RC25037 (W1008-053) August 17, 2010
Computer Science

IBM Research Report

Quantifying Resiliency of IaaS Cloud

Rahul Ghosh[1], Francesco Longo[2], Vijay K. Naik[3], Kishor S. Trivedi[1]

[1]Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708 USA

[2]Università degli Studi di Messina
Dipartimento di Matematica
Contrada di Dio, S. Agata
98166 Messina, Italy

[3]IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598 USA

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Quantifying Resiliency of IaaS Cloud

Rahul Ghosh*, Francesco Longo†, Vijay K. Naik‡, and Kishor S. Trivedi*
*Dept. of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA
†Università degli Studi di Messina, Dipartimento di Matematica, Contrada di Dio, S.Agata, 98166 Messina, Italia
‡IBM T. J. Watson Research Center, Hawthorne, NY 10532, USA
Email: {rg51, kst}@ee.duke.edu, flongo@unime.it, vkn@us.ibm.com

***Abstract*—Cloud based services may experience changes – internal, external, large, small – at any time. Predicting and quantifying the effects on the quality-of-service during and after a change are important in the resiliency assessment of a cloud based service. In this paper, we quantify the resiliency of infrastructure-as-a-service (IaaS) cloud when subject to changes in demand and available capacity. Using a stochastic reward net based model for provisioning and servicing requests in a IaaS cloud, we quantify the resiliency of IaaS cloud w.r.t. two key performance measures – job rejection rate and provisioning response delay.**

I. Introduction

An Infrastructure-as-a-Service (IaaS) cloud, e.g., Amazon EC2 [1] and IBM Smart Business Development and Test Cloud [2], delivers on-demand virtual machine instances deployed in the cloud provider's data center. This paper presents resiliency analysis for IaaS cloud. A key problem in performing such analysis is the lack of consensus on the definition and a systematic approach for quantifying the resiliency measures. Traditionally, resiliency has been interpreted as the fault-tolerant capability of the system. Sterbenz *et. al* [3] defined it as the combination of trustworthiness (dependability, security, and performability) and tolerance (survivability, disruption tolerance, and traffic tolerance). Laprie [4] and Simoncini [5] defined resiliency as the persistence of service delivery that can justifiably be trusted, when facing *changes*.

Our definition of resiliency includes this notion of *change*. For IaaS cloud, *changes* occur because of increased workload, faultload, system capacity or from security attacks and accidents (e.g., disasters). We consider two types of changes: change in client demand (job arrival rate) and change in system capacity (number of available physical machines). We quantify cloud resiliency in terms of effect of changes on two performance based quality-of-service (QoS) measures: job rejection rate and provisioning response delay as measured by the mean of jobs waiting to be serviced. Our contributions are: (1) A stochastic reward net based analysis: for performance analysis of IaaS cloud using a stochastic Petri net and determining performance measures using reward functions; (2) Resiliency metrics for IaaS Cloud for quantifying the effects of change; (3) Resiliency analysis approach for IaaS cloud when subject to changes in service demand and service capacity and quantify these effects using performance analysis models.

Rest of the paper is organized as follows. Section II presents IaaS cloud system model and assumptions. In Section III, we describe the stochastic reward net approach. Resiliency quantification of performance measures and numerical results are presented in Section IV. Finally, concluding remarks and future research are in Section V.

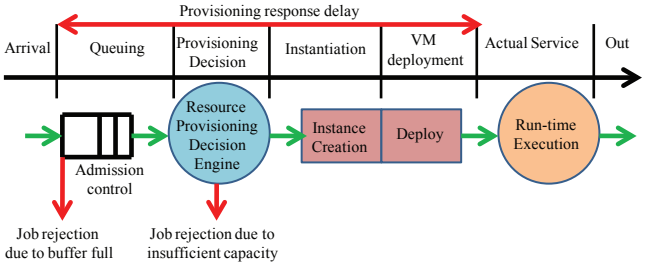

Fig. 1: Request provisioning and servicing steps in IaaS cloud.

