queue can be from 1 to *MC*. After the first waiting job enters service, the number of available vCPUs is $(m-j+c_{rel}-c_{req})$. Thus, the probability that the second job cannot enter service is $\left(\sum_{\gamma=m-j+c_{rel}-c_{req}+1}^{MC} P_{arrival}^{\gamma} \right)$. Then we get the transition rate formula for Equation (9).

(iii) All the waiting jobs can go into service after a job departure.

In this situation, $x = i \geq 2$. Equation (10) gives the possible state transitions.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2560158, IEEE Transactions on Cloud Computing

$$(i,j,k) \to (0, j - c_{rel} + c_{avail}, k + x - 1) \tag{10}$$

Here $c_{rel} \in [\max\{1, j-(k-1)\cdot MC\}, \min\{MC, j-k+1\}]$. c_{avail} denotes the number of free vCPUs after a departing job releases c_{rel} vCPUs and $c_{avail} \in [m-j+2, m-j+c_{rel}]$. The transition rate denoted as $rate\_D33$ is as follows:

$$k \cdot \mu \cdot P_{leave}^{(j,k,c_{rel})} \cdot \left(\sum_{c_{req}=m-j+1}^{\min\{MC, c_{avail}-x+1\}} \Omega_{all}[c_{avail}-c_{req}, x-1] \cdot P_{arrival}^{c_{req}} \Big/ \sum_{\gamma=m-j+1}^{MC} P_{arrival}^{\gamma} \right)$$

Some explanations about the rate is as follows. The number of vCPUs requested by the waiting jobs except the first one is in [1, *MC*]. $rate\_D33$ is composed of the following three terms:

- $\left(k \cdot \mu \cdot P_{leave}^{(j,k,c_{rel})}\right)$. A job departing rate with c_{rel} vCPUs.
- $\left(P_{arrival}^{c_{req}} \Big/ \sum_{\gamma=m-j+1}^{MC} P_{arrival}^{\gamma}\right)$. The probability that the first waiting job, requesting c_{req} vCPUs, goes into service.
- $\Omega_{all}[c_{avail}-c_{req}, x-1]$. The probability (denoted as P_{left}) that the waiting jobs behind the first waiting job enter into service. The remaining $(x-1)$ jobs share $(c_{avail}-c_{req})$ vCPUs. The definition of Ω_{all} indicates that $P_{left} = \Omega_{all}[c_{avail}-c_{req}, x-1]$.

(iv) More than one waiting job can go into service after a job departure but the number of jobs entering service is less than i.

Equation (11) gives all the possible state transitions.

$$(i,j,k) \to (i-x, j-c_{rel}+c_{avail}, k+x-1) \tag{11}$$

Here $c_{rel} \in [\max\{1, j-(k-1)\cdot MC\}, \min\{MC, j-k+1\}]$, $c_{avail} \in [m-j+2, m-j+c_{rel}]$ and $x \in [2, \min\{i, c_{avail}-m+j\}]$. The transition rate denoted as $rate\_D34$ is $\left(rate\_D33 \cdot \left(\sum_{\gamma=(m-j+c_{rel}-c_{avail}+1)}^{MC} P_{arrival}^{\gamma}\right)\right)$, which consists of the following two terms:

- $rate\_D33$. In Equation (11), $rate\_D33$ denotes the probability that the first x waiting jobs enter into service.
- $\sum_{\gamma=(m-j+c_{rel}-c_{avail}+1)}^{MC} P_{arrival}^{\gamma}$. The probability that the $(x+1)^{th}$ waiting job cannot go into service.

4.2 Formulas for performance measures

We can now discuss how to compute all $\pi_{(i,j,k)}$ values. With the state transition rates in Section 4.1, we obtain the generator matrix Q with $\left((K+1)(m+1)^2\right)^2$ elements. Namely, $\left((K+1)(m+1)^2\right)$ balance equations, described in Equation (12).

$$\Pi Q = 0 \tag{12}$$

Here $\Pi = \{\pi_{(i,j,k)}, 0 \le i \le K, 0 \le j \le m, 0 \le k \le m\}$. We define

$$\Pi_{nonzero} = \{\pi_{(i,j,k)},\ i=0,\ 0 \le k \le m, k \le j \le \min(m, k \cdot MC)\} \cup \{\pi_{(i,j,k)}, 1 \le i \le K,\ 0 \le k \le m, \max\{k, m-MC+1\} \le j \le \min\{m, k \cdot MC\}\}$$

All the elements in $\Pi_{nonzero}$ are non-zero. $|\Pi_{nonzero}| = \left(\sum_{i=1}^{K}\sum_{k=0}^{m}\sum_{j=\max\{k,m-MC+1\}}^{\min\{m,k\cdot MC\}} 1 + \sum_{k=0}^{m}\sum_{j=k}^{\min\{m,k\cdot MC\}} 1\right)$ denotes the total elements in $\Pi_{nonzero}$. TABLE 2 gives the values of $|\Pi|$ and $|\Pi_{nonzero}|$ when m=120, K is from 20 to 60 and MC is from 1 to 8. TABLE 2 suggests the system parameters which cause computation overhead.

