A Stochastic Model to Investigate Data Center Performance and QoS in IaaS Cloud Computing Systems

Dario Bruneo, *Member, IEEE*

Abstract—Cloud data center management is a key problem due to the numerous and heterogeneous strategies that can be applied, ranging from the VM placement to the federation with other clouds. Performance evaluation of Cloud Computing infrastructures is required to predict and quantify the cost-benefit of a strategy portfolio and the corresponding Quality of Service (QoS) experienced by users. Such analyses are not feasible by simulation or on-the-field experimentation, due to the great number of parameters that have to be investigated. In this paper, we present an analytical model, based on Stochastic Reward Nets (SRNs), that is both scalable to model systems composed of thousands of resources and flexible to represent different policies and cloud-specific strategies. Several performance metrics are defined and evaluated to analyze the behavior of a Cloud data center: utilization, availability, waiting time, and responsiveness. A resiliency analysis is also provided to take into account load bursts. Finally, a general approach is presented that, starting from the concept of system capacity, can help system managers to opportunely set the data center parameters under different working conditions.

Index Terms—Cloud computing, stochastic reward nets, cloud-oriented performance metrics, resiliency, responsiveness.

1 Introduction

Cloud Computing is a promising technology able to strongly modify the way computing and storage resources will be accessed in the near future [1]. Through the provision of on demand access to virtual resources available on the Internet, cloud systems offer services at three different levels: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). In particular, IaaS clouds provide users with computational resources in the form of virtual machine (VM) instances deployed in the provider data center, while PaaS and SaaS clouds offer services in terms of specific solution stacks and application software suites, respectively.

In order to integrate business requirements and application level needs, in terms of Quality of Service (QoS), cloud service provisioning is regulated by Service Level Agreements (SLAs): contracts between clients and providers that express the price for a service, the QoS levels required during the service provisioning, and the penalties associated with the SLA violations. In such a context, performance evaluation plays a key role allowing system managers to evaluate the effects of different resource management strategies on the data center functioning and to predict the corresponding costs/benefits.

Cloud systems differ from traditional distributed systems. First of all, they are characterized by a very large number of resources that can span different administrative domains. Moreover, the high level of resource abstraction allows to implement particular resource management techniques such as VM multiplexing [2] or VM live migration [3] that, even if transparent to final users, have to be considered in the design of performance models in order to accurately understand the system behavior. Finally, different clouds, belonging to the same or to different organizations, can dynamically join each other to achieve a common goal, usually represented by the optimization of resources utilization. This mechanism, referred to as cloud *federation* [4], allows to provide and release resources on-demand thus providing elastic capabilities to the whole infrastructure.

For these reasons, typical performance evaluation approaches such as simulation or on-the-field measurements can not be easily adopted. Simulation [5], [6] does not allow to conduct comprehensive analyses of the system performance due to the great number of parameters that have to be investigated. On-the-field experiments [7], [8] are mainly focused on the offered QoS, they are based on a black box approach that makes difficult to correlate obtained data to the internal resource management strategies implemented by the system provider. On the contrary, analytical techniques [9], [10] represent a good candidate thanks to the limited solution cost of their associated models. However, to accurately represent a cloud system an analytical model has to be:

- Scalable. In order to deal with very large systems composed of hundreds or thousands of resources.
- Flexible. Allowing to easily implement different strategies and policies and to represent different working conditions.

In this paper, we present a stochastic model, based on

• *The author is with the Dipartimento di Ingegneria Civile, Informatica, Edile, Ambientale e Matematica Applicata (DICIEAMA), Università di Messina, Contrada di Dio, 98166 Messina, Italy.*
E-mail: dbruneo@unime.it

Stochastic Reward Nets (SRNs) [11], that exhibits the above mentioned features allowing to capture the key concepts of an IaaS cloud system. The proposed model is scalable enough to represent systems composed of thousands of resources and it makes possible to represent both physical and virtual resources exploiting cloud specific concepts such as the infrastructure elasticity. With respect to the existing literature, the innovative aspect of the present work is that a generic and comprehensive view of a cloud system is presented. Low level details, such as VM multiplexing, are easily integrated with cloud based actions such as federation, allowing to investigate different mixed strategies. An exhaustive set of performance metrics are defined regarding both the system provider (e.g., utilization) and the final users (e.g., responsiveness). Moreover, different working conditions are investigated and a resiliency analysis is provided to take into account the effects of load bursts. Finally, to provide a fair comparison among different resource management strategies, also taking into account the system elasticity, a performance evaluation approach is described. Such an approach, based on the concept of system capacity, presents a holistic view of a cloud system and it allows system managers to study the better solution with respect to an established goal and to opportunely set the system parameters.

The paper is organized as follows. In Section 2 we formulate the problem and we describe the proposed analytical model. In Section 3 and 4 we illustrate the steady state and transient solutions of the model and the corresponding performance metrics that can be defined. In Section 5 we discuss some numerical results that can be obtained through the model solution also presenting an approach to assess the performance of a cloud system, while in Section 6 we summarize our results providing insights for future work. Finally, in the supplementary file, available online, we provide a critical overview of the state of the art and a numerical comparison with one of the referred works, as well as the needed background and some in-depth discussions.

2 The Analytical Model

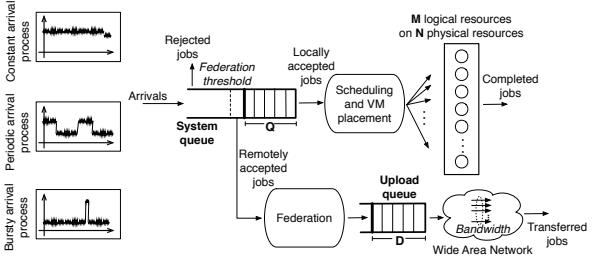


Fig. 1. An IaaS cloud system with federation.

We consider an IaaS cloud system composed of N physical resources (see Fig. 1). Job requests (in terms of VM instantiation requests) are enqueued in the *system queue*. Such a queue has a finite size Q, once its limit is reached further requests are rejected. The system queue is managed according to a FIFO scheduling policy. When a resource is available a job is accepted and the corresponding VM is instantiated. We assume that the instantiation time is negligible and that the service time (i.e., the time needed to execute a job) is exponentially distributed with mean $1/\mu$.