II. System Model

Shown in Figure 1 is the life-cycle of a job as it moves through an IaaS cloud. When a request is processed, a pre-built image is used to create one or more Virtual Machine (VM) instances. When the VM instances are deployed, they are provisioned with request specific CPU, RAM, and disk capacity. VMs are deployed on Physical Machines (PMs) each of which may be shared by multiple VMs. To reduce overall VM provisioning delays and operational costs, we assume that PMs are grouped into three servers pools: hot (i.e., running), warm (turned on, but not ready) and cold (turned off). A pre-instantiated VM can be provisioned and brought to ready state on hot PMs with minimum provisioning delay. Instantiating a VM from an image and provisioning it on a warm PM needs additional provisioning time. PMs in the cold pool need additional startup time to be turned on before a VM deployment.

In the subsequent discussions we use the term job to mean a user request for provisioning a VM and making it available for use by a cloud user. We assume that all requests are homogeneous and each request is for one VM with fixed size CPU cores and RAM. User requests (i.e., jobs) are submitted to a global resource provisioning decision engine (RPDE) that processes requests on a first-come, first-served (FCFS) basis as follows. The request at the head of the queue is provisioned on a hot server if there is capacity to run a VM on one of the hot servers. If no hot PM is available, a PM from warm pool is used for provisioning the requested VM. If all warm machines are busy, a PM from cold pool is used. If none of these servers are available, the request is rejected. When a

running job exits, the capacity used by that VM is released and becomes available for provisioning the next job.

For the above described scenario, we quantify the resiliency of IaaS w.r.t. two pure performance measures - (i) net job rejection rate (ρ_{reject}) and (ii) mean number of jobs in RPDE (E_{RPDE}), due to changes in job arrival rate and system capacity. We assume that all the inter-event times are exponentially distributed and hence the system model is a homogeneous continuous time Markov chain (CTMC). Since the CTMC has a large state space, we use a variant of stochastic Petri net for concise specification and automated solution of the model.

III. Proposed Stochastic Reward Net Approach

Stochastic reward nets (SRNs) [6] are extensions of generalized stochastic Petri nets (GSPNs) [7]. In SRNs, every tangible marking of the net can be associated with a reward rate thus facilitating the computation of a variety of performance measures. Key features of SRNs are: (1) each transition may have an enabling function (also called a guard) so that a transition is enabled only if its marking-dependent enabling function is true; (2) marking dependent arc multiplicities are allowed; (3) marking dependent firing rates are allowed; (4) transitions can be assigned different priorities; (5) besides traditional output measures obtained from a GSPN, such as throughput of a transition and mean number of tokens in a place, more complex measures can be computed by using reward functions.

SRN model for IaaS cloud. An SRN model for request provisioning in an IaaS cloud is shown in Figure 2. Transition T_{arr} with rate λ models the job arrivals while tokens in place P_{d_h} represent job requests waiting to be served by the RPDE. Transition T_{d_h} models the searching delay to find a hot PM that can be used for resource provisioning. A token in place P_{sel_h} represents a request that is trying to get provisioned on a hot PM. We assume there are n_h, n_w, and n_c PMs in hot, warm, and cold pool, respectively and maximum number of VMs on each PM is m. Tokens in places $P_{vm_{hi}}$ (with $1 \leq i \leq n_h$) represent the number of jobs that the i^{th} PM in the hot pool can still accept. When i^{th} PM is not running any job, m tokens will be present in $P_{vm_{hi}}$ place. A job accepted in the hot pool is provisioned on a PM, randomly selected from the set of hot PMs which are not full. This is modeled by the conflict among transitions $t_{y_{hi}}$ (with $1 \leq i \leq n_h$) each of which takes a token from the corresponding place $P_{vm_{hi}}$ and inserts a token in the corresponding place $P_{inst_{hi}}$. VM provisioning and deployment delay on a hot PM is modeled by transitions $T_{inst_{hi}}$ (with rate β_h). Places $P_{serv_{hi}}$ and transitions $T_{serv_{hi}}$ represent servicing of jobs. In particular, the rates of transitions $T_{serv_{hi}}$ depend on the number of tokens in places $P_{serv_{hi}}$ to model the concurrent job executions within the same PM. If no tokens are present in places $P_{vm_{hi}}$ (all the PMs in the hot pool are occupied), transition t_{n_h} is enabled modeling the necessity for the RPDE to start looking for a free PM in the warm pool by moving a token from place P_{sel_h} to place P_{d_w}. Transition T_{d_w} models the searching delay for the warm pool. We assume that the