TABLE 2 STATE NUMBER

(m,K,MC)	$\lvert\Pi\rvert$	$\lvert\Pi_{nonzero}\rvert$
(120,60,8)	878401	55621
(120,60,4)	878401	27001
(120,60,1)	878401	181
(120,40,8)	590401	39221
(120,40,4)	590401	19841
(120,40,1)	590401	161
(120,20,8)	302401	22821
(120,20,4)	302401	12681
(120,20,1)	302401	141

To compute the steady-state probabilities, we first set the value of each element in $\Pi - \Pi_{nonzero}$ to zero, representing that the states in $\Pi - \Pi_{nonzero}$ do not exist. With $\left((K+1)(m+1)^2\right)$ balance equations from Equation (12) and the normalization equation $\sum_{i=0}^{K}\sum_{j=0}^{m}\sum_{k=0}^{m}\left(\pi_{(i,j,k)}\right) = 1$, we can use **Algorithm 1** to compute all $\pi_{(i,j,k)} \in \Pi_{nonzero}$ in the scenario where a job is dropped only due to no *nonFULL* PMs if this job is not rejected due to PMSA queue full. Note that these equations cannot be solved in closed form we must resort to a numerical solution as outlined in Section 5. Once obtaining the steady-state probabilities, we are able to compute the following performance measures:

1) Mean number of jobs in the PM waiting queue, given by

$$\sum_{i=0}^{K}\sum_{j=0}^{m}\sum_{k=0}^{m}\left(i \cdot \pi_{(i,j,k)}\right) \tag{13}$$

2) Mean number of jobs in VM service, given by

$$\sum_{i=0}^{K}\sum_{j=0}^{m}\sum_{k=0}^{m}\left(k \cdot \pi_{(i,j,k)}\right) \tag{14}$$

3) Mean number of busy vCPUs, given by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2560158, IEEE Transactions on Cloud Computing

$$\sum_{i=0}^{K}\sum_{j=0}^{m}\sum_{k=0}^{m}\left(j\cdot\pi_{(i,j,k)}\right) \tag{15}$$

4) Immediate service probability. If the waiting queue is empty and the number of the free vCPUs is no less than *MC*, the arriving job will go into service immediately. The corresponding probability is $\sum_{j=0}^{m-MC}\sum_{k=0}^{m}\pi_{(0,j,k)}$. If the waiting queue is empty and the number of the available vCPUs is less than *MC*, the probability that the arriving job will also get into service is $\sum_{j=m-MC+1}^{m}\left(\sum_{k=0}^{m}\left(\pi_{(0,j,k)}\cdot\sum_{\gamma=1}^{m-j}P_{arrival}^{\gamma}\right)\right)$. Hence, the immediate service probability is

$$\sum_{j=0}^{m-MC}\sum_{k=0}^{m}\pi_{(0,j,k)}+\sum_{j=m-MC+1}^{m}\left(\sum_{k=0}^{m}\left(\pi_{(0,j,k)}\cdot\sum_{\gamma=1}^{m-j}P_{arrival}^{\gamma}\right)\right) \tag{16}$$

5) Waiting service probability. If the waiting queue is not empty, the arriving job will go into the waiting queue immediately. The corresponding probability is $\sum_{i=1}^{K}\sum_{j=0}^{m}\sum_{k=0}^{m}\left(\pi_{(i,j,k)}\right)$. If the waiting queue is empty, the probability that the job will also get into the waiting queue is $\sum_{j=m-MC+1}^{m}\sum_{k=0}^{m}\left(\pi_{(0,j,k)}\cdot\sum_{\gamma=m-j+1}^{MC}P_{arrival}^{\gamma}\right)$. Hence, the total waiting service probability is

$$\sum_{i=1}^{K}\sum_{j=0}^{m}\sum_{k=0}^{m}\pi_{(i,j,k)}+\sum_{j=m-MC+1}^{m}\sum_{k=0}^{m}\left(\pi_{(0,j,k)}\cdot\sum_{\gamma=m-j+1}^{MC}P_{arrival}^{\gamma}\right) \tag{17}$$

6) Job rejection probability (P_{drop}). If the PM waiting queue is full, an arriving job is dropped. This probability is

$$\sum_{j=0}^{m}\sum_{k=0}^{m}\left(\pi_{(K,j,k)}\right) \tag{18}$$

7) Mean response time, given by

$$\frac{1}{\lambda\left(1-\sum_{j=0}^{m}\sum_{k=0}^{m}\left(\pi_{(K,j,k)}\right)\right)}\sum_{i=0}^{K}\sum_{j=0}^{m}\sum_{k=0}^{m}\left(i\cdot\pi_{(i,j,k)}\right)+\frac{1}{\mu} \tag{19}$$

5 Numerical Results

Our literature investigation of analytic model-based performance evaluation of IaaS CDCs indicated that only the models proposed in [16] and [17] considered VM heterogeneity. However, a direct comparison between their model results and the results of the model proposed in this paper is not very meaningful. The reason is that different assumptions (see in Section 2) are made in each modeling. Although several real CDC workloads such as CloudSuite [20] and Cloud traces [24][25] are available, these workloads give very little information about their common VM characteristics, such as the *VM size* [18]. Birke *et al.* [18] analyzed how virtualization technologies were utilized in contemporary data centers but the workload is not public and no information about job arrival and job service times were given. Thus, this section performs model evaluation by comparing our models' numerical results with the simulation results. We use the parameter settings used in [7] and [8] for experiments. In addition, in the PM model evaluation, we assume a wide range of values for our model parameters so that the model can represent a large variety of PMs in IaaS CDCs.