According to the VM multiplexing technique [2], [12] the cloud system can provide a number M of logical resources greater than N. In this case, multiple VMs can be allocated in the same physical machine (PM), e.g., a core in a multi-core architecture. Multiple VMs sharing the same PM can incur in a reduction of the performance mainly due to I/O interference between VMs [13]. We define the degradation factor d (≥ 0) as the percentage increase in the expected service time experienced by a VM when multiplexed with another VM. The performance degradation of multiplexed VMs depends on the multiplexing technique and on the VM placement strategy [14], [15], [16]. We assume that, in order to reduce the degradation and to obtain a fair distribution of VMs, the system is able to optimally balance the load among the PMs with respect to the resources required by VMs (e.g., trying to multiplex CPU-bound VMs only with I/O-bound VMs), thus reaching a homogeneous degradation factor. Then, indicating with $T = 1/\mu$ the expected service time of a VM in isolation, we can derive the expected time needed to execute two multiplexed VMs as $T_2 = T \cdot (1 + d)$. In general we can express the expected execution time of i multiplexed VMs as:

$$T_i = T \cdot (1 + d)^{i-1}. \quad (1)$$

System managers can obtain an estimation of parameter d by means of theoretical studies or statistical observations.

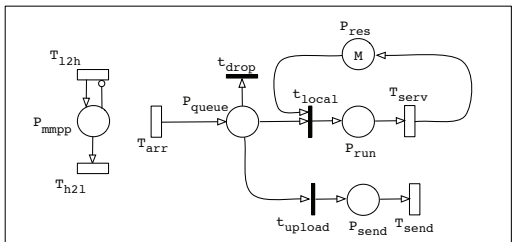
Cloud federation [4] allows the system to use, in particular situations, the resources offered by other public cloud systems through a sharing and paying model. In this way, elastic capabilities can be exploited in order to respond to particular load conditions. Job requests can be redirected to other clouds by transferring the corresponding VM disk images through the network. With respect to the federation technique we make the following assumptions:

- A job is redirected only if it arrives when the system queue is full.
- Federate clouds are characterized by an availability a_f.
- Federate clouds are also characterized by a quality level q_f ($0 < q_f \leq 1$) that determines the QoS reached by a request, in terms of expected service time (i.e., a VM that needs a time $T = 1/\mu$ to accomplish it works will experience an execution time $T_f = 1/(q_f \cdot \mu) \geq T$).
- A redirected job is inserted in the *upload queue* waiting for the VM transfer completion (see Fig. 1).
- There is a maximum number of concurrent redirected jobs (elasticity level) i.e., the upload queue has a finite size equal to D.
- The network bandwidth allows to transmit up to k VMs in parallel.
- The time needed to transfer a VM disk image is exponentially distributed with mean $1/\eta$.

Finally, we respect to the arrival process we will investigate three different scenarios. In the first one (*Constant arrival process*) we assume the arrival process be a homogeneous Poisson process with rate λ. However, large-scale distributed systems with thousands of users, such as cloud systems, could exhibit self-similarity/long-range dependence with respect to the arrival process [17]. For these reasons, in order to take into account the dependencies of the job arrival rate on both the days of a week and the hours of a day, in the second scenario (*Periodic arrival process*) we also choose to model the job arrival process as a Markov Modulated Poisson Process (MMPP). In particular, we will refer to an $MMPP(\lambda_h, \lambda_l, \lambda_{h2l}, \lambda_{l2h})$, where λ_h and λ_l represent the expected arrival rate in high and low load conditions while $1/\lambda_{h2l}$ and $1/\lambda_{l2h}$ represent the expected duration of the two load conditions. The last scenario (*Bursty arrival process*) takes into account the presence of a burst whit fixed and short duration and it will be used in order to investigate the system resiliency.

To capture the main features of a typical IaaS cloud we make use of SRNs [11]. SRNs are an extension of Generalized Stochastic Petri Nets (GSPNs) [18] that allow us to associate reward rates with the marking (i.e., the distribution of tokens in the various places) [19]. In the reminder of the paper we will use the notation $P^{\#}$ to refer to the number of token in place P. Moreover, in the function definitions we adopt a C-like syntax using the ternary operator $(?:)$ instead of the $if-else$ construct. We give a formal overview of the SRN notation in the Appendix A of the supplementary file.

The proposed SRN cloud performance model is depicted in Fig. 2. Transition T_{arr} models the arrival process. If the



Transition	Guard function
t_{drop}	$(P^{\#}_{res} = 0 \ \& \ P^{\#}_{queue} > Q) \ ? \ 1 : 0$
t_{upload}	$(P^{\#}_{res} = 0 \ \& \ P^{\#}_{queue} > Q \ \& \ P^{\#}_{send} < D) \ ? \ 1 : 0$

Transition	Weight
t_{drop}	$1-a_f$
t_{upload}	a_f

Fig. 2. The proposed SRN cloud performance model.

Constant arrival process is taken into account, we can characterize transition T_{arr} with an exponentially distributed firing time with mean $1/\lambda$. On the contrary, to model the Periodic arrival process, the rate of transition T_{arr} can be computed as a function of the number of tokens in place P_{mmpp}, thus obtaining the two different load conditions: low load (with rate equal to λ_l) when $P^{\#}_{mmpp} = 0$ and high load (with rate equal to λ_h) when $P^{\#}_{mmpp} = 1$. The alternation between low and high load conditions is modeled by exponentially distributed transitions T_{h2l} and T_{l2h} with rates λ_{h2l} and λ_{l2h}, respectively. The technique adopted to model the Bursty arrival process will be discussed later in Section 4.

The system queue is modeled through place P_{queue}. A token in this place represents a job waiting in the queue. When the number of tokens in P_{queue} is greater than the queue size Q, transition t_{drop} is enabled (see the corresponding guard function in Fig. 2) thus modeling the rejection of a request.

Cloud resources are modeled by tokens in place P_{res}. Such tokens represent the M logical resources offered by the system. When a resource is available and there are pending requests, a token is removed (through transition t_{local}) from the system queue and is added to place P_{run}. If VM multiplexing is not allowed ($M = N$) service time is modeled through transition T_{serv} with firing rate equal to μ while the VM parallel execution (one for each PM) is modeled by setting transition T_{serv} with the infinite server semantic [18] in order to increase its firing rate in proportion to the number of tokens in the enabling place P_{run}. More formally, indicating with $R_{serv}(P^{\#}_{run})$ the marking dependent rate of transition T_{serv} we have:

$$R_{serv}(P^{\#}_{run}) = P^{\#}_{run} \cdot \mu \ , \ 1 < P^{\#}_{run} \leq N. \tag{2}$$

Insights on how to relax the assumption on exponentially distributed service time can be found in Section 9 of the supplementary file.

