Performance of Cloud Centers with High Degree of Virtualization under Batch Task Arrivals

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

Abstract—In this paper, we evaluate the performance of cloud centers with high degree of virtualization and Poisson batch task arrivals. To this end, we develop an analytical model and validate it with independent simulation model. Task service times are modeled with a general probability distribution, but the model also accounts for the deterioration of performance due to the workload at each node. The model allows for calculation of important performance indicators such as mean response time, waiting time in the queue, queue length, blocking probability, probability of immediate service, and probability distribution of the number of tasks in the system. Furthermore, we show that the performance of a cloud center may be improved if incoming requests are partitioned on the basis of the coefficient of variation of service time and batch size.

Index Terms—cloud computing, performance modeling, quality of service, response time, virtualized environment, stochastic process

1 Introduction

Cloud computing is a novel computing paradigm in which different computing resources such as infrastructure, platforms and software applications are made accessible over the internet to remote users as services [20]. The use of cloud computing is growing quickly: spending on cloud-related technologies, hardware and software is expected to grow to more than 45 billion dollars by 2013 [16]. Performance evaluation of cloud centers is an important research task, which is rendered difficult by the dynamic nature of cloud environments and diversity of user requests [23]. It is not surprising, then, that only a portion of a number of recent research results in the area of cloud computing has been devoted to performance evaluation. Performance evaluation is particularly challenging in scenarios where virtualization is used to provide a well defined set of computing resources to the users [6], and even more so when the degree of virtualization – i.e., the number of virtual machines (VMs) running on a single physical machine (PM) is high, at the current state of technology, means well over hundred of VMs. For example, the recent VMmark benchmark results by VMware [22], a single physical machine can run as many as 35 tiles or about 200 VMs with satisfactory performance. Table 1 shows the workloads and applications being run with each VMmark tile. A tile consists of six different VMs that imitate a typical data center environment. Note that the standby server virtual machine does not run an application; however, it does run an operating system and is configured as 1 CPU with a specified amount of memory and disk space [21].

In this paper, we address this deficiency (i.e., lack of research in the area) by proposing an analytical and simulation model for performance evaluation of cloud centers. The model utilizes queuing theory and probabilistic analysis to obtain a number of performance indicators, including response time waiting time in the queue, queue length, blocking probability, probability of immediate service, and probability distribution of the number of tasks in the system.

We assume that the cloud center consists of many physical machines, each of which can host a number of virtual machines, as shown in Fig. 1. Incoming requests are routed through a load balancing server to one of the PMs. Users can request one or more VMs at a time, i.e., we allow batch (or super-) task arrivals, which is consistent with the so-called *On-Demand* services in the Amazon Elastic Compute Cloud (EC2) [1]; as these services provide no advance reservation and no long-term commitment, clients may experience delays in fulfillment of requests. While the number of potential users is high, each user typically submits single batch request at a time with low probability. Therefore, the super-task arrival can be adequately modeled as a Poisson process [8].

When a super-task arrives, the load balancing server attempts to provision it – i.e., allocate it to a single PM with the necessary capacity. As each PM has a finite input queue for individual tasks, this means that the incoming super-tasks are processed as follows:

- If a PM with sufficient spare capacity is found, the

- *H. Khazaei is with the Department of Computer Science, University of Manitoba, Winnipeg, MB, Canada R3T 2N2.*
  E-mail: hamzehk@cs.umanitoba.ca
- *J. Mišić and V. B. Mišić are with the Department of Computer Science, Ryerson University, Toronto, ON, Canada M5B 2K3.*
  E-mail: jmisic@scs.ryerson.ca, vmisic@scs.ryerson.ca

TABLE 1
VMmark workload summary per tile (adapted from [9]).

Virtual Machine	Benchmark	Operating System	Virtual Machine Spec.	Metric
Database Server	SysBench with MySQL	SUSE Linux ES x64	2 vCPUs & 2GB RAM	Commits/min
Mail Server	LoadSIM	Microsoft Win. 2003 x86	2 vCPUs & 1GB RAM	Actions/min
Java Server	Modified SPECjbb2005	Microsoft Win. 2003 x64	2 vCPUs & 1GB RAM	New orders/min
File Server	Dbench	SUSE Linux ES x86	1 vCPUs & 256 MB RAM	MB/sec
Web Server	Modified SPECjbb2005	SUSE Linux ES x64	2 vCPUs & 512 MB RAM	Accesses/min
Standby	N/A	Microsoft Win. 2003 x86	1 vCPUs & 256 MB RAM	N/A

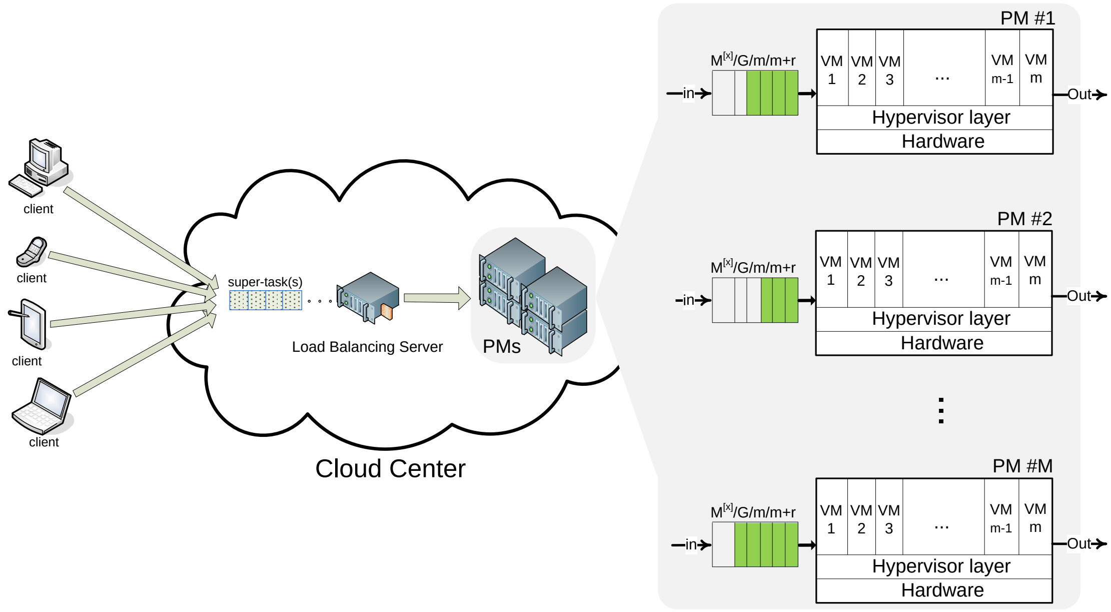


Fig. 1. The architecture of the cloud center.

super-task is provisioned immediately.

