2013 International Conference on Parallel and Distributed Systems

Availability Modeling and Evaluation of Cloud Virtual Data Centers

Mohammad Roohitavaf, Reza Entezari-Maleki, and Ali Movaghar
Performance and Dependability Laboratory, Department of Computer Engineering,
Sharif University of Technology, Tehran, Iran
Email: {roohitavaf, entezari}@ce.sharif.edu, movaghar@sharif.edu

***Abstract*—Availability of the service delivered by cloud providers is one of the most important QoS factors of the service level agreements between providers and customers. Since current Infrastructure-as-a-Service providers use virtualization technology to manage data centers, virtual data centers (VDCs) have become a popular infrastructure for cloud computing. In order to study the service availability, a stochastic activity network (SAN) model is presented in this paper. The proposed SAN model can be appropriately used to investigate the impact of different characteristics and policies on service availability of VDCs.**

***Index Terms*—Availability; virtual data center; cloud computing; stochastic activity network**

I. Introduction

Cloud computing is a promising paradigm delivering IT services as computing utilities [1]. Customers of cloud computing use the infrastructure of a data center provided by another corporation named cloud provider. In the lowest level of abstraction, the cloud provider delivers machines to the customers, and customers manage the software stack of the machines by themselves. This service model of cloud computing is called Infrastructure-as-a-Service (IaaS). Current IaaS providers usually use virtualization technology to manage their own data centers. Virtualization using a software layer called Virtual Machine Monitor (VMM) enables the execution of multiple Virtual Machines (VMs) on a single Physical Machine (PM). Data centers which are built using virtualization technology with VMs as the basic processing elements are called Virtual Data Centers (VDCs) [2], [3].

Large cloud service providers provide customers with service level agreements (SLAs) specifying the availability of the cloud service [4], [5]. To guarantee the service availability given in SLA, an IaaS provider needs to investigate the impact of different characteristics, configurations, and resource management policies of the VDC on the service availability. However, due to the complexity of the architecture and virtualization mechanisms, availability evaluation of VDCs is a challenging issue.

Virtualization technology enables server consolidation which makes it possible to reduce the number of required hardware resources by executing multiple VMs on a single PM [6]. The server consolidation can be considered as a power saving policy by the IaaS provider. In order to reduce the number of required PMs by server consolidation, VMs should be frequently migrated between PMs which considerably reduces the service availability of the VDC in some cases. Therefore, the impact of server consolidation should be taken into account whenever the service availability is evaluated.

In this paper, a comprehensive model using Stochastic Activity Networks (SANs) [7], [8] is proposed to evaluate the measures related to the service availability and power consumption of VDCs. Using the proposed model, it is possible to study the effect of variations in workload, different characteristics, and different resource management policies on the service availability.

The rest of this paper is organized as follows. Section II introduces some related work done in this research area. Section IV describes VDC system and some assumptions considered in this paper. The proposed model and measures of interest are described in Section V. Numerical results obtained by solving the proposed model are reported in Section VI. Finally, Section VII concludes the paper and presents future work.

II. Related Work

There are several studies focusing on dependability evaluation of a virtualized server. Zhang et al. [9] have presented a Markov chain model to analyze the dependability aspects of a virtual cluster node. Kim et al. [10] have presented a model for a system consisting of two virtualized physical servers using two-level hierarchical approach in which fault trees and Markov chain sub-models are used in the upper and lower levels, respectively. Wei et al. [2] have studied reliability and availability of a VDC using hybrid models combining Reliability Block Diagrams (RBDs) and Generalized Stochastic Petri Nets (GSPNs). Ghosh et al. [11] have proposed a model-based approach for performability analysis of a cloud service. Longo et al. [12] have studied the availability of IaaS cloud using Stochastic Reward Nets (SRNs) which is limited to PMs.

Tein et al. [13] have proposed a method to improve the software rejuvenation mechanism. The proposed method in [13] uses Markov chains to model and investigate the effect of virtualization on the system availability. Rezaei et al. [14] have modeled a virtualized server using SRNs, and then analyzed the efficiency of the proposed method for improving software rejuvenation. Machida et al. [15] have studied how different methods of VMM rejuvenation can affect the availability of a system consisting of two physical servers. More details of the system including different methods for migrating VMs were considered by the same authors in [16].