2.1 Modeling cloud federation

Federation with other clouds is modeled allowing tokens in place P_{queue} to be moved, through transition t_{upload}, in the upload queue represented by place P_{send}. In accordance with the assumptions made before, transition t_{upload} is enabled only if the number of tokens in place P_{queue} is grater than Q and the number of tokens in place P_{send} is less than D. Moreover, in order to take into account the federated cloud availability, concurrent enabled transition t_{upload} and t_{drop} are managed by setting their weights with the values a_f and $1-a_f$, respectively. In this way, there will be a probability equal to a_f to send an incoming request to federated clouds and a probability equal to $1-a_f$ to drop it. The VM transferring time is modeled by transition T_{send}. In order to take into account the parallel transferring with a limited network bandwidth, the rate of transition T_{send} is designed as follows. Until the bandwidth is not saturated (i.e., the number of VMs in the upload queue is less than k) the transition rate is proportional to the number of waiting VMs. When the number of VMs in the upload queue is greater than k the transition rate is no more increased and it remains proportional to k. Indicating with $R_{send}(P^{\#}_{send})$ the marking dependent rate of transition T_{send} we obtain:

$$R_{send}(P^{\#}_{send}) = \begin{cases} P^{\#}_{send} \cdot \eta & if \quad 0 < P^{\#}_{send} \leq k \\ k \cdot \eta & if \quad k < P^{\#}_{send} \leq D \end{cases}. \tag{3}$$

Insights on how to model private cloud federations can be found in Section 10 of the supplementary file.

2.2 Modeling VM multiplexing

When VM multiplexing is allowed, the number of running VMs can be greater than N, i.e., $0 \leq P^{\#}_{run} \leq M$ and each PM can be loaded with more than one VM. Assuming an optimal scheduling algorithm able to balance the load among the N PMs, the maximum multiplexing level l reached by each PM (i.e., the maximum number of VMs running on a single PM) is given by:

$$l = \left\lceil P^{\#}_{run}/N \right\rceil . \tag{4}$$

The set J (with cardinality $|J| = P^{\#}_{run}$) of the instantiated VMs can be partitioned into two sets J_l and J_{l-1} (with $J = J_l \cup J_{l-1}$) that correspond to the set of VMs running with a multiplexing level equal to l and the set of VMs running with a multiplexing level equal to $l-1$, respectively. The cardinality of such sets can be obtained as:

$$|J_l| = l \cdot \left[P^{\#}_{run} - (l-1) \cdot N \right] \tag{5}$$

$$|J_{l-1}| = (l-1) \cdot \left[l \cdot N - P^{\#}_{run} \right] . \tag{6}$$

To model such a configuration, we need to calculate the equivalent rate of transition T_{serv} taking into account the parallel execution of the VMs and the performance degradation factor d. We start calculating the expected execution time $\overline{T}_{P^{\#}_{run}}$ averaged among all the $P^{\#}_{run}$ running VMs:

$$\overline{T}_{P^{\#}_{run}} = \frac{|J_l| \cdot T_l + |J_{l-1}| \cdot T_{l-1}}{P^{\#}_{run}} \tag{7}$$

where T_l and T_{l-1} can be computed by eq. (1).

Then, in order to also consider the VM parallel execution, in accordance to the infinite server semantic, the marking dependent rate of transition $T_{service}$ when $P^{\#}_{run}$ VMs are running on N PMs is given by multiplying the number of running VMs by the average execution rate given by the inverse of eq. (7):

$$\begin{aligned} R_{serv}(P^{\#}_{run}) &= P^{\#}_{run} \cdot \frac{1}{\overline{T}_{P^{\#}_{run}}} \\ &= \frac{(P^{\#}_{run})^2}{|J_l| \cdot T_l + |J_{l-1}| \cdot T_{l-1}} \text{ , } 1 < P^{\#}_{run} \leq M. \end{aligned} \tag{8}$$

It can be noticed that, when $1 < P^{\#}_{run} \leq N$, eq. (8) is equivalent to eq. (2).

For an illustrative example of the multiplexing strategy implementation, refer to the Appendix B of the supplementary file.

2.3 Understanding the model complexity

In order to respect the scalability requirement of the proposed model, we need to analyze its complexity. In particular, we are interested in the analysis of the state space cardinality that is the parameter that mainly influences the performance of the numerical solution techniques.

The state space $\mathcal{S}$ of the model is given by the set of all its tangible markings [18]. Considering the SRN of Fig. 2 and the corresponding guard functions we can make some considerations on the token distributions. Tokens in places P_{queue}, P_{res}, and P_{run} are governed by the following rules:

$$\begin{cases} P^{\#}_{res} + P^{\#}_{run} = M \\ P^{\#}_{queue} \leq Q \\ P^{\#}_{res} > 0 \Rightarrow P^{\#}_{queue} = 0 \\ P^{\#}_{queue} > 0 \Rightarrow P^{\#}_{run} = M \end{cases}$$

giving rise to a number of possible combinations of the token distribution in such places equal to $M+Q+1$. Each of these $M+Q+1$ combinations can be obtained with zero or one token in place P_{mmpp} and with a number of tokens in place P_{send} ranging from 0 to D.

The cardinality of the state space $\mathcal{S}$ is then given by:

$$|\mathcal{S}| = 2 \cdot (D+1) \cdot (M+Q+1) \tag{9}$$

and the model complexity is $O(D \cdot (M+Q))$. However, being normally $M \gg D, Q$ we can state that the model complexity grows linearly with the number of logical resources of the cloud system under exam and we can definitely assert that the model is scalable and that it is also suitable for on-line performance analyses.

3 Steady State Analysis

SRNs allow us to define reward functions that can be associated to a particular state of the model in order to evaluate the performance level reached by the system during the sojourn in that state [11].

Let $\{X(t), t \geq 0\}$ be the stochastic process (with finite state space $\mathcal{S}$) modeling the evolution in time of the SRN of Fig. 2 and r ($r : \mathcal{S} \rightarrow \Re$) a reward function. We indicate with $r_{X(t)}$ the stochastic process representing the system reward rate at time t. By solving the model, we can obtain the probabilities π_i of being in marking $i \in \mathcal{S}$ at steady state and the corresponding reward $r(i)$ normally expressed as r_i. The transient solution of the model allows us also to obtain the corresponding measures at time t ($\pi_i(t)$ and $r_i(t)$). The expected reward rate at steady state $E[r_X]$ is then given by $E[r_X] = \sum_{i \in \mathcal{S}} r_i \cdot \pi_i$.

In the following we are interested in performance metrics able to characterize the system behavior from both the provider and the user point-of-views. Such metrics will help system designer to size and manage the cloud data center and they will also be determinant in the SLA definitions.