- Else, if a PM with sufficient space in the input queue can be found, the tasks within the super-task are queued for execution.
- Otherwise, the super-task is rejected.

In this manner, all tasks within a super-task are processed together, whether they are accepted or rejected. This policy, known as total acceptance/rejection, benefits the users that typically would not accept partial fulfillment of their service requests. It also benefits the cloud providers since provisioning of all tasks within a super-task on a single PM reduces inter-task communication and, thus, improves performance.

As statistical properties of task service times are not well known and cloud providers don't publish relevant data, it is safe to assume a general distribution for task service times, preferably one that allows the coefficient of variation (*CoV*, defined as the ratio of standard deviation and mean value) to be adjusted independently of the mean value [4]. However, those service times apply to a VM running on a non-loaded PM. When the total workload of a PM increases, so does the overhead required to run several VMs simultaneously, and the actual service times will increase. Our model incorporates this adjustment as well, as will be seen below.

In summary, our work advances the performance analysis of cloud computing centers by incorporating a high degree of virtualization, batch arrival of tasks and generally distributed service time. These key aspects of cloud centers have not been addressed in a single performance model previously.

Specifically, the contributions of this paper are as follows:

- We have developed a tractable model for the performance modeling of a cloud center including PMs that support a high degree of virtualization, batch arrivals of tasks, generally distributed batch size and generally distributed task service time. The approximation error becomes negligible at moderate to high loads.
- Our model provides full probability distribution of the task waiting time, task response time, and the number of tasks in the system – in service as well as in the PMs' queues. The model also allows easy calculation of other relevant performance indicators such as the probability that a super-task is blocked and probability that a super-task will obtain immediate service.
- We show that the performance of cloud centers is very dependent on the coefficient of variation, *CoV*, of the

task service time as well as on the batch size (i.e., the number of tasks in a super-task). Larger batch sizes and/or values of coefficient of variation of task service time that exceed one, result in longer response time but also in lower utilization for cloud providers.

- Finally, we show that performance for larger batch sizes and/or high values for *CoV* might be improved by partitioning the incoming super-tasks to make them more homogeneous, and processing the resulting super-task streams through separate cloud sub-centers.

The paper is organized as follows. In Section 2, we survey related work in cloud performance analysis as well as in queuing system analysis. Section 3 presents our model and the details of the analysis. Section 4 presents the numerical results obtained from the analytical model, as well as those obtained through simulation. Finally, Section 5 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. In [24], 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 [26], 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 [25], the cloud center was modeled as an $M/M/m/m+r$ queuing system from which the distribution of response time was determined. Inter-arrival and service times were both assumed to be exponentially distributed, and the system has a finite buffer of size r. 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). We addressed the primary defect of [25] by allowing the service time to be generally distributed in [11].

A hierarchical modeling approach for evaluating quality of experience in a cloud computing environment was proposed in [17]. Due to simplified assumptions, authors applied classical Erlang loss formula and $M/M/m/K$ queuing system for outbound bandwidth and response time modeling respectively.

In [13], [10], [7], [12], the authors proposed general analytic models based approach for an end-to-end performance analysis of a cloud service. The proposed approach reduces the complexity of performance analysis of cloud centers by dividing the overall model into Markovian sub-models and then obtaining the overall solution by iteration over individual sub-model solutions. Models are limited to exponentially distributed inter-events times and a weak virtualization degree (i.e., less than 10 VMs per PM), which are not quite realistic.

As can be seen, there is no research work that considers batch arrival of user tasks (i.e., super-task). In all related work, service time is assumed to be exponentially distributed while there is no concrete evidence for such an assumption. Also, virtualization has not been sufficiently addressed or ignored at all. Our performance model tries to address these deficiencies.

Related work regarding theoretical analysis of queuing systems is presented in the Appendix A[1].

3 Analytical Model

We assume that the cloud center consists of M identical PMs with up to m VMs each, as shown in Fig. 1. Super-task arrivals follow a Poisson process, as explained above, and the inter-arrival time A is exponentially distributed with a rate of $\frac{1}{\lambda_i}$. We denote its cumulative distribution function (CDF) with $A(x) = Prob[A < x]$ and its probability density function (pdf) with $a(x) = \lambda_i e^{-\lambda x}$; the Laplace-Stieltjes Transform (LST) of this pdf is

$$A^*(s) = \int_0^\infty e^{-sx} a(x)dx = \frac{\lambda_i}{\lambda_i + s}.$$

Note that the total arrival rate to the cloud center is λ so that

$$\lambda = \sum_{i=1}^{M} \lambda_i.$$

Let g_k be the probability that the super-task size is $k = 1, 2, \ldots MBS$, in which *MBS* is the maximum batch size which, for obvious reasons, must be finite; both g_k and *MBS* depend on user application. Let $\overline{g}$ and $\Pi_g(z)$ be the mean value and probability generating function (PGF) of the task burst size, respectively:

$$\begin{aligned} \Pi_g(z) &= \sum_{k=1}^{MBS} g_k z^k \\ \overline{g} &= \Pi_g^{(1)}(1) \end{aligned} \tag{1}$$

Therefore, we model each PM in the cloud center as an $M^{[x]}/G/m/m+r$ queuing system. Each PM may run up to m VMs and has a queue size of r tasks.

Service times of tasks within a super-task are identically and independently distributed according to a general distribution. However, the parameters of the general distribution (e.g., mean value, variance and etc.) are dependent on the virtualization degree of PMs. The degradation of mean service time due to workload can be approximated using data from recent cloud benchmarks [2]; unfortunately, nothing can be inferred about the behavior of higher moments of the

[1] Appendices can be found in the online supplement of this paper.

service time [3]. For example, Fig. 2 depicts the VMmark performance results for a family of same-generation PMs under different number of deployed tiles [22]. The VMmark score is an overall performance of a server under virtualized environment; a detailed description of VMmark score can be found in [21]. From date in Fig. 2, we can extract the dependency of mean service time on the number of deployed VMs, normalized to the service time obtained when the PM runs a single tile, as shown in Fig. 2. Note that mean service time represents both computation and communication time for a task within super-tasks.