In the following, we first evaluate the accuracy of PM's monolithic model in Section 5.1 and then evaluate the capability of hierarchical model in Section 5.2. The results for both the analytical models and simulations are computed using Maple 18 [20]. All simulations and numerical analysis are performed on a workstation with Intel(R) Core(TM) i7-4790 CPU@3.60GHz and 8GB memory. In addition, the operating system is Windows 7. Note that each simulation is repeated 10 times and the average value of these repetitions is presented as the simulation results in the corresponding figures.

5.1 PM model accuracy under different parameters

We first present experiment results and discussions when *VM size* follows uniform distribution in Section 5.1.1. Then Section 5.1.2 presents experiment results and discussions when *VM size* follows geometric distribution.

5.1.1 Model accuracy under uniform distribution

Our model parameters are set as follows:

- ***m***. *m* varies from 10 to 120. These values are obtained from the VMmark [26] results. Thus, the maximum of 10, 20, 40, 60, 80, and 120 VMs can be deployed on a PM, respectively.
- ρ . It is changed from 10% to 90%.
- **PM buffer size** K . K is varied from 10 to 60.

We first set *m*=10 and do experiments under different *MC*, *K* and ρ. Then these experiments are repeated under *m*=20, 40, 60, 80 and 120, respectively. Experiment results in terms of mean response time and job rejection probability are shown in Fig.5-Fig.7. "M-m*y*" denotes the numerical result and "S-m*y*" denotes the corresponding simulation result. Here, *y* is integer variable, denoting the maximum number of vCPUs, *m*. In all these experiments, we set μ=1 and then any variation in *m*, *MC* or ρ can result in the change in the value of λ. When c_{req} is uniformly distributed, $\overline{P_{arrival}^{x}}=(MC+1)/2$. From the results of Fig.5-Fig.7, we observe that

- The numerical results closely match with the simulation results in terms of mean response time and job rejection probability.
- For each *m*, when *K* and ρ are both fixed, λ decreases with the increasing *MC* due to $\rho=\frac{\lambda(MC+1)}{2m\mu}$. Although λ decreases, both mean response time and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2560158, IEEE Transactions on Cloud Computing

the job rejection probability increase with the increasing *MC*. See the results in Fig.5. The reason is that with the increasing *MC*, it is more difficult for the arriving job to find enough free vCPUs and then it cannot go into service.

- For each *m*, when *K* and *MC* are both fixed, the number of free vCPUs is decreasing with the increasing ρ. Then the possibility that an arriving job obtains immediate service is becoming lower. Thus, mean response time and the job rejection probability both increase with the increasing ρ. See the results in Fig.6. Note that λ increases with the increasing ρ due to $\rho = \frac{\lambda(MC+1)}{2m\mu}$
- For each *m*, when ρ and *MC* are both fixed, variation in *K* has less effect on the mean number of jobs in service. But mean response time increases when increasing *K*. The job rejection probability decreases when increasing *K*. See the results in Fig.7.

5.1.2 Model accuracy under geometric distribution

We first set *m*=10 and do experiments by varying ρ from 10% to 90%. Then these experiments are repeated under *m*=20, 40, 60, 80 and 120, respectively. $MC = 1 + \frac{3}{p_{succ}}$, where p_{succ} is the probability of success in the geometric distribution. *MC* in all experiments is set to 7. Fig.8 shows the results, confirming the discussions and conclusions in Section 5.1.1.

5.2 Hierarchical model accuracy

This sub-section aims to evaluate the hierarchical model capability by comparing numerical results and simulation results. We assume *VM size* follows geometric distribution. *MC* is set as in Section 5.1.2. Considering the computation complexity of simulations under large N, we do experiments by setting N = 20, KC =50, K =16, $\varepsilon = 10^{-15}$, and $m = 20$. The job arrival rate at CDC varies from 10 jobs/hour to 28 jobs/hour. The PSAM searching delay is 1/900 hour. The mean job service time on a PM is 5 hours.