Utilization. The expected data center utilization U can be computed as the ratio between the number of physical resources used at steady state and the total number N of physical resources. Defining the following reward function:

$$r^u = P^{\#}_{run} \tag{10}$$

and considering the presence of virtual resources it can be computed as:

$$U = min\left\{1, E[r^u_X]/N\right\} . \tag{11}$$

Availability. The expected system availability A can be defined as the steady state probability that the system is able to accept a request. It can be computed defining two reward functions. One that checks when the drop condition is satisfied:

$$r^{d1} = (P^{\#}_{run} = M \ \& \ P^{\#}_{queue} = Q \ \& \ P^{\#}_{send} = D) \ ? \ 1 : 0 \quad (12)$$

and one that checks when a request is dropped due to the federated cloud unavailability:

$$r^{d2} = (P^{\#}_{run} = M \ \& \ P^{\#}_{queue} = Q \ \& \ P^{\#}_{send} < D) \ ? \ 1 - a_f : 0. \quad (13)$$

We obtain:

$$A = 1 - (E[r^{d1}_X] + E[r^{d2}_X]). \quad (14)$$

Waiting time. It is the expected time W spent by users in the queue (either the system or the upload queue). Using Little's law and computing the expected number of users in the queues through the following reward functions:

$$r^{q1} = P^{\#}_{queue} \quad (15)$$

$$r^{q2} = P^{\#}_{send} \quad (16)$$

we can derive [1] the waiting time expression as:

$$W = \frac{E[r^{q1}_X] + E[r^{q2}_X]}{Thr(T_{arr}) \cdot A} \quad (17)$$

where $Thr(T_{arr})$ is the expected throughput of transition T_{arr}.

Service time. It is the expected time S needed to execute a VM and it can be obtained as a function of the time S_l needed to locally execute a VM and the time S_r needed to execute a VM into the federated clouds. S_l can be computed as the ratio between the expected number of running VMs and the expected throughput of transition T_{serv}:

$$S_l = \frac{E[r^u_X]}{Thr(T_{serv})}. \quad (18)$$

When the VM multiplexing is not allowed such a value is equal to $1/\mu$ (see eqs. (2) and (10)). S_r depends on the federated cloud quality level (q_f) and it is given by:

$$S_r = \frac{1}{\mu \cdot q_f}. \quad (19)$$

In order to obtain the expression of S, the terms S_l and S_r have to be weighted summed with respect to the conditional probability of being locally or remotely accepted given that the request is not dropped. Defining the reward functions that allow to calculate the expected probabilities to be locally or remotely accepted:

$$r^{l} = (P^{\#}_{run} < M \mid P^{\#}_{queue} < Q) \ ? \ 1 : 0 \quad (20)$$

$$r^{r} = (P^{\#}_{run} = M \ \& \ P^{\#}_{queue} = Q \ \& \ P^{\#}_{send} < D) \ ? \ a_f : 0 \quad (21)$$

1. The proof of eq. (17) is provided in the Appendix C of the supplementary file.

we can derive the following expression:

$$S = S_l \cdot \frac{E[r^l_X]}{E[r^l_X] + E[r^r_X]} + S_r \cdot \frac{E[r^r_X]}{E[r^l_X] + E[r^r_X]}. \quad (22)$$

Responsiveness. It is the steady state probability R that the system is able to accept a request within a given time deadline τ. The computation of such a parameter requires the knowledge of the waiting time cumulative distribution function (CDF).

To this end it is possible to apply the *tagged customer* technique [20], [21] by modifying the SRN model in order to isolate the behavior of a single user request u and to observe its movements through the system. In the tagged

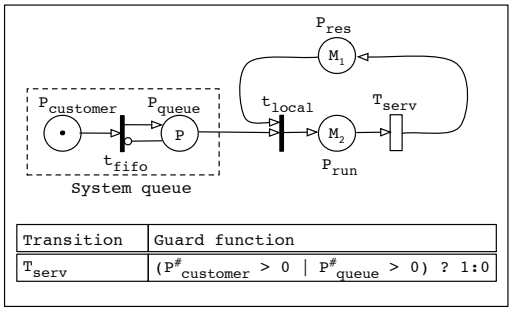


Fig. 3. The SRN tagged customer model used in the responsiveness evaluation.

customer model shown in Fig. 3, the system queue is modeled through two places. Place $P_{customer}$ contains a single token that represents the arrival of request u. The P tokens initially present in place P_{queue} represent the number of requests still waiting in the queue when u arrives, while the M_1 and M_2 tokens initially present in places P_{res} and P_{run} represent the corresponding system status. The FIFO policy is modeled through transition t_{fifo} that is enabled only when place P_{queue} is empty. Transitions t_{local} and T_{serv} behave as in the model of Fig. 2.

Fixing an initial marking m_0 (i.e., assigning a value to P, M_1, and M_2) that represents the conditions observed by request u when it is inserted in the queue, we can study the system temporal evolution. The tagged customer model contains absorbing markings (identified by the guard function of transition T_{serv}) and then it can be solved to compute the CDF giving the probability that an absorbing marking has been reached [19]. In our case, such CDF corresponds to the waiting time CDF $W^{m_0}(t)$ experienced by request u in the particular system condition expressed by initial marking m_0. In fact, an absorbing marking is reached when the queue becomes empty (i.e.,$P^{\#}_{customer} = P^{\#}_{queue} = 0$) and then immediately after the request u has been accepted in the system.

More formally, representing[2] a generic marking of the tagged customer model as a 4-vector

2. In the following the superscript *tc* will be used to refer to the tagged customer model while the superscript *cp* will be used to refer to the cloud performance model.

$[P^{\#}_{customer}, P^{\#}_{queue}, P^{\#}_{res}, P^{\#}_{run}]^{tc}$ we can define the set of absorbing markings as:

$$\mathcal{A}^{tc} = \{[0, 0, M-i, i]^{tc} : i = 1 \cdots M\}. \quad (23)$$

The expression of the waiting time distribution when the initial marking is m_0 can be derived as:

$$W^{m_0}(t) = \sum_{a \in \mathcal{A}^{tc}} \pi^{tc}_a(t) \quad (24)$$

where $\pi^{tc}_a(t)$ is the probability that the tagged customer model is in state a at time t and it can be obtained by the model solution.

In order to obtain the system behavior in all the possible conditions, we need to change the initial marking of the tagged customer model. In particular, we have to repeatedly solve the model considering, as initial marking, all the markings m in the set $\mathcal{I}^{tc}$ defined as:

$$\mathcal{I}^{tc} = \{[1, 0, i, M-i]^{tc} : i = 1 \cdots M\} \cup \{[1, j, 0, M]^{tc} : j = 0 \cdots Q-1\}. \quad (25)$$

With respect to the request u, the first subset in eq. (25) represents the condition to be immediately accepted by the system (i.e., no requests waiting in the queue and at least one available resource), while the second subset represents the condition to find the system busy with 0 or $Q-1$ waiting requests. Other conditions are not taken into account because when $P \geq Q$ the request is transferred or dropped and then (assuming the time needed to the VM upload be grater than τ) the corresponding responsiveness is equal to 0.