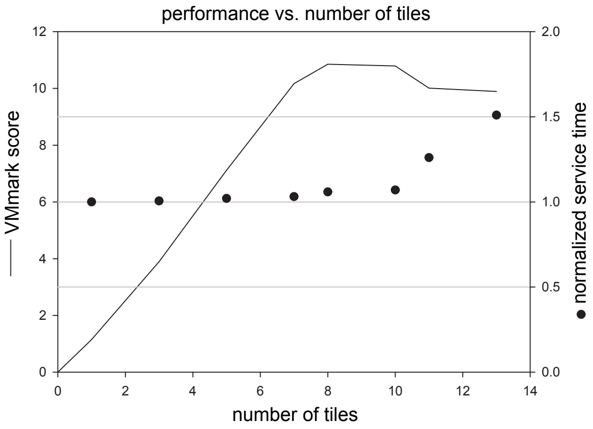


Fig. 2. Performance vs. no. of tiles.

The dependency shown in Fig. 2 may be approximated to give the normalized mean service time, $\overline{b}_n(y)$, per tile as a function of virtualization degree:

$$\overline{b}_n(y) = \frac{\overline{b}(y)}{\overline{b}(1)} = 1/(0.996 - (8.159 * 10^{-6}) * y^{4.143}) \quad (2)$$

where y denotes the number of tiles deployed on a single PM. Therefore, the CDF of the service time will be $B_y(x) = Prob\,[B_y < x]$ and its pdf is $b_y(x)$, while the corresponding LST is

$$B_y^*(s) = \int_0^\infty e^{-sx} b_y(x)dx.$$

The mean service time is then:

$$\overline{b}(y) = -B_y^{'*}(0).$$

If the maximum number of allowed tiles on each PM is Ω, then the aggregate LST will be

$$B^*(s) = \sum_{y=1}^{\Omega} p_y B_y^*(s).$$

where p_y is the probability of having y tiles on a single PM. Then the aggregate mean service time is

$$\overline{b} = -B^{'*}(0) = \sum_{y=1}^{\Omega} p_y \overline{b}(y).$$

The mass probabilities $\{p_y,\ y = 1\,..\,\Omega\}$ are dependent on the *traffic intensity* and obligations stemming from service level agreement between the provider and the user. A cloud provider may decide to deploy more tiles on PMs to reduce the waiting time – which, according to (2), will lead to deterioration of service time. We will examine the dependency between y and the traffic intensity in our numerical validation.

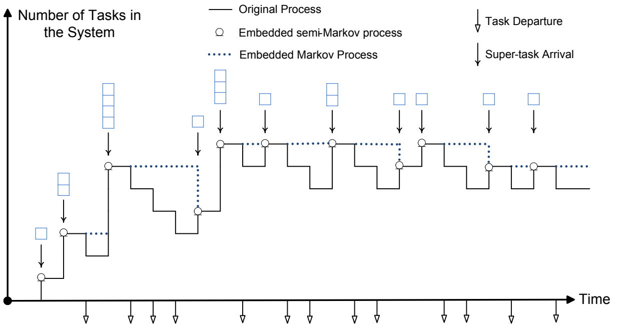


Fig. 3. A sample path of the original process, embedded semi-Markov process, and embedded Markov process.

Residual task service time, denoted with B_+, is the time interval from an arbitrary point during a service interval to the end of that interval. *Elapsed task service time*, denoted with B_-, is the time interval from the beginning of a service interval to an arbitrary point within that interval. Residual and elapsed task service times have the same probability distribution [19], the LST of which can be calculated as

$$B_+^*(s) = B_-^*(s) = \frac{1 - B^*(s)}{s\overline{b}} \quad (3)$$

The traffic intensity may be defined as $\rho \triangleq \frac{\lambda \overline{g}}{m\overline{b}}$. For practical reasons, we assume that $\rho < 1$.

3.1 Approximation by embedded processes

To solve this model, we adopt the technique of embedded Markov chain [19], [14]. The problem stems from the fact that our $M^{[x]}/G/m/m + r$ system is non-Markovian, and the approximate solution requires two steps.

First, we model our non-Markovian system with an embedded semi-Markov process, hereafter referred to as eSMP, that coincides with the original system at moments of super-task arrivals and task departures. To determine the transition probabilities of the embedded semi-Markov process, we need to count the number of task departures between two successive task arrivals (see Fig. 3).

Second, as the counting process is intractable, we introduce an approximate embedded Markov process, hereafter referred to as aEMP, that models the embedded semi-Markov process in a discrete manner (see Fig. 4). In aEMP, task departures that occur between two successive super-task arrivals (of which there can be zero, one, or several) are considered to occur at the times of super-task arrivals.

Therefore, aEMP models the embedded semi-Markov process (and, by extension, the original process) only at the moments of super-task arrivals. Since we can't determine

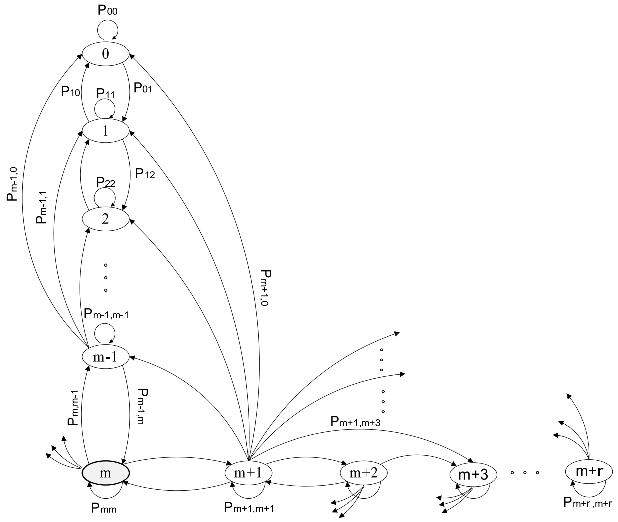


Fig. 4. Approximate embedded Markov chain (aEMC).

the exact moments of task departures, the counting process is only approximate, but the approximation error is rather low and decreases with an increase in load, as will be discussed below.