Some other related papers can be also found in this research area. Most of the papers only consider one or two physical machines, paying no attention to the systems with more machines. Therefore, the proposed approaches cannot be used to analyze a system with various numbers of PMs. Compared to the existing models, the proposed model in this paper can be used for analyzing a VDC with different number of PMs. Furthermore, to the best of our knowledge, none of the previously done studies in this research area has investigated details of VM placement and migration. Compared to these studies, the model presented in this paper considers details of resource management policies to investigate their effects on service availability of VDCs.

III. Overview of SAN

Stochastic Activity Networks (SANs) [7], [17], [8] are the stochastic generalization of Petri Nets (PNs) generally defined for modeling and analysis of distributed real time systems. SANs are more flexible than other stochastic extensions of PNs such as Stochastic Petri Nets (SPNs) and Generalized Stochastic Petri Nets (GSPNs) [17]. Informally, SANs can be described with the following primitives [18]:

Place: Places are similar to the places in PNs and graphically are represented by circles.

Timed activity: Timed activities are for modeling actions of the system whose duration affects performance of the modeled system noticeably. Graphically, timed activities are represented by thick vertical bars or boxes. Any timed activity can have several inputs and outputs. An input of a timed activity can be a place or an input gate, and similarly an output can be a place or an output gate. An activity distribution function, an enabling rate function, and a computable predicate called the reactivation predicate are associated to each timed activity.

Instantaneous activity: Instantaneous activities are for modeling actions of the system which are done in a negligible amount of time compared to the other actions which can be modeled using timed activities. Graphically, instantaneous activities are represented by thin vertical bars. An instantaneous activity can have several inputs and outputs.

Input gate: Gates provide higher flexibility in defining enabling and completion rules. An input gate has a finite set of inputs and one output. A computable predicate called enabling predicate and a computable function called input function are associated to each input gate.

Output gate: An output gate has a finite set of outputs and one input. A computable function called the output function is associated to each output gate.

Formal definition and some useful properties of SANs have been discussed in [7] and [17]. For the sake of brevity, the formal definition and other preliminaries of SANs are not presented here. For more information about SANs, please see the references mentioned above.

IV. System Description

In the system considered in this paper, there are M PMs in the VDC that each of them can host finite number of VMs. Requests for VMs are submitted to the system by customers of the IaaS cloud. For each VM request, IaaS provider initiates a VM on a PM. The PM allocated to the requested VM is selected by the VM placement policy defined for the VDC.

PMs in the VDC can be assumed to be in active or sleep mode. PMs in sleep mode consume less power compared to the PMs in active mode. However, a PM should be in active mode to be able to host a VM. It takes a while for a PM in sleep mode to wake up and switch to active mode. A PM involving in wake up process is considered to be unavailable to host VMs. Although keeping all PMs in the active mode provides higher availability, it can considerably increase power consumption of the VDC.

Once a VM finishes its own task, the utilization of the hosting PM decreases. If the utilization of the hosting PM reaches a specified threshold called $low\ threshold$, the PM is tagged as a low utilized PM. If there are other PMs with sufficient capacity, the VMs of the low utilized PM are migrated to them. The low utilized PM after migrating its VMs is switched to sleep mode to save power. VMs during migration are considered to be unavailable. As discussed in [16], the unavailability interval varies for different methods of VM migration. Using live migration method, this interval can be reduced to only few seconds.

Once a PM fails, all of its hosted VMs become unavailable and should be restarted on other available PMs. If there is no sufficient amount of computational capacity available on other PMs, VMs of the failed PM remains unavailable. PMs are repaired after they failed. Repaired PMs become available and they are considered to be in active mode after repair. However, if there is no request in the system they are switched to sleep mode.

V. The Proposed Model

A. SAN Model

Sub-models of the proposed SAN model for the VDC are shown in Fig. 1. The overall SAN model is obtained by connecting the sub-models. The $arrival\ and\ placement$ sub-model shown in Fig. 1(a) models the arrival of requests for VMs and their placement on PMs. Tokens inside place P_{vtp} represent VMs waiting to be placed on PMs. Those VMs are either new accepted VM requests or VMs of the failed PMs. Timed activity TA_{arr} represents the arrival of a new VM request. The activity distribution function associated with this timed activity is an exponential function. The parameter λ_{arr}, shown in the Fig. 1(a), is the rate of the exponential function.