For each initial marking $m_0 \in \mathcal{I}^{tc}$, the solution of the tagged customer model allows us to obtain a set of CDFs $\{W^{m_0}(t) : m_0 \in \mathcal{I}^{tc}\}$.

In order to obtain the overall waiting time distribution, we need to compute the probability that a particular condition is found when request u arrives. Representing with the 5-vector $[P^{\#}_{mmpp}, P^{\#}_{queue}, P^{\#}_{send}, P^{\#}_{res}, P^{\#}_{run}]^{cp}$ a generic marking of the cloud performance model of Fig. 2, it is possible to univocally associate to each $m_0 \in \mathcal{I}^{tc}$ a set of markings $\mathcal{M}^{cp}(m_0)$ given by:

$$\forall m_0 = [1, \Phi, \Psi, \Omega]^{tc} \in \mathcal{I}^{tc}$$
$$\Rightarrow \mathcal{M}^{cp}(m_0) = \{[i, \Phi, j, \Psi, \Omega]^{cp} : i = 0, 1; j = 0 \cdots D\}. \quad (26)$$

The probability that a particular condition is found when request u arrives can be then obtained by the solution of the cloud performance model, in particular by summing the probabilities that the SRN is in one of the markings in $\mathcal{M}^{cp}(m_0)$ at steady state. The overall waiting time distribution can be then computed as:

$$W(t) = \sum_{m_0 \in \mathcal{I}^{tc}} \left[W^{m_0}(t) \cdot \textstyle\sum_{m' \in \mathcal{M}^{cp}(m_0)} \pi^{cp}_{m'} \right]. \quad (27)$$

Finally, the responsiveness R can be obtained as:

$$R = W(\tau). \quad (28)$$

4 Resiliency Analysis

Through a transient solution of the cloud performance model of Fig. 2, it is possible to investigate the trend over time of some performance metrics. Such an analysis is straightforward to asses the resiliency of the cloud infrastructure, in particular when the load is characterized by bursts. In fact, even if the infrastructure is optimally sized with respect to the expected load, during a load burst users can experience a degradation of the perceived QoS with corresponding violations of SLAs. For this reason, it is needed to predict the effects of a particular load condition in order to study the ability of the system to react to an overload situation.

In order to study the system resiliency we highlight the arrival of a single burst taking into account a Bursty arrival process characterized by the following behavior:

- A regular arrival rate exponentially distributed with rate λ_n
- A single burst with deterministic begin time t_b, deterministic finish time t_f (and then deterministic duration given by $t_f - t_b$), and arrival rate exponentially distributed with rate λ_b ($> \lambda_n$).

The Bursty arrival process is modeled by opportunely changing the exponentially distributed firing time of the transition T_{arr} in the cloud performance model through the adoption of the technique described in [22], [23]. First of all, we can identify three temporal phases:

1) From 0 to t_b: regular load.
2) From t_b to t_f: load burst.
3) From t_f to ∞: regular load.

In each phase the model is solved in transitory by setting the firing rate of T_{arr} with the corresponding mean value: λ_n for the regular load, λ_b for the load burst. Moreover, at the beginning of each phase (i.e., before the change on the firing rate is applied) the initial state probabilities of the model have to be reloaded using the state probabilities obtained at the end of the previous phase. Indicating with R_{arr} the rate of transition T_{arr} and with $\boldsymbol{\pi}^p(t) = [\pi^p_0(t), \pi^p_1(t), \cdots]$ the state probability vector of the model in phase p at time t, we have to perform the following steps:

1) Solve the model in the time interval $[0 : t_b]$ with $R_{arr} = \lambda_n$.
2) Solve the model in the time interval $[0 : t_f - t_b]$ with $R_{arr} = \lambda_b$ and with $\boldsymbol{\pi}^2(0) = \boldsymbol{\pi}^1(t_b)$.
3) Solve the model in the time interval $[0 : \infty)$ with $R_{arr} = \lambda_n$ and with $\boldsymbol{\pi}^3(0) = \boldsymbol{\pi}^2(t_f - t_b)$.

The overall model solution can be then obtained by reconstructing the state probability vector as:

$$\boldsymbol{\pi}(t) = \begin{cases} \boldsymbol{\pi}^1(t) & if \quad t \in [0 : t_b] \\ \boldsymbol{\pi}^2(t - t_b) & if \quad t \in (t_b : t_f] \\ \boldsymbol{\pi}^3(t - t_f) & if \quad t \in (t_f : \infty) \end{cases}. \quad (29)$$

4.1 Performance metrics

Considering that some of the performance indices presented in Section 3 are not applicable in transient analysis

(due for example to the Little's law validity) we propose the following transient metrics that allow us to characterize the system resiliency.

Availability at time t. Denoted by $A(t)$, it can be computed by using the reward functions r^{d1} and r^{d2} defined in Section 3 as:

$$A(t) = 1 - (E[r^{d1}_{X(t)}] + E[r^{d2}_{X(t)}]). \tag{30}$$

Instant service probability at time t. It is the probability $I(t)$ that a request is immediately served and it corresponds to the probability that the system has at least an idle resource at time t. It can be computed through the following reward function

$$r^i = (P^{\#}_{res} > 0) \ ? \ 1 : 0. \tag{31}$$

as:

$$I(t) = E[r^i_{X(t)}]. \tag{32}$$

4.2 Quantitative resiliency metrics

During the resiliency analysis we are interested in the quantitative evaluation of the performance degradation experienced by the system during a load burst. To this end, we propose some temporal indices (see Fig. 4) able to capture the performance degradation trend. Such indices can be applied to both the Availability and Instant service probability metrics.

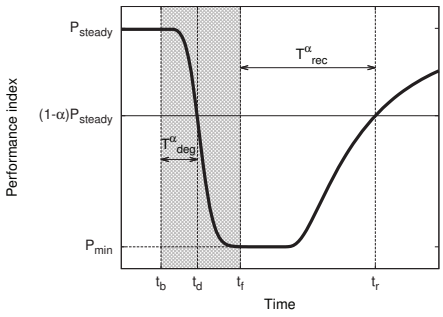


Fig. 4. Graphical representation of the quantitative resiliency metrics: the dashed area corresponds to the burst duration.