A sample path of the three processes: the original one, the embedded semi-Markov, and the approximate embedded Markov one, is shown schematically in Fig. 3.

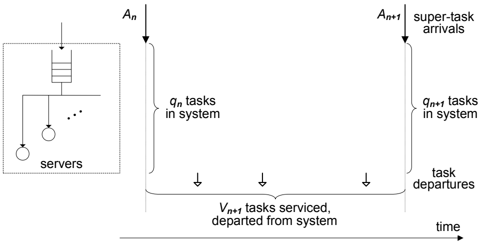


Fig. 5. Observation points.

The approximate embedded Markov chain, hereafter referred to as aEMC, that corresponds to aEMP is shown in Fig. 4. States are numbered according to the number of tasks currently in the system (which includes both those in service and those awaiting service). For clarity, some transitions are not fully drawn.

Let A_n and A_{n+1} indicate the moment of n^{th} and $(n+1)^{th}$ super-task arrivals to the system, respectively, while q_n and q_{n+1} indicate the number of tasks found in the system immediately before these arrivals; this is schematically shown in Fig.5. If k is the size of super-task and v_{n+1} indicates the number of tasks which depart from the system between A_n and A_{n+1}, then, $q_{n+1} = q_n - v_{n+1} + k$.

To solve aEMC, we need to calculate the corresponding transition probabilities, defined as

$$P(i,j,k) \triangleq Prob\left[q_{n+1} = j | q_n = i \text{ and } X_g = k\right].$$

In other words, we need to find the probability that $i+k-j$ customers are served during the interval between two successive super-task arrivals. Such counting process requires the exact knowledge of system behavior between two super-task arrivals. Obviously for $j > i+k$, $P(i,j,k) = 0$. Since there are at most $i+k$ tasks present between the arrival of A_n and A_{n+1}. For calculating other transition probabilities in aEMC, we need to identify the distribution function of residence time for each state in eSMP. Let us now describe these residence times in more detail.

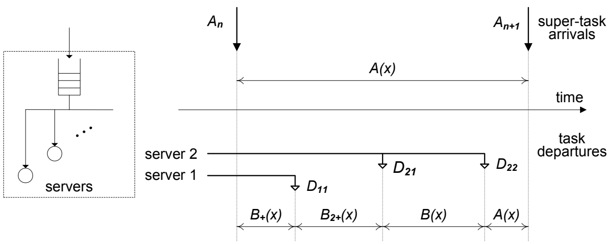


Fig. 6. System behavior between two observation points.

- Case 1: The state residence time for the first departure is remaining service time, $B_+(x)$, since the last arrival is a random point in the service time of the current task.
- Case 2: If there is a second departure from the same server, then clearly the state residence time is the service time ($B(x)$).
- Case 3: No departure between two super-task arrivals as well as the last departure before the next arrival make the state residence time exponentially distributed with the mean value of $\frac{1}{\lambda}$.
- Case 4: If i^{th} departure is from another server, then the CDF of state residence time is $B_{i+}(x)$. Consider the departure D_{21} in Fig. 6, which takes place after departure D_{11}. Therefore the moment of D_{11} could be considered as an arbitrary point in the remaining service time of the task in server #2, so the CDF of residence time for the second departure is $B_{2+}(x)$. As a result, the LST of $B_{2+}(x)$ is the same as $B^*_+(s)$, similar to (3) but with an extra recursion step. Generally, the LSTs of residence times between subsequent departures from different servers may be recursively defined as follow

$$B^*_{i+}(s) = \frac{1 - B^*_{(i-1)+}(s)}{s \cdot \overline{b}_{(i-1)+}}, \quad i = 1,2,3,\cdots \tag{4}$$

in which $\overline{b}_{(i-1)+} = [-\frac{d}{ds}B^*_{(i-1)+}(s)]_{s=0}$. To maintain the consistency in notation we also define $\overline{b}_{0+} = \overline{b}$, $B^*_{0+}(s) = B^*(s)$ and $B^*_{1+}(s) = B^*_+(s)$.

Let $R_k(x)$ denotes the CDF of residence times at state k in eSMP:

$$R_k(x) = \begin{cases} B_+(x), & \text{Case 1} \\ B(x), & \text{Case 2} \\ A(x), & \text{Case 3} \\ B_{i+}(x), & \text{Case 4} \quad i = 2,3,\cdots \end{cases} \tag{5}$$

3.2 Departure probabilities

To find the elements of the transition probability matrix, we need to count the number of tasks departing from the system in the time interval between two successive super-task arrivals. Therefore at first step, we need to calculate the probability of having k arrivals during the residence time of each state. Let $\mathcal{N}(B_+)$, $\mathcal{N}(B)$ and $\mathcal{N}(B_{i+})$, where $i = 2, 3, \ldots$, indicate the number of arrivals during time periods $B_+(x)$, $B(x)$ and $B_{i+}(x)$, respectively. Due to Poisson distribution of arrivals, we may define the following probabilities:

$$\begin{aligned}
\alpha_k &\triangleq \text{Prob}[\mathcal{N}(B_+) = k] = \int_0^\infty \frac{(\lambda x)^k}{k!} e^{-\lambda x} dB_+(x) \\
\beta_k &\triangleq \text{Prob}[\mathcal{N}(B) = k] = \int_0^\infty \frac{(\lambda x)^k}{k!} e^{-\lambda x} dB(x) \\
\delta_{ik} &\triangleq \text{Prob}[\mathcal{N}(B_{i+}) = k] = \int_0^\infty \frac{(\lambda x)^k}{k!} e^{-\lambda x} dB_{i+}(x)
\end{aligned} \tag{6}$$

We are also interested in the probability of having no arrivals; this will help us calculate the transition probabilities in aEMC.

$$\begin{aligned}
P_x &\triangleq \text{Prob}[\mathcal{N}(B_+) = 0] = \int_0^\infty e^{-\lambda x} dB_+(x) = B_+^*(\lambda) \\
P_y &\triangleq \text{Prob}[\mathcal{N}(B) = 0] = \int_0^\infty e^{-\lambda x} dB(x) = B^*(\lambda) \\
P_{ix} &\triangleq \text{Prob}[\mathcal{N}(B_{2+}) = 0] = \int_0^\infty e^{-\lambda x} dB_{i+}(x) = B_{i+}^*(\lambda) \\
P_{xy} &= P_x P_y
\end{aligned} \tag{7}$$