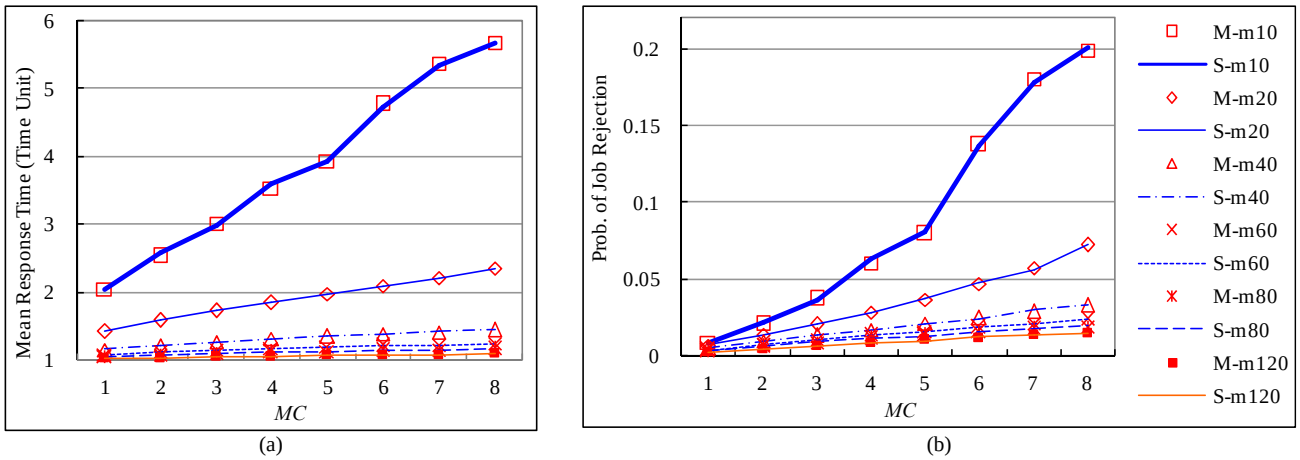


Fig.5 Effect of *MC* on PM model accuracy when ρ=0.9 and *K*=20

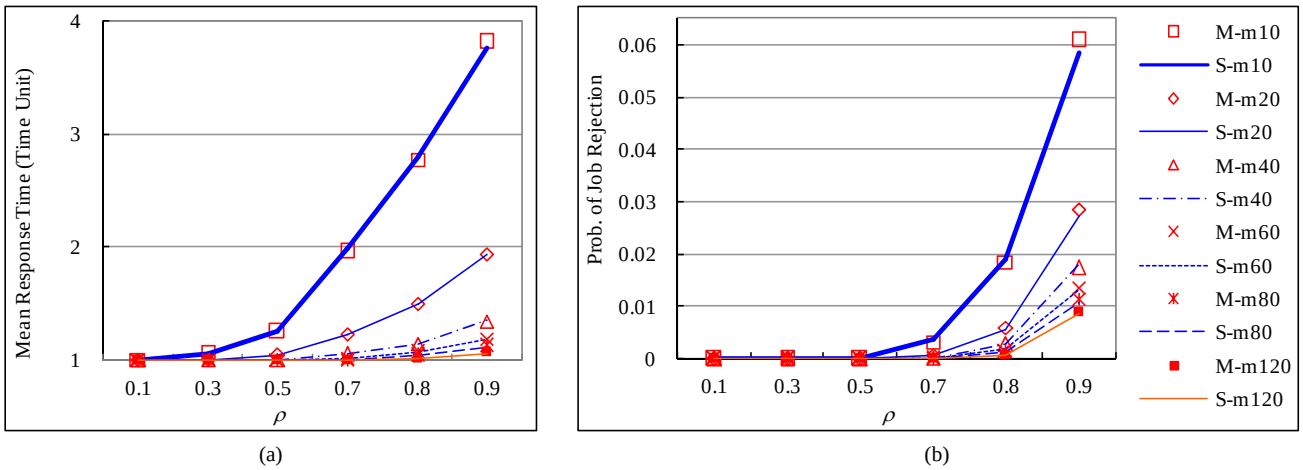
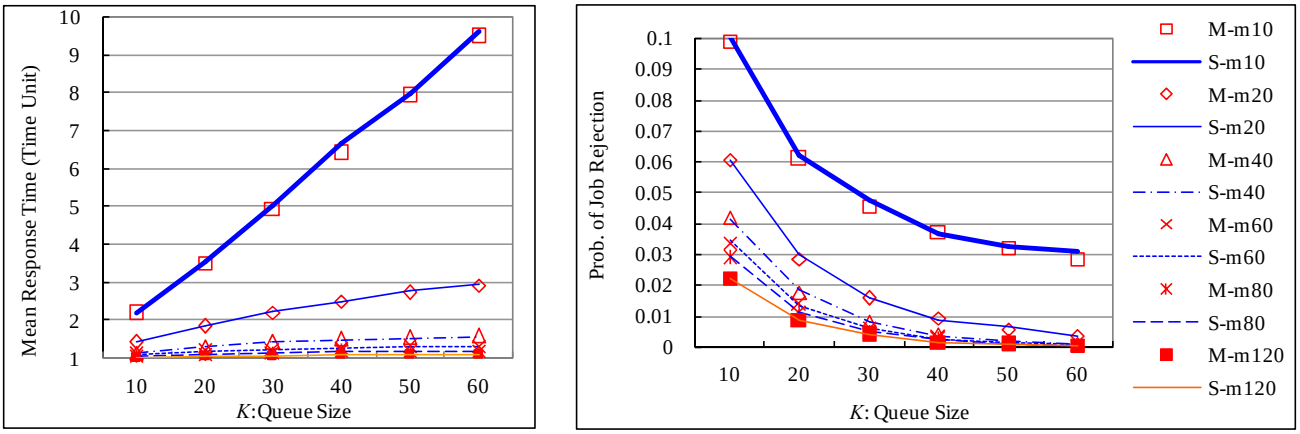