α-Degradation time. It is the time T^{α}_{deg} elapsed between the burst begin (t_b) and the instant (t_d) at which the performance index reaches a value lower to the $\alpha \cdot 100\%$ of its steady state value:

$$T^{\alpha}_{deg} = t_d - t_b. \tag{33}$$

It can be considered as an estimation of the time needed to observe the effects of the burst on the selected performance index.

α-Recovery time. It is the time T^{α}_{rec} elapsed between the burst end (t_f) and the instant (t_r) at which the performance indices returns to a value grater than the $\alpha \cdot 100\%$ of its steady state value.

$$T^{\alpha}_{rec} = t_r - t_f. \tag{34}$$

Finally, in order to quantify the overall burst effects we propose the following index.

Maximum performance loss. It is the peak of performance lost during the burst, expressed in percentage form.

$$MPL(\%) = \frac{P_{steady} - P_{min}}{P_{steady}} \cdot 100 \tag{35}$$

where P_{steady} and P_{min} are the steady state value of the considered performance index and its minimum value reached during the load burst.

5 Numerical Results

In this section, we present some numerical results, obtained by solving the proposed analytical model with the SPNP tool [24], illustrating and quantifying the effects of different cloud-based strategies on the performance of the system under different load conditions. Our aim is to provide a performance evaluation approach that can help system managers to opportunely set the system parameters. In order to have a common reference parameter that takes into account the system elasticity, we define the *system capacity* C as the overall number of users that can be concurrently managed by the system. It includes the users being served and those waiting in the queues (both the system and the upload queue). The M logical resources of the system can be divided in physical resources (N) and virtual resources (V) such that $M = N + V$. Then the system capacity can be expressed as:

$$C = N + V + Q + D. \tag{36}$$

We will analyze a cloud system characterized by the following initial parameters: $N = 400$, $V = 0$, $Q = 20$, $D = 0$; i.e., a system with an overall capacity $C = 420$ (composed by 400 physical resources and a system queue with a size set to a value equal to the 5% of the physical resources) and initially without VM multiplexing and federation.

Then we will investigate the system performance taking into account the following strategies able to modify the system capacity:

- `Physical`: modifying the system capacity by adjusting the number of physical resources (i.e., $C = C(N)$).
- `Queueing`: modifying the system capacity by adjusting the system queue size ($C = C(Q)$).
- `Multiplexing`: modifying the system capacity by adjusting the number of virtual resources ($C = C(V)$).
- `Federation`: modifying the system capacity by adjusting the elasticity level and then the size of the upload queue ($C = C(D)$).

We will consider the effects produced on the system by the application of the above strategies in isolation. However, mixed strategies can also be taken into account.

Other system parameters are set as follows. The expected service time is $1/\mu = 10,000sec$. The VM multiplexing degradation factor is $d = 0.2$. The availability of federated clouds is $a_f = 0.98$ and their quality level is $q_f = 0.95$. The expected uploading time is $1/\eta = 1,000sec$ and the number of parallel transfers is $k = 5$. Some of these values

have been taken in accordance with previous works present in the literature (see for example [10]) and they are able to represent a real cloud scenario. However, due to the large variety of cloud systems, other parameter configurations can be easily adopted, without invalidating the model capabilities.

5.1 Constant arrival process

We consider a cloud system characterized by a constant arrival process ($\lambda = 0.04 job/sec$). Let us suppose that a system manager is interested in increasing the system availability. We will show how the proposed model can be used to evaluate the improvements on the system availability obtained by adopting one of the above mentioned strategies, also taking into account the effects on the other performance indices. For each strategy, we increased the system capacity up to a maximum of 10% of the initial capacity.

Fig. 5 shows the obtained results. From Fig. 5(a), it can be observed that the strategy having the greatest impact on the system availability is the `Federation` strategy. `Physical` and `Multiplexing` strategies show similar trend even if `Physical` always outperforms `Multiplexing`, while `Queueing` is the solution that produces less effects. However, in order to effectively choose the right strategy we have to consider other performance metrics. Fig. 5(b) shows the effects on the system utilization. As expected, the `Federation` strategy does not affect the system utilization, because it involves remote resources. On the other hand, the strategies that allow us to further increase the system utilization are `Multiplexing` and `Queueing`. Finally, it is possible to observe that an increasing on physical resource (`Physical` strategy) will reflect on a lower steady state utilization.

From a user perspective, we will investigate the impact of the system capacity on the total service time (i.e., the sum of the waiting time and the service time) and on the system responsiveness (with a value of τ equal to $60 sec$). From the analysis of Figs. 5(c) and 5(d), it is possible to observe that from such a point-of-view the strategy that produces the most tangible effects is `Physical`, allowing us to both reduce the total service time and to increase the responsiveness. With respect to such metrics, `Multiplexing` outperforms `Federation`. In fact, it shows a quasi-constant trend with respect to the total service time thanks to the balancing between the decrease on waiting time (due to the increased number of available resources) and the increase on service time (due to the increase on the multiplexing level). Moreover, the `Multiplexing` strategy allows us to increase the system responsiveness reaching values comparable to those of the `Physical` strategy. Finally, it can be observed that the `Queueing` strategy does not produce any benefit in terms of user satisfaction, resulting in very long waiting times and in a drastic reduction of the system responsiveness.

5.2 Periodic arrival process

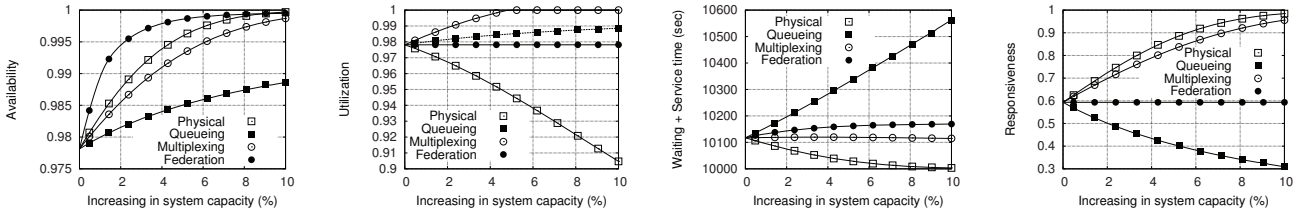
In order to analyze a system characterized by a periodic arrival process we set the MMPP arrival process with the following parameters: $\lambda_l = 0.004 job/sec$, $\lambda_h = 0.04 job/sec$, $1/\lambda_{h2l} = 8 hours$, $1/\lambda_{l2h} = 16 hours$. Such values represent a system with a periodic load characterized by a high load during the working daily period ($8 hours$).