Note that $P_{1x} = P_x$. We may also define the probability of having no departure between two super-task arrivals. Let A be an exponential random variable with the parameter of λ, and let B_+ be a random variable which is distributed according to $B_+(x)$ (remaining service time). The probability of having no departures is

$$\begin{aligned}
P_z = \; & Prob\,[A < B_+] \\
= \; & \int_{x=0}^{\infty} P\{A < B_+ | B_+ = x\} dB_+(x) \\
= \; & \int_0^\infty (1 - e^{-\lambda x}) dB_+(x) \\
= \; & 1 - B_+^*(\lambda) = 1 - P_x
\end{aligned} \tag{8}$$

Using probabilities P_x, P_y, P_{xy}, P_{ix} and P_z we may define the transition probabilities for aEMC. To keep the model tractable, we consider that no single VM will experience more than three task departures between two successive super-task arrivals. Since the cloud center is assumed to have a large number of PMs and each of which is capable of running a high number of VMs, assuming three departures makes our performance model very close to real cloud centers. Note that, we model each VM in between two super-task arrivals and for such a period of time we consider the probability of having up to three departures which is highly unlikely under stable configurations (i.e. $\rho < 0.9$).

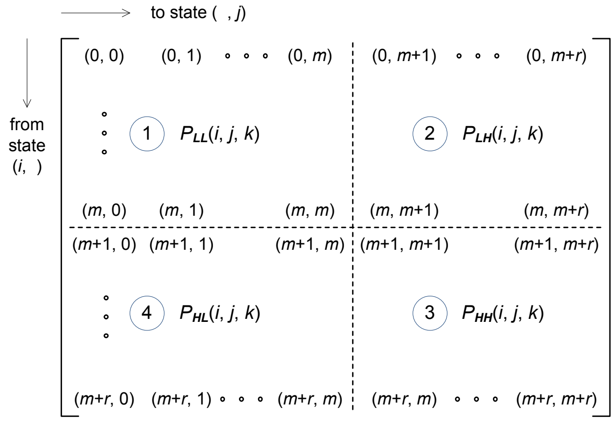


Fig. 7. One-step transition probability matrix: range of validity for $P(i, j, k)$ equations.

3.3 Transition matrix

The transition probabilities may be depicted in the form of a one-step transition matrix, as shown Fig. 7 where rows and columns correspond to the number of tasks in the system immediately before a super-task arrival (i) and immediately after the next super-task arrival (j), respectively. As can be seen, four operating regions may be distinguished, depending on whether the input queue is empty or non-empty before and after successive super-task arrivals. Each region has a distinct transition probability equation that depends on current state i, next state j, super-task size k, and number of departures between two super-task arrivals. Note that, in all regions, $P(i, j, k) = P_z$ if $i + k = j$.

3.3.1 Region 1

In this region, the input queue is empty and remains empty until next arrival, hence the transitions originate and terminate on the states labeled m or lower. Let us denote the number of tasks which depart from the system between two super-task arrivals as $w(k) = i + k - j$. For $i, j \le m$, the transition probability is

$$P_{LL}(i,j,k) = \begin{cases} \displaystyle\sum_{z=0}^{min(i,w(k))} \binom{i+k}{w(k)} P_x^{w(k)} (1-P_x)^j, & \text{if } i+k \le m \\ \displaystyle\sum_{z=i+k-m}^{min(i,w(k))} \binom{i}{z} P_x^z (1-P_x)^{i-z} \cdot \\ \binom{k}{w(k)-z} P_{xy}^{w(k)-z} (1-P_{xy})^{z-i+j}, & \text{if } i+k > m \end{cases}$$

3.3.2 Region 2

In this region, the queue is empty before the transition but non-empty afterward, hence $i \le m, j > m$. This means that the arriving super-task was queued because it could not be serviced immediately due to the insufficient number of idle servers. The corresponding transition probabilities are

$$P_{LH}(i,j,k) = \Pi_{s=1}^{w(k)} \left[(i - s + 1) P_{sx}\right] \cdot (1 - P_x)^{i - w(k)}$$

3.3.3 Number of idle servers

To calculate the transition probabilities for regions 3 and 4, we need to know the probability of having n idle servers out of m. Namely, due to the total rejection policy, a super-task may have to wait in the queue even when there are idle servers: this happens when the number of idle servers is smaller than the number of tasks in the super-task at the head of the queue. To calculate this probability, we shape the situation formally to the following scenario: consider a Poisson batch arrival process in which the batch size is a generally distributed random variable, and each arriving batch is stored in a finite queue. Storing the arrival batches in the queue will be continued until either the queue gets full or the last arrival batch can't fit in. If the queue size is t, the mean batch size is $\overline{g}$, and the maximum possible batch size is equal to *MBS*, what is the probability, $P_i(n)$, of having n unoccupied spaces in the queue? Unfortunately, this probability can't be computed analytically. To this end, we have simulated the queue size for different mean batch sizes, using the object-oriented Petri net-based simulation engine Artifex by RSoftDesign, Inc. [18]. We have fixed the queue size at $m = 200$, and ran the simulation experiment one million times; the resulting probability distribution is shown in Fig. 8.

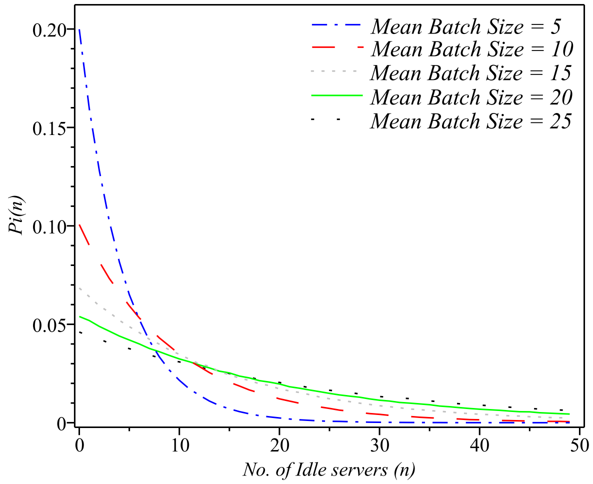


Fig. 8. Probability of having n idle servers, $P_i(n)$, for different mean batch sizes.