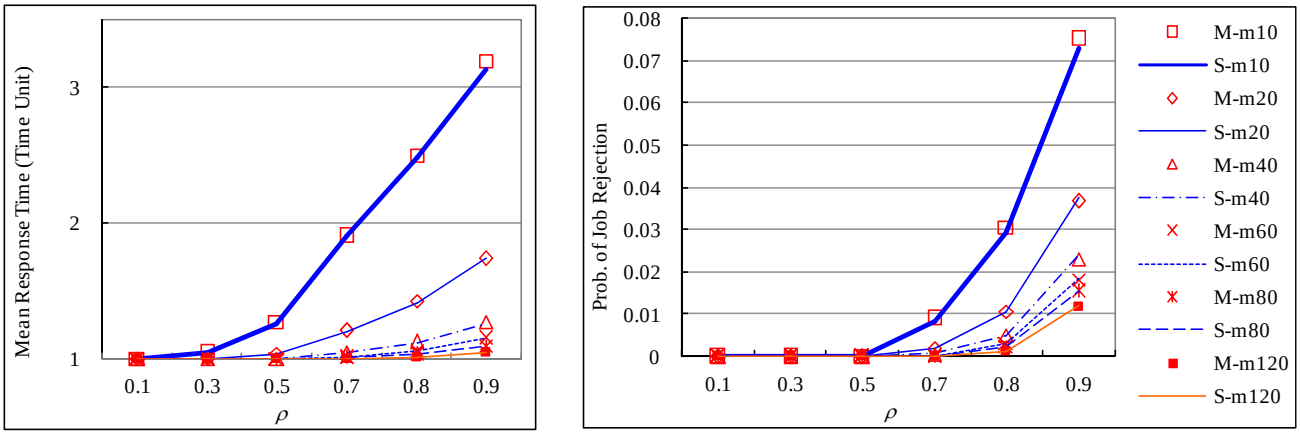
Fig.6 Effect of ρ on PM model accuracy when *MC*=4 and *K*=20

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2560158, IEEE Transactions on Cloud Computing



(a) (b)

Fig.7 Effect of K on PM model accuracy when MC=4 and ρ=0.9

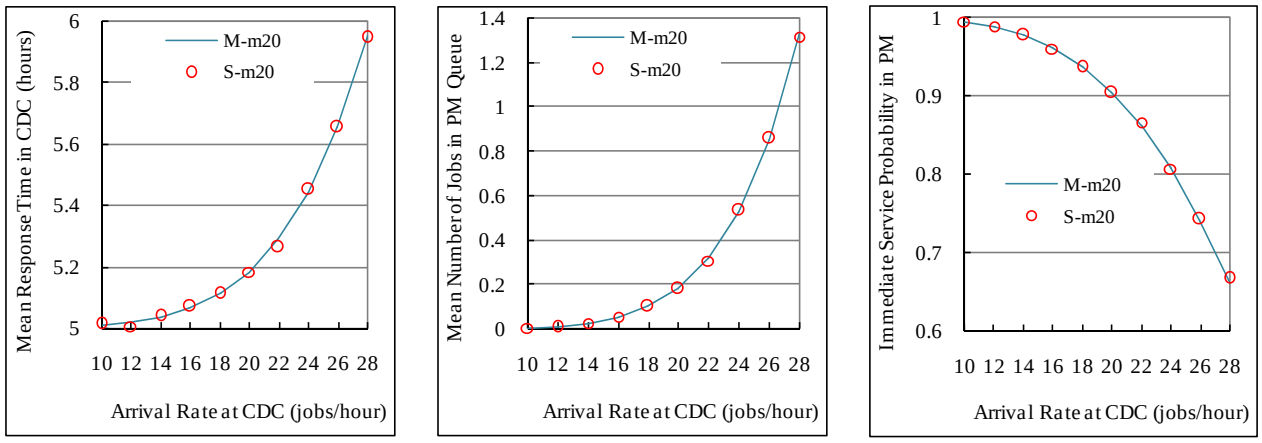


(a) (b)