A new VM request is accepted if there is enough free available capacity in the data center. The possibility of accepting a new VM request is checked by the enabling predicate of input gate IG_{arr} which is shown in Table I. Once timed activity TA_{arr} fires, a token is deposited into place P_{vtp} by the output function of the output gate OG_{arr}.

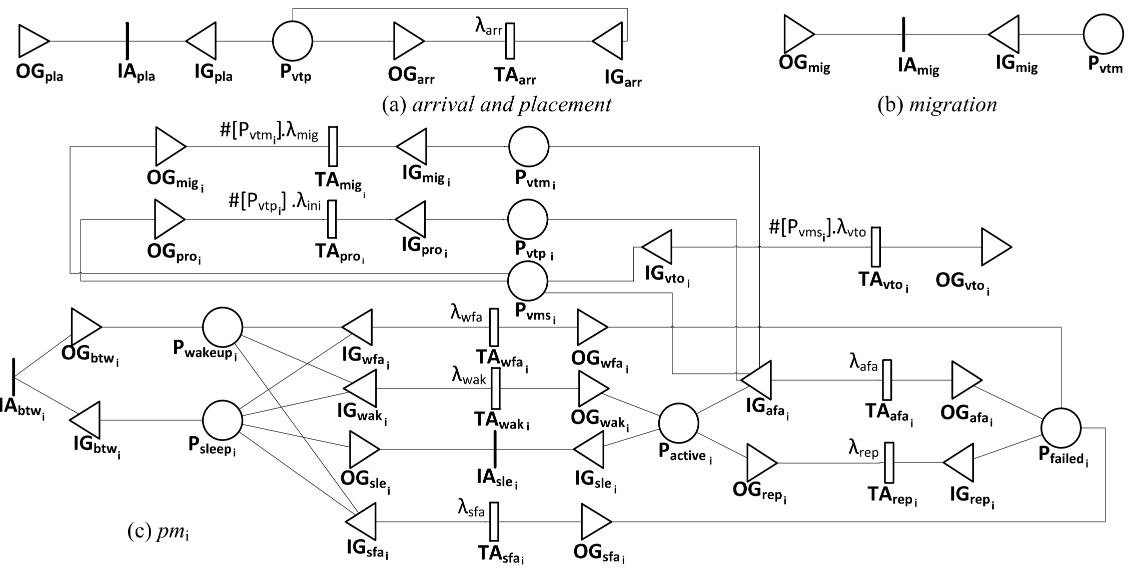

Fig. 1. The proposed SAN model

Instantaneous activity IA_{pla} models the placement of a VM on a PM. The enabling predicate of input gate IG_{pla} checks the possibility of placing a VM on a PM. This predicate returns true if there is at least one VM waiting for placement and there exists at least one active PM with enough free capacity to host a new VM. Upon firing instantaneous activity IA_{pla}, a token is removed from place P_{vtp} and a token is deposited into place P_{vtp_i} of pm_i sub-model which is selected for hosting the VM. The PM for hosting a VM is selected by the placement algorithm implemented in the output function of output gate OG_{pla}. The algorithm used in this output function places the VM on the most utilized PM which has enough capacity to host the VM. Different algorithms for placing VMs can be applied in the output function of the output gate OG_{pla}.

The $migration$ sub-model shown in Fig. 1(b) models the migration of VMs from their original hosting PMs to the target PMs. Tokens inside place P_{vtm} represent VMs which should be migrated to other PMs. The target PM is selected by the output function of the output gate OG_{mig} which is the same as the corresponding function of the gate OG_{pla}. Input gate IG_{mig} is a simple input gate which checks the existence of at least one token in place P_{vtm}. Upon firing instantaneous activity IA_{mig}, a token is removed from place P_{vtm} and a token is deposited into place P_{vtm_i} of the target pm_i sub-model.

The pm_i sub-model shown in Fig. 1(c) models a PM. A token inside place P_{active_i} (P_{sleep_i}) shows that PM i is in active (sleep) mode. The requested capacity and the capacity available on both PMs in active mode and PMs involved in the wake up process are checked by the enabling predicate of input gate IG_{btw_i} which is show in Table I. This predicate returns true if PM i is in sleep mode and the requested capacity is more than the capacity available on both PMs in active mode and PMs involved in the wake up process.