The shape indicates an exponential dependency so using a curve fitter tool [5], we have empirically found the parameters that give the best fit, as shown in Table 2, for different values of mean batch size. In all cases, the approximation error remains below 0.18%. This allows us to calculate the transition probabilities for regions 3 and 4 in the transition probability matrix.

TABLE 2
Parameters for optimum exponential curves ab^x.

	mean super-task size				
Param.	5	10	15	20	25
a	1.99E-01	1.00E-01	6.87E-02	5.40E-02	4.59E-02
b	8.00E-01	9.00E-01	9.33E-01	9.50E-01	9.60E-01

3.3.4 Region 3

Region 3 corresponds to the case where the queue is not empty before and after the arrivals, i.e., $i, j > m$. In this case, transitions start and terminate at states above m in Fig. 4, and the state transition probabilities can be computed as following product:

$$\begin{aligned}
&P_{HH}(i,j,k) \simeq \\
&\sum_{\psi=(m-MBS+1)}^{m} \sum_{s_1=\min(w(k),1)}^{\min(w(k),\psi)} \binom{\psi}{s_1} P_x^{s_1}(1-P_x)^{\psi-s_1} \cdot P_i(m-\psi)\times \\
&\sum_{\delta=max(0,m-\psi+s_1-MBS+1)}^{m-\psi+s_1} \sum_{s_2=\min(w(k)-s_1,1)}^{\min(\delta,w(k)-s_1)} \binom{\delta}{s_2} P_{2x}^{s_2}(1-P_{2x})^{\delta-s_2} \cdot P_i(m-\psi+s_1-\delta)\times \\
&\sum_{\phi=max(0,m-\psi+s_1-\delta+s_2-MBS+1)}^{m-\psi+s_1-\delta+s_2} \binom{\phi}{w(k)-s_1-s_2} \\
&\qquad P_{3x}^{w(k)-s_1-s_2}(1-P_{3x})^{\phi-w(k)+s_1+s_2} \cdot \\
&\qquad\qquad P_i(m-\psi+s_1-\delta+s_2-\phi)
\end{aligned} \tag{9}$$

3.3.5 Region 4

Finally, region 4 corresponds to the situation where the queue is non-empty at the time of the first arrival, but it is empty at the time of the next arrival: $i > m, j \leq m$. The transition probabilities for this region are

$$\begin{aligned}
&P_{HL}(i,j,k) \simeq \\
&\sum_{\psi=(m-MBS+1)}^{m} \sum_{s_1=\min(0,\psi-j)}^{\min(w(k),\psi)} \binom{\psi}{s_1} P_x^{s_1}(1-P_x)^{\psi-s_1} \cdot P_i(m-\psi)\times \\
&\sum_{\delta=max(0,m-\psi+s_1-MBS+1)}^{m-\psi+s_1} \sum_{s_2=\min(w(k)-s_1,\psi-j)}^{\min(\delta,w(k)-s_1)} \binom{\delta}{s_2} P_{2x}^{s_2}(1-P_{2x})^{\delta-s_2} \cdot P_i(m-\psi+s_1-\delta)\times \\
&\sum_{\phi=max(0,m-\psi+s_1-\delta+s_2-MBS+1)}^{m-\psi+s_1-\delta+s_2} \binom{\phi}{w(k)-s_1-s_2} \\
&\qquad P_{3x}^{w(k)-s_1-s_2}(1-P_{3x})^{\phi-w(k)+s_1+s_2} \cdot \\
&\qquad\qquad P_i(m-\psi+s_1-\delta+s_2-\phi)
\end{aligned} \tag{10}$$

3.4 Equilibrium balance equations

We are now ready to set the balance equations for the transition matrix:

$$\pi_i \; = \sum_{j=\max[0,i-MBS]}^{m+r} \pi_j p_{ji} \;, \quad 0 \le i \le m+r \tag{11}$$

augmented by the normalization equation $\sum_{i=0}^{m+r} \pi_i = 1$. The total number of equations is $m+r+2$, but there are only $m+r+1$ variables $[\pi_0, \pi_1, \pi_2, \ldots, \pi_{m+r}]$. Therefore, we need to remove one of the equations in order to obtain the unique equilibrium solution (which exists if the corresponding Markov chain were *ergodic*). A wise choice would be the last equation in (11) which holds the minimum amount of information about the system in comparison with the others. Here, the steady state balance equations can't be solved in closed form, hence we must resort to a numerical solution.

3.5 Distribution of the number of tasks in a PM

Once we obtain the steady state probabilities we are able to establish the PGF for the number of tasks in the PM at the time of a super-task arrival:

$$\Pi(z) = \sum_{k=0}^{m+r} \pi_z z^k.$$

Due to batch arrivals, the PASTA property doesn't hold; thus, the PGF $\Pi(z)$ for the distribution of the number of tasks in the PM at an arbitrary time is not the same as the PGF $P(z)$ for the distribution of the number of tasks in the PM at the time of a super-task arrival. To obtain the former, we employ a two-step approximation technique with an embedded semi-Markov process and an approximate embedded Markov process, as explained above. aEMP imitates the original process but it will be updated only at the moments of super-task arrivals. Let $H_k(x)$ be the CDF of the residence time that aEMP stays in the state k:

$$\begin{aligned} &H_k(x) \triangleq Prob[t_{n+1} - t_n \le x \mid q_n = k] = 1 - e^{\lambda x}, \\ &\quad k = 0, 1, 2, ..., m+r \end{aligned} \tag{12}$$

which in our model does not depend on n. The mean residence time in state k is

$$\overline{h}_k = \int_0^\infty [1 - H_k(x)] dx = \frac{1}{\lambda}, \quad k = 0, 1, 2, ..., m+r.$$

and the steady-state distribution in aEMP is given by [19]

$$p_k^{sm} = \frac{\pi_k \overline{h}_k}{\sum_{j=0}^{m+r} \pi_j \overline{h}_j} = \frac{\pi_k}{\lambda \sum_{j=0}^{m+r} \pi_j 1/\lambda} = \pi_k \tag{13}$$

where $\{\pi_k; \;\; k = 0, 1, \; \ldots \;, m+r\}$ is the distribution probability at aEMC. So the steady-state probability of aEMP is identical with the embedded aEMC. We now define the CDF of the elapsed time from the most recent observing point looking form an arbitrary time by

$$H_k^-(y) = \frac{1}{\overline{h}_k} \int_0^y [1 - H_k(x)] dx, k = 0 \,..\, m+r \tag{14}$$

The arbitrary-time distribution is given by

$$p_i = \sum_{j=i}^{m+r} p_j^{sm} \int_0^\infty P^+ \, dH_j^-(y) = \sum_{j=i}^{m+r} \pi_j P(j, i, 0).$$

in which P^+ is the probability of changes in y that bring the state from j to i. The PGF of the number of tasks in the PM is given by

$$P(z) = \sum_{i=0}^{m+r} p_i z^i \tag{15}$$

Mean number of tasks in the PM, then, obtained as $\overline{p} = P^{'}(1)$. We also obtained the distribution of response and waiting time; due to space limitation, the results are presented in Appendix B.