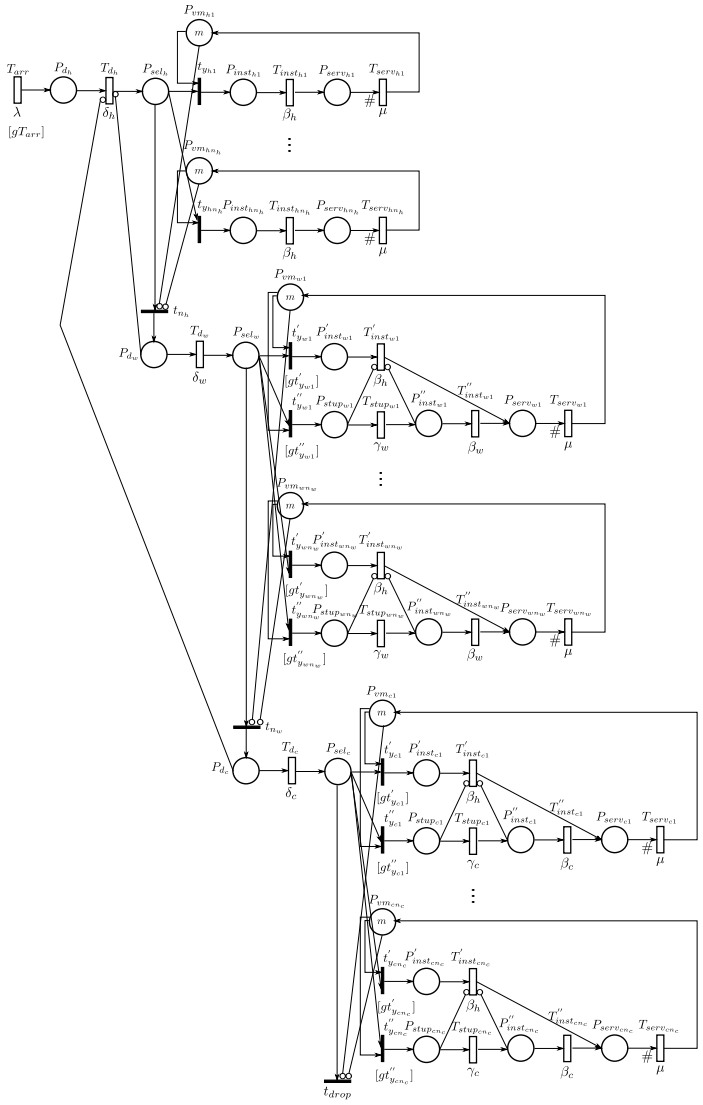

Fig. 2: Stochastic reward net model for IaaS cloud.

RPDE can process only one request at a time and this is represented by the inhibitor arc from place P_{d_w} to transition T_{d_h}. The modeling of the n_w PMs in the warm pool is similar to that of PMs in the hot pool. However, when there is no job being executed or provisioned on a warm PM, the first request coming to the warm PM requires an additional startup delay. For this reason, two immediate transitions are present for each PM ($t^{'}_{y_{wi}}$ and $t^{''}_{y_{wi}}$) with proper guards. Transitions $t^{''}_{y_{wi}}$ will be enabled only if the number of tokens in the corresponding places $P_{vm_{wi}}$ is equal to m, i.e., only for the first request to the PM ($[gt^{''}_{y_{wi}}] = 1 \; if \; \#P_{vm_{wi}} = m$). In such a way, we model the additional delay by means of places $P_{stup_{wi}}$ and transitions $T_{stup_{wi}}$ (with rate γ_w). On the other hand, Transitions $t^{'}_{y_{wi}}$ will be enabled only if the number of tokens in places $P_{vm_{wi}}$ is strictly less than m, i.e., for every subsequent requests after the first request ($[gt^{'}_{y_{wi}}] = 1 \; if \; \#P_{vm_{wi}} < m$).