Once instantaneous activity IA_{btw_i} fires, a token is deposited into place P_{wakeup_i}. Timed activity TA_{wak_i} represents the wake up process of a PM. The enabling predicate of input gate IG_{wak_i} returns true if there is at least one token in place P_{wakeup_i}, and one token in P_{sleep_i}. Upon firing timed activity TA_{wak_i}, a token from place P_{wakeup_i} together with another token from P_{sleep_i} is removed and one token is deposited into place P_{active_i} which shows PM i is in active mode and ready to host VMs.

Timed activity TA_{sle_i} represents the event of putting a PM into sleep mode.The enabling predicate of input gate IG_{sle_i}, shown in Table I returns true whenever the capacity available on active PMs other than pm_i is enough to host the requested VMs, and pm_i is active, but it is not hosting any VM.

Timed activity TA_{afa_i} represents the failure event of a PM in active mode. The enabling predicate of input gate IG_{afa_i} checks the existence of at least one token in place P_{active_i}. The output function of gate OG_{afa_i} is shown in Table II. Timed activity TA_{sfa_i} represents the failure event of a PM in sleep mode. The enabling predicate of input gate IG_{sfa_i} returns true if there is at least one token in place P_{sleep_i} and there is no token in place P_{wakeup_i}. Upon firing timed activity TA_{sfa_i}, a token is removed from place P_{sleep_i} and one token is deposited into place P_{failed_i}. Moreover, a PM may fail during wake up process. This event is modeled by timed activity TA_{wfa_i}. The input predicate of input gate IG_{wfa_i} checks the existence of at least one token in place P_{wakeup_i}. Upon firing timed activity TA_{wfa_i} a token from place P_{wakeup_i} and another token from place P_{sleep_i} are removed and one token is deposited into place P_{failed_i}. Timed activity TA_{rep_i} represents the repair process of a failed PM. The input predicate of input gate IG_{rep_i} checks the existence of at least one token in place P_{failed_i}. Upon firing timed activity TA_{rep_i}, one token is removed from place P_{failed_i} and one token is deposited into place P_{active_i}.

Tokens inside place P_{vms_i} represent VMs hosted by PM i. Timed activity TA_{vto_i} represents the turn off event of a VM. The firing rate of this activity is marking dependent, so its actual firing rate is computed as $\#[P_{vms_i}] \cdot \lambda_{vto}$, where $\#[P_{vms_i}]$ is the number of tokens inside place P_{vms_i}. The enabling predicate of input gate IG_{vto_i} checks the existence of at least one token in place P_{vms_i}. Upon firing timed activity TA_{vto_i}, one token is removed from place P_{vms_i}, and simultaneously current number of VMs hosted by PM i of PM i is checked. If it reaches the *low threshold*, and the total available capacity on other active PMs is enough to host VMs of PM i, those VMs should be migrated to other hosts. In this case, all tokens existing in places P_{vms_i}, P_{vtm_i}, and P_{vtp} are removed by firing timed activity TA_{vto_i}. The tokens removed from place P_{vtp_i} are deposited into place P_{vtp}, and the tokens removed from places P_{vtm_i} and P_{vms_i} are deposited into place P_{vtm}.

Tokens inside place P_{vtm_i} model the VMs migrating to PM i. Firing timed activity TA_{mig_i}, migration of a VM to PM i is completed. The enabling predicate of input gate IG_{mig_i} checks the existence of at least one token in place P_{vtm_i}. Upon firing timed activity TA_{mig_i}, one token is removed from place P_{vtm_i} and deposited into place P_{vms_i}.

Tokens inside place P_{vtp_i} represent VMs which would be initiated on PM i. Timed activity TA_{pro_i} represents completion of initiating a VM on PM i. The enabling predicate of input gate IG_{pro_i} checks the existence of at least one token in place P_{vtp_i}. Upon firing timed activity TA_{pro_i}, one token is removed from place P_{vtp_i}, and one token is deposited into place P_{vms_i}.