3.6 Blocking probability

Since arrivals are independent of buffer state and the distribution of number of tasks in the PM was obtained, we are able to directly calculate the blocking probability of a super-task in a PM with buffer size of r:

$$PM_{BP}(r) \; = \sum_{k=0}^{MBS-1} \left[\sum_{i=0}^{MBS} p_{m+r-i-k}(1 - G(i)) \right] \cdot P_i(k).$$

The buffer size, r_ϵ, required to keep the blocking probability below the certain value, ϵ, is:

$$r_\epsilon \; = \; \{r \ge 0 \mid PM_{BP}(r) \le \epsilon \text{ and } PM_{BP}(r-1) > \epsilon\}.$$

Therefore, the blocking probability of a cloud center with M PMs is $BP \; = \; (PM_{BP})^M$.

3.7 Probability of immediate service

We are also interested in the probability that a super-task will get service immediately upon arrival, without any queuing. In this case, the response time would be equal to the service time:

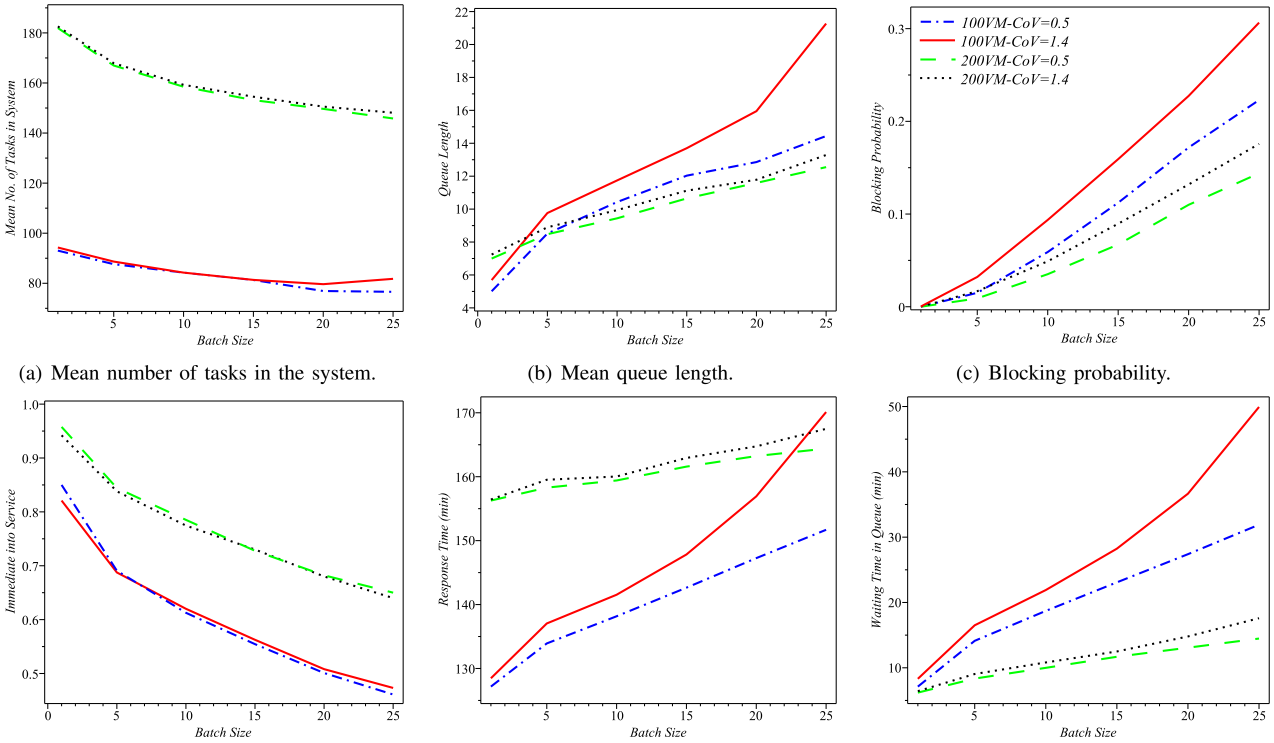
$$\begin{aligned} PM_{pis} \; = & \sum_{k=0}^{MBS-1} \\ & \left[\sum_{j=0}^{m-k-MBS} p_j + \sum_{i=m-k-MBS+1}^{m-k-1} p_i G(m-k-i) \right] \cdot P_i(k) \end{aligned}$$

Thus, the probability of immediate service for a cloud center having M PMs is given by:

$$P_{imm} = 1 - (1 - PM_{pis})^M \tag{16}$$

4 Numerical Validation

The resulting balance equations of analytical model have been solved using Maple 15 from Maplesoft, Inc. [15]. To validate the analytical solution, we have also built a discrete event simulator of the cloud center using the Artifex engine [18]. Due to clarity and space limitation we present all the simulation results in the supplemental of the paper. We have configured a cloud center with $M = 1000$ PMs under two degrees of virtualization: 100 and 200 VMs per



(a) Mean number of tasks in the system.

(b) Mean queue length.

(c) Blocking probability.

(d) Probability of getting immediate service.

(e) Response time.

(f) Waiting time in the queue.

Fig. 9. Performance measures as function of mean batch size of super-tasks.

PM. We refer them as *moderate* and *heavy* configuration respectively. Latest VMmark results [22], show that only a few types of current state-of-the-art PMs can run up to 200 VMs with an acceptable performance while most of them are optimized to run up to 100 VMs.