When PMs in the warm pool are all busy (no tokens in places $P_{vm_{wi}}$), transition t_{n_w} is enabled modeling the RPDE trying to provision the request in the cold pool. Cold pool

is modeled similarly as the warm pool. Main difference with warm pool is that when all PMs are busy in the cold pool, transition t_{drop} is enabled and will fire. Firing of t_{drop} models the rejection of a request due to insufficient machine capacity. In our model N is the maximum number of requests that can be present in RPDE. By associating a guard function ($[gT_{arr}] = 1 \; if \; \#P_{d_h} + \#P_{d_w} + \#P_{d_c} < N$) to transition T_{arr} we model the finite length FCFS queue of the RPDE.

Model outputs. Outputs of the model are obtained using the Markov reward approach by assigning an appropriate reward rate to each marking of the SRN and then computing the expected reward rate both in transient and steady state as the desired measures [8]. Let r_i be the reward rate assigned to marking i of the SRN in Figure 2. If $\pi_i(t)$ denotes the probability for the SRN to be in marking i at time t then the expected reward rate at time t is given by $\sum_i \pi_i(t) r_i$. The expected steady state reward rate can be computed by taking into consideration the steady state probabilities π_i of the SRN as $\sum_i \pi_i r_i$. Our measures of interest are following. **(i) Net job rejection rate** (ρ_{reject}). There are two component of ρ_{reject}. The first component results from admission control and is denoted as ρ_{block}, which results from rejection of jobs when RPDE buffer is full. Even after job is admitted in the RPDE queue, if during its provisioning decision step all (hot, warm and cold) machines are fully occupied then we assume that the job is dropped. This second component of the rejection rate is denoted as ρ_{drop}. Reward assignments to compute ρ_{block} and ρ_{drop} are shown in Table I. Net rejection rate ρ_{reject} is the sum of ρ_{block} and ρ_{drop}. **(ii) Mean number of jobs in the RPDE** ($E[N_{_{RPDE}}]$). It is given by the sum of the number of jobs that are waiting in the RPDE queue and the job that is currently undergoing provisioning decision. Provisioning response delay is a function of this measure [9]. Reward assignment for this measure is shown in Table I.

TABLE I: Reward rates to compute different output measures

Measures	**Reward rates**
Job rejection rate due to buffer full (ρ_{block})	λ if $[gT_{arr}] = 0$; 0 o/w
Job rejection rate due to insufficient capacity (ρ_{drop})	δ_c if t_{drop} is enabled; 0 o/w
Mean number of jobs in RPDE ($E[N_{_{RPDE}}]$)	$\#P_{d_h} + \#P_{d_w} + \#P_{d_c}$

IV. Resiliency Quantification

We use Stochastic Petri Net Package (SPNP) [10] to solve the SRN model and quantify the IaaS cloud resiliency w.r.t. the above performance measures. General steps of resiliency quantification are:

(1) Determine the stead state behavior of the system to compute steady state values of the performance measures of interest.

(2) Apply change(s) in the system model by changing the value(s) of input parameter(s).

(3) Analyze the transient behavior of the system model to compute the transient performance measures using input parameters after the change. Initial probabilities for this transient analysis are obtained from the steady state probabilities as obtained in step (1). The SPNP package allows to easily map the state space of the step (1) model into the state space of the step (3) model and to compute the initial probability to be used for step (3) analysis in a straightforward manner.