Fig.8 Effect of ρ on model accuracy under geometric distribution and MC=7

TABLE 3 PM ρ AND ALGORITHM 1 ITERATION NUMBER OVER JOB ARRIVAL RATE UNDER GEOMETRIC DISTRIBUTION

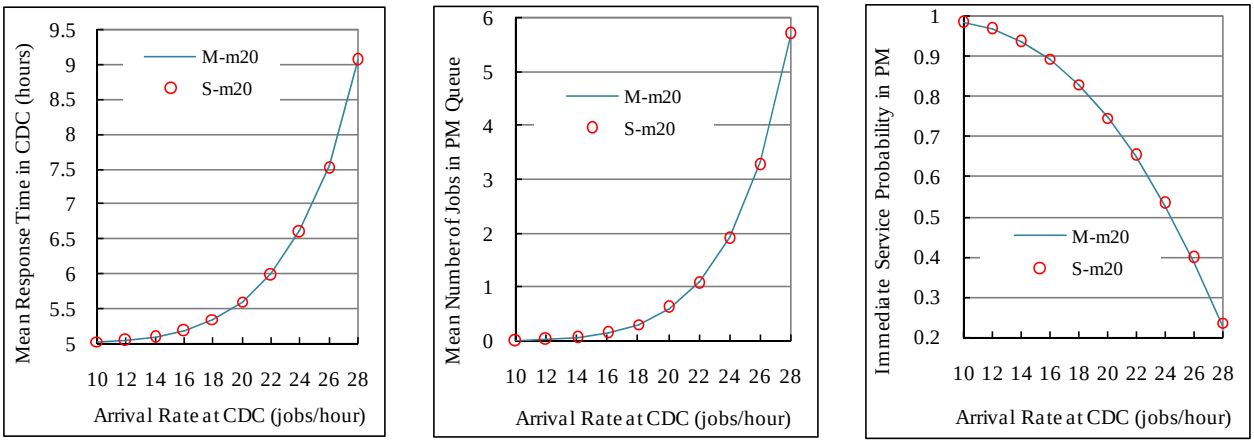
Job arrival rate (jobs/hour)	10	12	14	16	18	20	22	24	26	28
PM ρ	0.3125	0.3750	0.4375	0.5000	0.5625	0.6250	0.6877	0.7506	0.8140	0.8788
Iteration number	3	3	3	4	3	5	6	7	9	11



(a) (b) (c)

Fig.9 Effect of job arrival rate on system performance under geometric distribution

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2560158, IEEE Transactions on Cloud Computing



(a) (b) (c)

Fig.10 Effect of job arrival rate on system performance under uniform distribution

TABLE 4 PM ρ AND ALGORITHM 1 ITERATION NUMBER OVER JOB ARRIVAL RATE UNDER UNIFORM DISTRIBUTION

Job arrival rate (jobs/hour)	10	12	14	16	18	20	22	24	26	28
PM ρ	0.3130	0.3750	0.4380	0.5000	0.5625	0.6252	0.6890	0.7538	0.8251	0.9150
Iteration number	2	3	3	4	5	7	9	12	19	36

We compare the values of three performance measures: (i) mean response time denoting the mean time of a job in the CDC, (ii) PM immediate service probability defined in Equation (16), and (iii) mean number of jobs in PM queue. Job rejection probability from the CDC is too small to be shown. TABLE 3 gives the corresponding vCPU utilization ρ and iteration number of **Algorithm 1** under different job arrival rates. Fig.9 shows the performance variation under different job arrival rates. We also repeat experiments when *VM size* follows uniform distribution with MC=4. TABLE 4 and Fig.10 show the results. The results indicate that:

- The hierarchical model can approximately capture the CDC system behavior while being scalable, shown in Fig.9 (a) and Fig.10 (a). This means that the errors introduced by the decomposition of the monolithic model for a CDC are insignificant. In addition, this indicates the effectiveness of **Algorithm 1**.
- With the increasing job arrival rate, the iteration number is increasing. It is because the steady-state value of $N_{nonFULL}$ is decreased with the increasing job arrival rate. Then more time is required for $N_{nonFULL}$ from its initial value until its steady-state value.

6 Model Discussions

This section first describes a new PMSA model. Then we discuss the situations where there exist different types of applications.

6.1 New PMSA model

Note that that there are a lot of PMs on a CDC. Therefore, even if the number of job arriving to an IaaS data center is high, the probability of job arriving at an individual PM is low. So Poisson process can be used as an adequate model for the job arrival process at a PM. It is certainly possible to relax the assumption of exponential distribution by using several known approaches at the cost of less scalable models [36]. Thus, this section only discusses how to extend the PMSA model described in Section 3 to the situation of a more complex (non-Poisson) job arrival process at CDC.

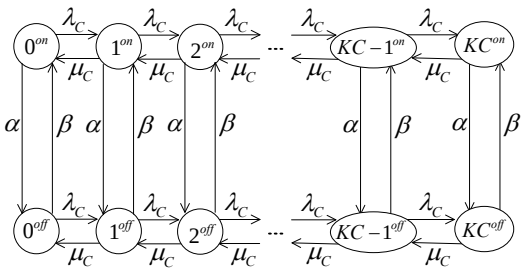


Fig.11 New PMSA model under an On-Off arrival process

We replace the Poisson job arrival process at the PMSA with an On-Off arrival process. An On-Off arrival process is a special case of the Markov Modulated Poisson Process (MMPP) [28], where an arrival source alternates between an "On" state and an "Off" state. In the "On" state jobs arrive with exponentially distributed inter-arrival times, while in the "Off" state no job arrival occurs. The sojourn times of the arrival source in both states are assumed to follow exponential distributions with possibly different rates. This choice of the arrival processes serves two purposes. Firstly, it captures the output process of PMSA more accurately than the previous Poisson assumption, while retaining much of the simplicity. Secondly, it serves as the

basis for analyzing more complex job arrival processes at the system, which in reality may be rarely Poisson.