One of the most important challenge in systems characterized by a periodic load is to obtain a high value of utilization without reducing the QoS experienced by users during the high load period. Then, let us suppose that a system manager is interested in increasing the utilization of a cloud system by reducing the number of its physical resources (for example by activating energy saving strategies thus powering off some servers). In order to reduce the effects on the perceived QoS, the system capacity will be kept constant by replacing the powered off physical resources with a corresponding increase of the number of virtual resources (`Multiplexing` strategy), of the system queue size (`Queueing` strategy), or of the elasticity level (`Federation` strategy).

In Fig. 6 we will show the results obtained by applying such three different strategies to a system with a constant capacity $C = 420$ and with a percentage of replaced physical resources that ranges from 0% to 45%. As can be observed in Fig. 6(a), the `Multiplexing` strategy outperforms the others increasing the utilization of the $87,5\%$ when the 45% of physical resources are replaced. By analyzing the negative effects produced on the other performance metrics, we can notice that with respect to the system availability (Fig. 6(b)) we have two different trends: when the percentage of replaced physical resources is low ($< 10\%$) the `Federation` strategy is the one that better limits the performance degradation while for high percentage of replaced physical resources the winning strategy results to be the `Multiplexing`. Also concerning the total service time (Fig. 6(c)) and the responsiveness (Fig. 6(d)), we can quantify the performance degradation related to the increase in the system utilization and we can asses that the `Multiplexing` is the one that allows us to reduce the negative effects.

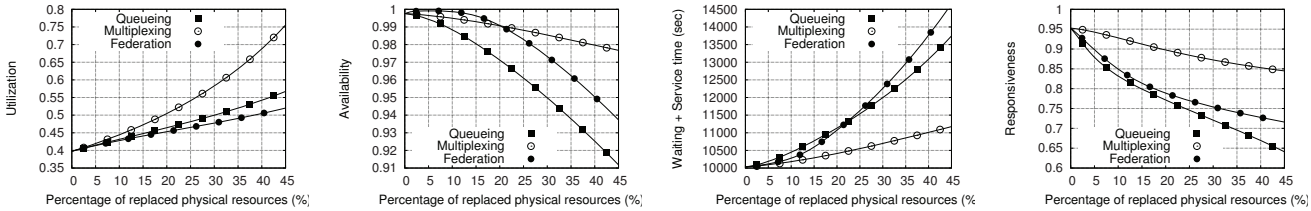
5.3 Bursty arrival process

Finally, in this subsection we carry out a resiliency analysis in order to investigate the effects due to load bursts. Also in this case we will start with an initial configuration (referred to as `Base` configuration) composed of a cloud system with capacity $C = 420$ ($N = 400$, $V = 0$, $Q = 20$, $D = 0$) and we will take into consideration a bursty arrival process characterized by the following parameters: $\lambda_n = 0.035 job/sec$, $\lambda_b = 0.07 job/sec$, $t_b = 5,000 sec$, $t_f = 9,000 sec$. In such conditions the system is highly available during the regular load conditions; in the following we are interested in the evaluation of the system resiliency, in terms of performance degradation, and in the quantification of the benefit reached with a 10% increase of



(a) Availability. (b) Utilization. (c) Waiting + Service time. (d) Responsiveness ($\tau = 60sec$).

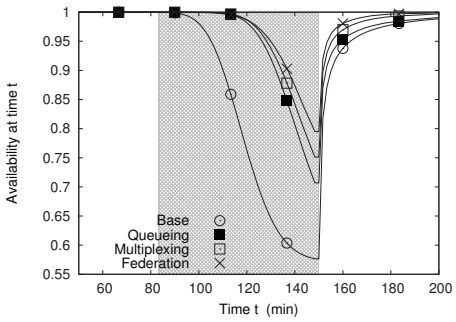
Fig. 5. Steady state measures varying the system capacity under different strategies (Constant arrival process).



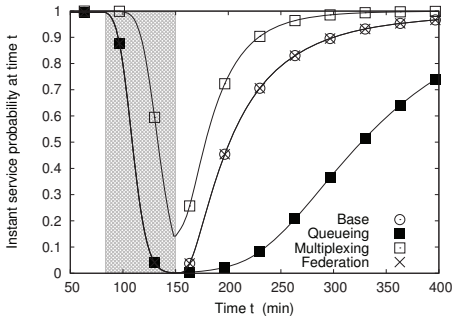
(a) Utilization. (b) Availability. (c) Waiting + Service time. (d) Responsiveness ($\tau = 60sec$).

Fig. 6. Steady state measures replacing a percentage of physical resources under different strategies (Periodic arrival process).

the system capacity obtained by means of the `Queueing`, `Multiplexing`, and `Federation` strategies.



(a) Availability.



(b) Instant service probability.

Fig. 7. Resiliency analysis considering a 10% increase of the system capacity obtained with different strategies: the dashed area corresponds to the burst duration (Bursty arrival process).

In Fig. 7 we show the results obtained using the performance metrics defined in Section 4.1: system availability and instant service probability. With respect to the system availability (Fig. 7(a)), it can be observed that the system performance slightly degrade at the burst begin, reaching a value of 0.57 in the `Base` configuration. Once the burst has finished, the performance rapidly return to the normal values. On the contrary, with respect to the instant service probability (Fig. 7(b)) we have a rapid performance degradation (reaching a value near to 0 in the `Base` configuration) and a consequently slow recovering phase. Then we can asses that the instant service probability is more sensible to the burst load. Moreover, from the analysis of the graphs we can derive a qualitative analysis of the improvements obtained using the cloud strategies. In particular, the use of the `Federation` strategy allows us to sensibly reduce the availability degradation while the `Multiplexing` strategy is the one producing the major effects with respect to the instant service probability. Furthermore, we can observe that the adoption of the `Queueing` strategy gives rise to a strong increase of the time needed to the system to reach the normal values of $I(t)$; this is due to the great number of requests waiting in the queue that generates long waiting times.

A quantitative evaluation of the resiliency properties of the system can be done through the analysis of data in Table 1. Let us start discussing the aspects related to the system availability. As can be observed, the `Federation` strategy increases the 0.05-Degradation time of 25 minutes with respect to the `Base` configuration and of a value ranging from 3 to 5 minutes with respect to the other strategies. Then, using such a strategy the system is able to maintain its performance for a longer time thus reducing the QoS degradation. Similar consideration can be made with respect to the 0.05-Recovery time, in this case the `Federation` strategy produces the shortest time interval.

Such a trend is also confirmed by the value of MPL(%) that, in the case of the `Federation` strategy, is reduced by half with respect to the `Base` configuration.