Resiliency Metrics. Let $M_{ss}^{(bc)}$ and $M_{ss}^{(ac)}$ denote the steady state values of the measures (for which resiliency is computed) before and after the application of change respectively. We define following metrics [11] to quantify resiliency. **(1) Settling Time** (t_{set})**.** This is defined as the elapsed time from the time when the change is applied to the system until the measure of interest reaches and stays within $\pm\delta\%$ of $|M_{ss}^{(bc)} - M_{ss}^{(ac)}|$. **(2) Peak Overshoot or Undershoot** (PO)**.** Let $M_{peak}^{(ac)}$ be the maximum deviation of the measure from its steady state value after application of change. Peak overshoot/undershoot (in percentage) is then defined as:

$$PO(\%) = \frac{|M_{peak}^{(ac)} - M_{ss}^{(ac)}|}{M_{ss}^{(ac)}} \times 100 \qquad (1)$$

(3) Peak Time (t_{peak})**.** Time taken to reach maximum deviation $M_{peak}^{(ac)}$ of the measure, from its steady state value ($M_{ss}^{(ac)}$) attained after the change is applied, is defined as peak time. **(4) γ-Percentile Time** ($t_{percentile}$)**.** This is useful when the measure of interest never goes above or below the steady state value ($M_{ss}^{(ac)}$), after the change is applied. It is defined as the elapsed time from the time when the change is applied to the system until the output measure reaches $(100-\gamma)\%$ of $|M_{ss}^{(bc)} - M_{ss}^{(ac)}|$. $(100-\gamma)$ is always greater than δ as used in settling time definition.

In this paper, we consider two types of changes: change in job-arrival rate (λ) and change in the number of PMs. For the former case, we consider two different mean provisioning delays on hot PM (1 minute and 5 minutes). Mean provisioning delay on each warm PM was assumed to be twice the mean provisioning delay on a hot PM. Mean provisioning delay on each cold PM was assumed to be four times the mean provisioning delay on a hot PM. In Table II, we define ten cases that we have analyzed and we numerically compare the corresponding resiliency metrics (with $\gamma = 90$ and $\delta = 0.1$). In general, we observe that, settling time is longer for net rejection rate compared to mean number of jobs in RPDE. We also observe that in some cases, peak overshoot/undershoot is not observed and hence percentile time gives better insight.

Effect of changing job-arrival rate (λ)**.** Figure 3(a) shows the transient effect of changing the job arrival rate (λ) on net job rejection rate. At time instance ($t = 0$, value of λ is changed from 10 to 20 jobs/hr. In case-I and case-II, settling times are $t_{set}^{(I)}$ and $t_{set}^{(II)}$ respectively. When the change is removed (at $t = 1$ hr, job arrival rate is reduced to initial value, i.e., 10 jobs/hr), net rejection rate reduces and finally reaches a steady state value. Settling times after removal of the change are $t_{set}^{(III)}$ and $t_{set}^{(IV)}$ respectively. Observe, if we increase the mean provisioning delay, net rejection rate becomes higher. This is because jobs take longer duration to be provisioned, hence, queue in front of the RPDE builds