Fig.11 describes new RMPA model. $1/\alpha$ denotes mean time for the arrival source to stay in the "On" state and $1/\beta$ denotes mean time for the arrival source to stay in the "Off" state. In this new model, $P_{block} = \pi(KC^{on})$. When the value of KC is small (e.g., $KC \le 10$), the steady-state balance equations of the Markov chain in Fig.11 can be solved using well-known numerical approaches such as Gauss-Seidel method or Successive Over-relaxation. However, as KC grows larger it is desirable to find a more efficient solution method. There are several candidates. One is to first get the block structure of the underlying generator matrix of the PMSA model. Then we apply methods such as matrix-geometric solution [29] and spectral expansion method [30] to get model solution.

6.2 Different virtualized resource types

This paper only considers the vCPU allocation. In real CDCs, we need to consider the allocation of other virtualized resources, including memory, I/O, and network bandwidth. When only one type of resource is crucial, our modeling approach can be directly applicable. When a tenant requires more than one type of resource, and these resources are all crucial, it could be handled by the modeling approach in two ways. The first is that we could define a composite measure of these weighted resources. Then the modeling approach also could be applicable directly. The second is for the situation where we need to consider each type of resource explicitly. Then we need to extend the state space of PM model proposed in this paper and also re-define the state transition rules. We leave the modeling of the second situation for future work.

7 Conclusions

This paper presents a hierarchical stochastic model for quantifying the effects of variation in job arrival rate, buffer size, the maximum vCPU numbers on a PM and *VM size* distribution on the quality of cloud services. We describe the details of the proposed model and the formulas for key performance metrics. Extensive experiments are conducted to evaluate the proposed model.

This paper assumes all PMs in the active state. We are planning to extend the proposed models to the scenario including PMs in other states, such as sleep and idle. In addition, we plan to analyze the existing public workload traces and then explore the possibilities of mapping some trace information to *VM size*. We will also consider collecting private cloud workloads for job processing information and virtualized resource consumption information. Then we will use these realistic traces to parameterize the proposed models and carry out simulations. Note that this paper only considers the simple FCFS scheduling policy. Resource allocation mechanisms have a significant influence on QoS and system resource utilization [31]-[34]. Furthermore, resource sharing techniques and system virtualization also often introduce performance interference among co-hosted VMs and then cause performance degradation [27]. Thus, our future work will extend the proposed model to capture these occurrences and the scenarios of the other scheduling policies.

Acknowledgment

The research of Xiaolin Chang and Jiqiang Liu is supported in part by NSF 61572066 of China. The research of Ruofan Xia and Kishor S. Trivedi is supported in part under US NSF grant number CNS-1523994, by IBM under a faculty grant and by NATO under Science for Peace project number 984425. The research of Jogesh K. Muppala is supported in part by RGC GRF HKUST 612912.

References

[1] Gartner: Forecast Analysis: Public Cloud Services, Worldwide 2Q2014 Update August 11, 2014.

[2] C. Reiss, A. Tumanov, G. Ganger, R. Katz, and M. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," In ACM Symposium on Cloud Comp, 2012.

[3] http://aws.amazon.com/ec2/instance-types.

[4] http://azure.microsoft.com/en-gb/pricing/details/virtual-machines/?rnd=1.

[5] http://www.rackspace.com/cloud/public-pricing.

[6] https://cloud.google.com/compute/docs/machine-types.

[7] R. Ghosh, F. Longo, V. K. Naik, and K.S. Trivedi, "Modeling and Performance Analysis of Large Scale IaaS Clouds," In Future Generation Computing Systems (FGCS), 2013.

[8] D. Bruneo, F. Longo, and A. Puliafito, "A Stochastic Model to Investigate Data Center Performance and QoS in IaaS Cloud Computing Systems," In IEEE Trans. Parallel Distrib. Syst. 25(3): 560-569 (2014).

[9] X. Chang, B. Wang, J.K. Muppala, and J.Q. Liu, "Modeling Active Virtual Machines on IaaS Clouds Using an M/G/m/m+K Queue," To be published in IEEE Transactions on Services Computing.

[10] H. Khazaei, J. V. Misic, Vojislav B. Misic, and S. Rashwand, "Analysis of a Pool Management Scheme for Cloud Computing Centers," IEEE Trans. Parallel Distrib. Syst. 24(5): 849-861 (2013).

[11] S. K. Garg, A. N. Toosi, S.K. Gopalaiyengar, and R. Buyya,"SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter," J. Network and Computer Applications 45: 108-120 (2014)

[12] M. Guevara, B. Lubin, and B.C. Lee, "Market mechanisms for managing datacenters with heterogeneous microarchitectures," ACM Trans. Comput. Syst. 32(1): 3 (2014).

[13] K.V. Vishwanath, and N. Nagappan, "Characterizing cloud computing hardware reliability," In SoCC '10: Proceedings of the 1st ACM symposium on Cloud computing, pages 193–204, Indianapolis, USA, 2010.

[14] R. Ghosh, F. Longo, F. Frattini, S. Russo, and K.S. Trivedi, "Scalable Analytics for IaaS Cloud Availability," IEEE T. Cloud Computing 2(1): 57-70 (2014).