On the other hand, observing the data referred to the instant service probability, we can quantify the improvements obtained using the `Multiplexing` strategy. In fact, taking as reference the `Base` configuration, we can observe that the 0.05-Degradation time is increased of 20 minutes, while the 0.05-Recovery time is reduced of more than 100 minutes. Moreover, we can argue that the `Federation` strategy does not influence such a system performance index and that the `Queueing` strategy gives rise to the worst results, in particular with respect to the 0.05-Recovery time that reaches a value near to 7 hours (very high if compared with a burst duration of about 1 hour). Finally, the high values of MPL(%) confirm that such a performance metric is strongly influenced by a burst load.

Performance parameter	Strategy	$T_{deg}^{0.05}(min)$	$T_{rec}^{0.05}(min)$	$MPL(\%)$
Availability	`Base`	23.3	13.3	42.3%
	`Queueing`	43.3	10	29.3%
	`Multiplexing`	45	6.7	24.8%
	`Federation`	48.3	5	20.7%
Instant service probability	`Base`	10	208.3	99.8%
	`Queueing`	10	419.7	99.8%
	`Multiplexing`	30	103.3	85.8%
	`Federation`	10	208.3	99.8%

TABLE 1
Resiliency quantitative metrics referred to the results depicted in Fig. 7 ($\alpha = 0.05$).

6 Conclusion and Future Work

In this paper, we have presented a stochastic model to evaluate the performance of an IaaS cloud system. Several performance metrics have been defined, such as availability, utilization, and responsiveness, allowing to investigate the impact of different strategies on both provider and user point-of-views. In a market-oriented area, such as the Cloud Computing, an accurate evaluation of these parameters is required in order to quantify the offered QoS and opportunely manage SLAs. Future works will include the analysis of autonomic techniques able to change on-the-fly the system configuration in order to react to a change on the working conditions. We will also extend the model in order to represent PaaS and SaaS Cloud systems and to integrate the mechanisms needed to capture VM migration and data center consolidation aspects that cover a crucial role in energy saving policies.

References

[1] R. Buyya *et al.*, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comput. Syst.*, vol. 25, pp. 599–616, June 2009.

[2] X. Meng *et al.*, "Efficient resource provisioning in compute clouds via vm multiplexing," in *Proceedings of the 7th international conference on Autonomic computing*, ser. ICAC '10. New York, NY, USA: ACM, 2010, pp. 11–20.

[3] H. Liu *et al.*, "Live virtual machine migration via asynchronous replication and state synchronization," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 12, pp. 1986 –1999, dec. 2011.

[4] B. Rochwerger *et al.*, "Reservoir - when one cloud is not enough," *Computer*, vol. 44, no. 3, pp. 44 –51, march 2011.

[5] R. Buyya, R. Ranjan, and R. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," in *High Performance Computing Simulation, 2009. HPCS '09. International Conference on*, june 2009, pp. 1 –11.

[6] A. Iosup, N. Yigitbasi, and D. Epema, "On the performance variability of production cloud services," in *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, may 2011, pp. 104 –113.

[7] V. Stantchev, "Performance evaluation of cloud computing offerings," in *Advanced Engineering Computing and Applications in Sciences, 2009. ADVCOMP '09. Third International Conference on*, oct. 2009, pp. 187 –192.

[8] S. Ostermann *et al.*, "A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing," in *Cloud Computing*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, 2010, vol. 34, ch. 9, pp. 115–131.

[9] H. Khazaei, J. Misic, and V. Misic, "Performance analysis of cloud computing centers using m/g/m/m+r queuing systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 5, pp. 936 –943, may 2012.

[10] R. Ghosh, K. Trivedi, V. Naik, and D. S. Kim, "End-to-end performability analysis for infrastructure-as-a-service cloud: An interacting stochastic models approach," in *Dependable Computing (PRDC), 2010 IEEE 16th Pacific Rim International Symposium on*, dec. 2010, pp. 125 –132.

[11] G. Ciardo *et al.*, "Automated generation and analysis of Markov reward models using stochastic reward nets." *IMA Volumes in Mathematics and its Applications: Linear Algebra, Markov Chains, and Queueing Models*, vol. 48, pp. 145–191, 1993.

[12] D. Gupta, L. Cherkasova, R. Gardner, and A. Vahdat, "Enforcing performance isolation across virtual machines in xen," in *Proceedings of the ACM/IFIP/USENIX International Conference on Middleware*, New York, NY, USA: Springer-Verlag New York, Inc., 2006, pp. 342–362.

[13] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, pp. 50–58, Apr. 2010.

[14] J. N. Matthews *et al.*, "Quantifying the performance isolation properties of virtualization systems," in *Proceedings of the 2007 workshop on Experimental computer science*, ser. ExpCS '07. New York, NY, USA: ACM, 2007.

[15] M. Mishra and A. Sahoo, "On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, july 2011, pp. 275 –282.

[16] A. V. Do *et al.*, "Profiling applications for virtual machine placement in clouds," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, july 2011, pp. 660 –667.

[17] A. Verma *et al.*, "Server workload analysis for power minimization using consolidation," in *Proceedings of the USENIX Annual technical conference*, Berkeley, CA, USA, 2009, pp. 28–28.

[18] G. Balbo *et al.*, *Modelling with Generalized Stochastic Petri Nets*. John Wiley and Sons, 1995.

[19] R. Sahner, K. S. Trivedi, and A. Puliafito, *Performance and reliability analysis of computer systems: an example based approach using the SHARPE software package*. Kluwer Academic Publishers, 1995.

[20] J. K. Muppala, K. S. Trivedi, and S. P. Woolet, "On modeling performance of real-time systems in the presence of failures." *Readings in Real-Time Systems*, pp. 219–239, 1993.

[21] A. Puliafito, S. Riccobene, and M. Scarpa, "Evaluation of performability parameters in client-server environments." *The Computer Journal*, vol. 39, no. 8, pp. 647–662, 1996.

[22] R. Ghosh, F. Longo, V. Naik, and K. Trivedi, "Quantifying resiliency of iaas cloud," in *Reliable Distributed Systems, 29th IEEE Symposium on*, 2010, pp. 343 –347.

[23] "Spnp manual," www.ee.duke.edu/~chirel/MANUAL/SPNPv6-manual.pdf.

[24] G. Ciardo, J. Muppala, and K. S. Trivedi, "SPNP: Stochastic Petri Net Package," in *3rd International Workshop on Petri-nets and Performance Models* Los Alamitos, California, 1989, pp. 142–151.

Dario Bruneo received the Ph.D. in Advanced Technologies for Information Engineering at the University of Messina (Italy) in 2005. He is an Assistant Professor at the Engineering Faculty of the University of Messina. His scientific activity has been focused on studying distributed systems, particularly with regard to management techniques. His main research interests include Grid and Cloud computing, sensor networks, QoS management, and performance evaluation.

www.redpel.com
+917620593389