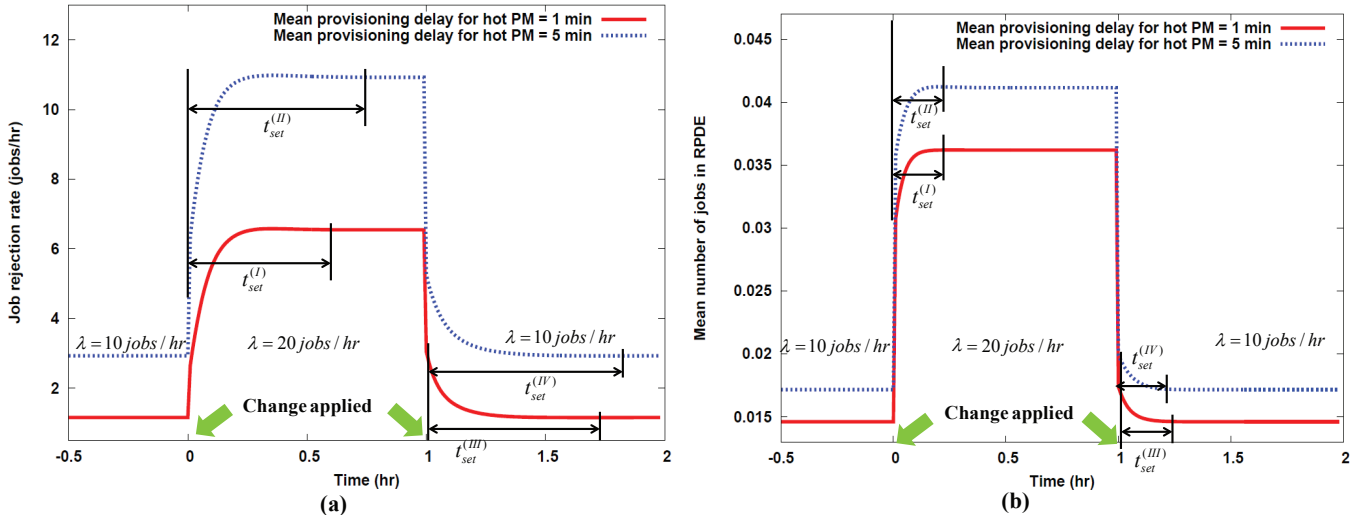

Fig. 3: Effect of changing job arrival rate on (a) net rejection rate and (b) mean number of jobs in the RPDE.

TABLE II: Comparison of resiliency metrics for different changes.

Changes	**Resiliency measures for rejection rate**				**Resiliency measures for mean # jobs in RPDE**			
	t_{set}	$PO(\%)$	t_{peak}	$t_{percentile}$	t_{set}	$PO(\%)$	t_{peak}	$t_{percentile}$
Increase in λ, mean provisioning delay in hot PM is 1 minute (case-I)	0.64	0.485	0.35	0.07	0.12	0.008	0.24	0.01
Increase in λ, mean provisioning delay in hot PM is 5 minutes (case-II)	0.82	0.519	0.36	0.03	0.12	0.168	0.02	0.01
Decrease in λ, mean provisioning delay in hot PM is 1 minute (case-III)	0.81	0.045	1.68	0.39	0.10	–	–	0.02
Decrease in λ, mean provisioning delay in hot PM is 5 minutes (case-IV)	0.95	0.520	0.77	0.10	0.08	0.024	0.38	0.01
Removal of a hot PM (case-V)	1.94	–	–	0.16	0.70	–	–	0.34
Removal of a warm PM (case-VI)	0.98	–	–	0.55	0.52	–	–	0.49
Removal of a cold PM (case-VII)	0.85	–	–	0.46	0.08	–	–	0.04
Addition of a hot PM (case-VIII)	0.92	34.6	0.04	0.03	0.22	21.4	0.04	0.03
Addition of a warm PM (case-IX)	0.97	49.0	0.04	0.03	0.55	8.94	0.04	0.03
Addition of a cold PM (case-X)	0.99	60.0	0.04	0.03	0.58	1.02	0.04	0.03

up quickly and more jobs are rejected. After increasing the arrival rate, difference between the net rejection rates (for two different mean provisioning delays) increases by more than factor of two. Moreover, settling time also increases if the mean provisioning delay is longer. This shows that IaaS cloud with faster provisioning process is more *resilient* to changing workload. Similar effects are shown in Figure 3(b), where the resiliency is computed for the mean number of jobs in RPDE (a measure of congestion in the cloud). **Effect of changing system capacity.** Next, we study the resiliency of IaaS cloud due to addition or removal of PMs. Figure 4(a) shows the effect of removing one PM on the net job rejection rate. We consider three cases: removal of a hot PM (case-V), removal of a warm PM (case-VI) and removal of a cold PM (case-VII). In all three cases, the new PM is removed from a pool (hot/warm/cold) at the time $t = 0$. After the removal of the PM, rejection rate increases and finally reaches a steady state. Removal of a cold PM (case-VII) has the lowest settling time and hence, change in net rejection rate stabilizes quickly. Clearly, cloud resiliency is maximum in this case. In contrast, resiliency is minimum in case-V, when a hot PM is removed. Similar effects on the mean number of jobs in RPDE are in Figure 4(b). Figure 5(a) shows the transient effect of adding a new PM on net job rejection rate. We consider three cases: addition of a hot PM (case-VIII), addition of a warm PM (case-IX) and addition of a cold PM (case-X). In all three cases, the new PM is added to a pool (hot/warm/cold) at the time instance ($t = 0$). After the addition of a new PM, rejection rate momentarily reduces to a low value and then starts increasing and finally reaches a steady state. This sharp "notch" in the rejection rate value can be explained in following way. When the new PM is added to a pool, initially it is empty for a very small duration of time. After that, once the PM is full, rejection rate increases and finally reaches a steady state. Addition of a hot PM provides maximum benefit as the rejection rate is minimum in this case. Similar effects on the mean number of jobs in RPDE are shown in Figure 5(a). Addition of cold PM has almost negligible effect on the mean number of jobs. Settling time and net rejection rate are the smallest for adding a hot PM.