[15] Y.N. Xia, X. Luo, J. Li, and Q.S. Zhu, "A Petri-Net-Based Approach to Reliability Determination of Ontology-Based Service Compositions," In IEEE T. Systems, Man, and Cybernetics: Systems 43(5): 1240-1247 (2013).

[16] H. Khazaei, J. V. Misic, Vojislav B. Misic, and N. B. Mohammadi, "Modeling the Performance of Heterogeneous IaaS Cloud Centers," ICDCS Workshops 2013: 232-237.

[17] B. Wang, X.L. Chang, J.Q. Liu, "Modeling the Performance of Heterogeneous IaaS Cloud Centers," IEEE Communications Letters 19(4): 537-540 (2015).

[18] R. Birke, A. Podzimek, L. Y. Chen, and E. Smirni, "State-of-the-practice in data center virtualization: Toward a better understanding of VM usage," DSN 2013: 1-12.

[19] X. Chen, C.P. Ho, R. Osman, P.G. Harrison, W.J. Knottenbelt, "Understanding, modelling, and improving the performance of web applications in multicore virtualised environments," ICPE 2014: 197-207.

[20] M. Ferdman, A. Adileh, Y. O. Koçberber, S. Volos, M.Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the clouds: a study of emerging scale-out workloads on modern hardware," ASPLOS 2012: 37-48.

[21] Maplesoft, Inc., Maple 18, http://www.maplesoft.com/products/maple/.

[22] S. Meng, A. K. Iyengar, I. M. Rouvellou, L. Liu, K. Lee, B. Palanisamy, and Y.Z. Tang, "Reliable State Monitoring in Cloud Data centers," In Proc. IEEE CLOUD, 2012, pp. 951-958.

[23] G. Ren, E. Tune, T. Moseley, Y.X. Shi, S. Rus, and R. Hundt,"Google-Wide Profiling: A Continuous Profiling Infrastructure for Data Centers," In IEEE Micro 30(4): 65-79, 2010.

[24] Google. Google Cluster Data V2. Available: http://code.google.com/p/googleclusterdata/wiki/ClusterData2011_1.

[25] Yahoo. Yahoo! M45 Supercomputing Project . Available: http://research.yahoo.com/node/1884.

[26] http://www.vmware.com/a/assets/vmmark/

[27] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving mapreduce performance in heterogeneous environments." In OSDI, 2008.

[28] K. S. Trivedi, Probability and Statistics with Reliability, Queuing and Computer Science Applications, second ed., Wiley, 2001.

[29] M. F. Neuts. Matrix-geometric solutions in stochastic models: an algorithmic approach. Courier Dover Publications, 1981.

[30] R. Chakka, Spectral expansion solution for some finite capacity queues, Annals of Operations Research, 79(1998), 27-44.

[31] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant Resource Fairness: fair allocation of multiple resource types," NSDI 2011: 323–326.

[32] C. Delimitrou and C. Kozyrakis, "Quasar: resource-efficient and QoS-aware cluster management, "ASPLOS 2014: 127–144.

[33] S. Di, D. Kondo, and C.L. Wang, "Optimization of Composite Cloud Service Processing with Virtual Machines," In IEEE Trans. Computers 64(6): 1755-1768 (2015).

[34] A.Verma, L.Pedrosa, M.Korupolu, D.Oppenheimer, E.Tune, AND J.Wilkes, "Large-scale cluster management at Google with Borg. EuroSys 2015: 18:1-18:17.

[35] M. Balduzzi, J. Zaddach, D. Balzarotti, E.Kirda, and S. Loureiro,"A security analysis of amazon's elastic compute cloud service," SAC 2012: 1427-1434.

[36] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, "Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications," John Wiley, second edition, New York, NY, 2006.

Xiaolin Chang received the PhD degree in computer science from Hong Kong University of Science and Technology in 2005. She is currently an associate professor in School of Computer and Information Technology at Beijing Jiaotong University. Her current research interests include Cloud data center and Network security. She is a member of IEEE.

Ruofan Xia received the Ph. D. degree in Electrical Engineering from Duke University, Durham, NC in 2015. His research interests include computer system reliability, availability, performance modeling and optimization with a focus on distributed systems.

Jogesh K. Muppala (M'85-SM'02) received the Ph. D. degree in Electrical Engineering from Duke University, Durham, NC in 1991. He is currently an associate professor in the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. He was involved in the development of modeling techniques for systems and software. Dr. Muppala is a Senior Member of IEEE, IEEE Computer Society and IEEE Communications Society, and a participating representative from HKUST with EDUCAUSE.

Kishor S. Trivedi holds the Hudson chair in the Department of Electrical and Computer Engineering at Duke University. He is the author of Probability and Statistics with Reliability, Queuing and Computer Science Applications, published by John Wiley. He has published more than 500 articles and has supervised 45 PhD dissertations. He received the IEEE Computer Society Technical Achievement Award for his research on Software Aging and Rejuvenation.

Jiqiang Liu received his B.S. (1994) and PhD (1999) degree from Beijing Normal University. He is currently a Professor at the School of Computer and Information Technology, Beijing Jiaotong University. He has published over 60 scientific papers in various journals and international conferences. His main research interests are trusted computing, cryptographic protocols, privacy preserving and network security.