The input queue size is set to $r = 300$ tasks. Traffic intensity is set to $\rho = 0.85$, which may seem too high but could easily be reached in private cloud centers or public centers of small/medium size providers. The distribution of batch size is assumed to be truncated Geometric, with maximum batch size set to $MBS = \frac{3}{g} + 1$ in which g is the probability of success in the Geometric distribution. Note that real cloud centers may impose an upper limit for the number of servers requested by a customer. Task service time is assumed to have Gamma distribution, which is chosen because it allows the coefficient of variation to be set independently of the mean value. The mean value depends on the degree of virtualization: on account of (2), we set the task service time as 120 and 150 minute for moderate and heavy virtualization, respectively. Two values are used for the coefficient of variation: $CoV = 0.5$ and 1.4, which give rise to hypo- and hyper-exponentially distributed service times, respectively.

Diagrams in Fig. 9 show the analytical results for various performance indicators vs. super-task size. The corresponding simulation results are presented in Appendix C. As can be seen, the queue size increases with mean batch size whereas the mean number of tasks in the system decreases with it. The moderate configuration always outperforms the heavy configuration for single request arrivals (i.e., super-tasks with size one), as opposed to bigger super-tasks. This may be attributed to the total rejection policy, which lets some servers to remain idle even though there exist some super-tasks (potentially big super-tasks) in the queue. As might be expected, the blocking probability increases as the size of batches increases, while the probability of immediate service (which might be termed availability) decreases almost linearly. Nevertheless, the probability of immediate service is above 0.5 in the observed range of mean batch sizes, which means that more than 50% of user requests will be serviced immediately upon arrival, even though the total traffic is $\rho = 0.85$, which is rather high.

We have also computed the system response time and queue waiting time for super-tasks. Note that, because of the total rejection policy explained above, the response and waiting times for super-tasks are identical for all individual tasks within the super-task. As depicted in Fig. 9(e), response time—which is the sum of waiting time and service time— and waiting time, Fig. 9(f), slowly increase with an increase in mean batch size. This may be attributed to the fact that larger mean batch size leads to larger super-tasks, which are more likely to remain longer in the queue before the number of idle servers becomes sufficiently high to accommodate all of its individual tasks. Also, heavy configuration imposes shorter waiting time in queue on super-tasks. In other words, waiting time in queue is heavily influenced by admission policy rather than overhead of virtualization. However, response time is more sensitive to the degree of virtualization. It can also be seen that CoV of service time plays an important role on total response time for large super-tasks (i.e., size 25). In all of the experiments described above, the agreement between analytical and simulation results is very good, which confirms the validity of the proposed approximation procedure using eSMP and aEMP/aEMC.

Not unexpectedly, performance at hyper-exponentially distributed service times ($CoV = 1.4$) is worse in general than that at hypo-exponential ones ($CoV = 0.5$). To obtain further insight into cloud center performance, we have calculated the higher moments of response time which are shown in Appendix D. We have also examined two types of homogenization, based on coefficient of variation of task service time and super-task size, the result of which presented in Appendix E.

5 Conclusions

In this paper, we have described an analytical model for performance evaluation of a highly virtualized cloud computing center under Poisson batch arrivals with generally distributed task sizes and generally distributed task service times, with additional correction for performance deterioration under heavy workload. The model is based on a two-stage approximation technique where the original non-Markovian process is first modeled with an embedded semi-Markov process, which is then modeled by an approximate embedded Markov process but only at the time instants of super-task arrivals. This technique provides accurate computation of important performance indicators such as the mean number of tasks in the system, queue length, mean response and waiting time, blocking probability and the probability of immediate service.

Our results show that admitting requests with widely varying service times to a unified, homogeneous cloud center may result in longer waiting time and lower probability of getting immediate service. Both performance metrics may be improved by request homogenization, obtained by partitioning the incoming super-tasks on the basis of super-task size and/or coefficient of variation of task service times and servicing the resulting sub-streams by separate cloud sub-centers (or sub-clouds). This simple approach thus offers distinct benefits for both cloud users and cloud providers.

References

[1] Amazon Elastic Compute Cloud, *User Guide*, Amazon Web Service LLC or its affiliate, Aug. 2012. [Online]. Available: http://aws.amazon.com/documentation/ec2

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

[3] 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.

[4] A. Corral-Ruiz, F. Cruz-Perez, and G. Hernandez-Valdez, "Teletraffic model for the performance evaluation of cellular networks with hyper-erlang distributed cell dwell time," in *71st IEEE Vehicular Technology Conference (VTC 2010-Spring)*, May. 2010, pp. 1–6.

[5] CurveExpert, "Curveexpert professional 1.1.0," Website, Mar. 2011, http://www.curveexpert.net.

[6] 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.

[7] 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*, 2010, pp. 125–132.

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

[9] Hewlett-Packard Development Company, Inc., "An overview of the VMmark benchmark on HP Proliant servers and server blades," white paper, May 2012, ftp://ftp.compaq.com/pub/products/servers/benchmarks/VMmark_Overview.pdf.

[10] H. Khazaei, J. Mišić, and V. B. Mišić;, "A fine-grained performance model of cloud computing centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 99, no. PrePrints, p. 1, 2012.

[11] 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.

[12] H. Khazaei, J. Mišić, V. B. Mišić, and N. Beigi Mohammadi, "Availability analysis of cloud computing centers," in *Globecom 2012 - Communications Software, Services and Multimedia Symposium (GC12 CSSM)*, Anaheim, CA, USA, Dec. 2012.

[13] H. Khazaei, J. Mišić, V. B. Mišić, and S. Rashwand, "Analysis of a pool management scheme for cloud computing centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 99, no. PrePrints, 2012.

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

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

[16] A. Patrizio, "IDC sees cloud market maturing quickly," *Datamation*, Mar. 2011. [Online]. Available: http://itmanagement.earthweb.com/netsys/article.php/3870016/IDC-Sees-Cloud-Market-Maturing-Quickly.htm

[17] 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.

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

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

[20] 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.

[21] VMware, Inc., "VMmark: A Scalable Benchmark for Virtualized Systems," Technical Report, Sept. 2006, http://www.vmware.com/pdf/vmmark_intro.pdf.

[22] ——, "VMware VMmark 2.0 benchmark results," Website, Sept. 2012, http://www.vmware.com/a/vmmark/.

[23] L. Wang, G. von Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu, "Cloud computing: a perspective study," *New Generation Computing*, vol. 28, pp. 137–146, 2010.

[24] 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.

[25] 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.

[26] 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.