V. Conclusions

In this paper, we quantify the resiliency of IaaS cloud w.r.t. two QoS measures when subjected to changes in job arrival rate and system capacity. To the best of our knowledge,

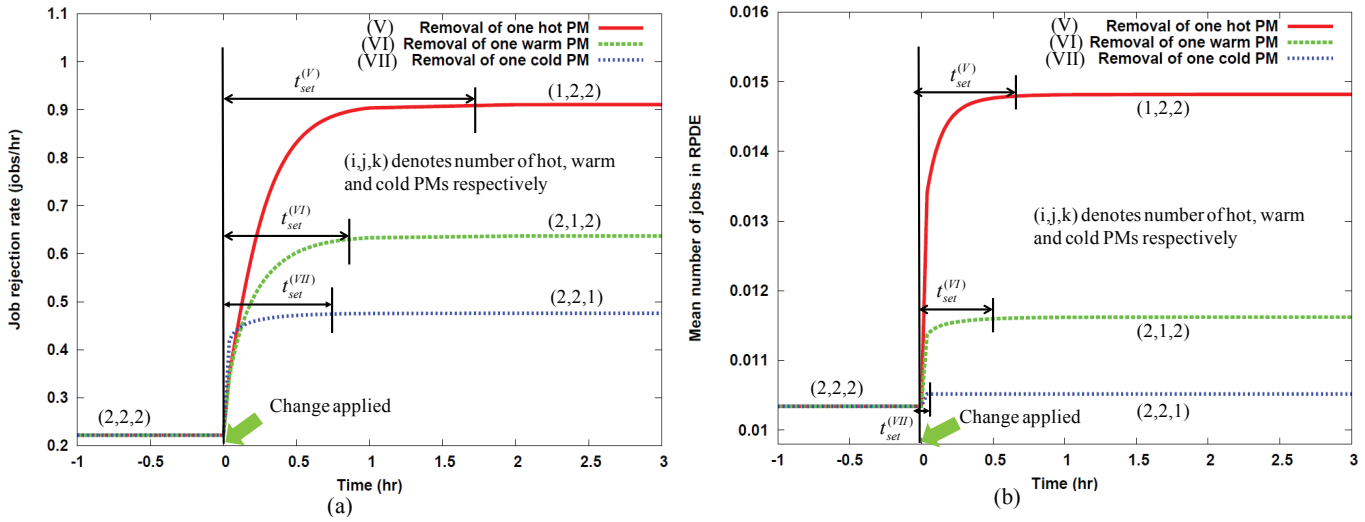

Fig. 4: Effect of removing a PM on (a) net rejection rate and (b) mean number of jobs in the RPDE.

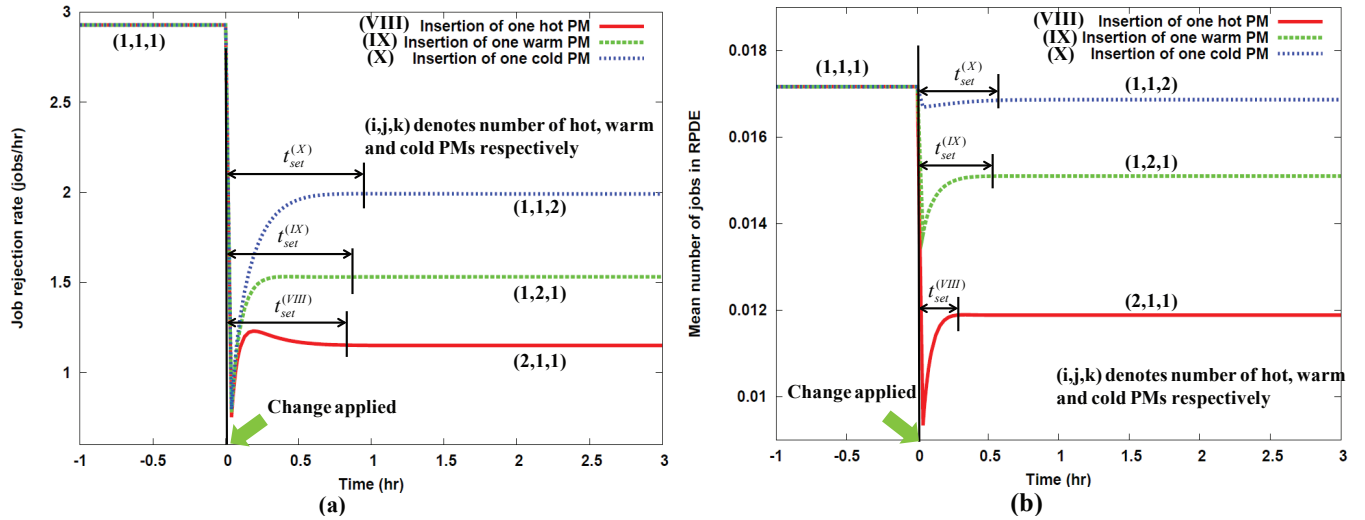

Fig. 5: Effect of adding a PM on (a) net rejection rate and (b) mean number of jobs in the RPDE.

this is perhaps the first work that quantifies resiliency of IaaS cloud. In our future work, we plan on investigating the effects of different types of failure (such as failures of PMs and VMs) and compute on the resiliency of availability and performability [9] of IaaS cloud.

Acknowledgments: Ghosh and Trivedi were supported in part under a 2009 IBM Faculty Award and by NSF under Grant NSF-CNS-08-31325. The authors thank Dan Dias and Murthy Devarakonda from IBM Research for their support and DongSeong Kim for his insightful comments and providing background materials for this work. This work was completed during Rahul's internship at IBM T. J. Watson Research Center.

REFERENCES

[1] "Amazon EC2," http://aws.amazon.com/ec2.
[2] "IBM SBDTC," http://www-180.ibm.com/cloud/enterprise/beta/dashboard.
[3] J. P. Sterbenz *et al.*, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Elsevier Computer Networks*, June 2010.
[4] J. C. Laprie, "From dependability to resilience," in *DSN*, 2008.
[5] L. Simoncini, "Resilient computing: An engineering discipline," in *IPDPS*, 2009.
[6] G. Ciardo *et al.*, "Automated generation and analysis of Markov reward models using stochastic reward nets," in *Linear Algebra, Markov Chains and Queuing Models*. Springer, 1993.
[7] M. A. Marsan, G. Balbo, and G. Conte, "A class of generalized stochastic petri nets for the performance evaluation of the multiprocessor systems," *ACM Transactions on Computer Systems*, 1984.
[8] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. Wiley, 2001.
[9] 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 *PRDC*, 2010.
[10] C. Hirel, B. Tuffin, and K. S. Trivedi, "SPNP: stochastic petri nets. version 6," in *Lecture Notes in Computer Science*, 2000.
[11] T1A1.2 Working Group on Network Survivability Performance, "Enhanced Network Survivability Performance," Technical Report T1.TR.68-2001, February 2001.