B. Measures of Interest

In order to calculate different measures of interest, appropriate reward rates are assigned to states of the underlying Markov chain resulting from the proposed SAN model. Corresponding to each feasible marking of the SAN, there is a state in its underlying Markov chain. Let r_m denote the reward rate associated to state m, and Ω represent the state space of the Markov chain. Therefore, interesting measures can be obtained by computing the expected reward rate in the steady state as:

$$\sum_{m \in \Omega} r_m \cdot \pi_m, \tag{1}$$

where π_m is the steady state probability of the Markov chain to be in state m. The output measures of the proposed SAN model are as follows.

Ratio of available VMs to accepted VM requests (R_{ava}). This measure is defined as the mean ratio of the number of available VMs to the total number of accepted VM requests in steady state. This measure reflects the availability of the service considering all the customers of the VDC, and can be used to optimized the VDC. The reward assigned to markings of the proposed SAN to compute R_{ava} is as:

$$r_m = \Big(\sum_{i=1}^{M} \#P^m_{vms_i}\Big) \Big/ \Big(\#P^m_{vtp} + \#P^m_{vtm} + \sum_{i=1}^{M} \big(\#P^m_{vms_i} + \#P^m_{vtp_i} + \#P^m_{vtm_i}\big)\Big), \tag{2}$$

where $\#P^m$ denotes the number of tokens inside place P in marking m. In the states in which the denominator of the fraction is zero, the reward rate is 1, as there is no request in the system and the system is considered as fully operational.

Mean number of PMs in active mode (N_{PM}). The greater value for the mean number of PMs in active mode results in more power consumption of the VDC. Actually, this measure is useful for a cloud provider, because a cloud provider wishes to minimize the power consumption of the VDC. The reward assigned to compute N_{PM} is as:

$$r_m = \sum_{i=1}^{M} \#P^m_{active_i} \tag{3}$$

VI. Numerical Results

We use Möbius tool [19] to solve the proposed SAN model. To analyze the impact of different system configurations on output measures, results obtained from applying the proposed SAN to the sample VDC is reported in this section, and sensitivity analysis is done on the results. The default rate values used for the sample VDC are shown in Table III. We set all rates according to the values used in previous studies [15], [16] and typical values of occurrence frequency. However, in a practical VDC, the values for the parameters can be obtained using measurement and estimation. We use the model to study a VDC with four PMs that each of them can host three VMs. However, a VDC with different number of PMs can be analyzed using the proposed model. We also set *low threshold* to one, which means, in power saving mode, a PM is marked as low utilized PM when it hosts only one VM.

The effect of increasing mean time to migration on R_{ava} is shown in Fig. 2. The mean time to migration reflects the expected time to migrate a VM from its hosting PM to another one. As can be seen in Fig. 2, the value of R_{ava} decreases by applying power saving policy in the VDC. Moreover, the value of R_{ava} decreases with increasing mean time to migration in power saving mode. However, there is no change in R_{ava} if power saving policy is not applied to the VDC. Therefore, to keep the service availability in an acceptable level, using effective migration methods is important to have a short mean time to migration.

The effect of increasing mean time to waking up a PM on R_{ava} is shown in Fig. 3. Increasing the mean time to waking up a PM decreases the value of R_{ava}. The decrement rate of R_{ava} when the power saving policy is applied is more than the situation in which it is not applied. The reason behind this

TABLE I
ENABLING PREDICATES

Gate	Code
IG_{arr}	$AvailCapacity = (M - \sum_{i=1}^{M} \#[P_{failed_i}]) \cdot C;$ $systemLoad = \#[P_{vtm}] + \#[P_{vtp}] + \sum_{i=1}^{M} \#[P_{vtm_i}] + \#[P_{vtp_i}] + \#[P_{vms_i}];$ $(AvailCapacity > systemLoad)\ ?\ return 1 : 0;$
IG_{pla}	$\left(\sum_{i=1}^{M} \#[P_{active_i}] \cdot \left(C - PMLoad(i)\right) \geq 1\right)$ **and** $(\#[P_{vtp}] > 0)\ ?\ return 1 : 0;$
IG_{btw_i}	$AWCapacity = \sum_{i=1}^{M} \left(\#[P_{active_i}] + \#[P_{wakeup_i}]\right) \cdot \left(C - PMLoad(i)\right);$ $((\#[P_{vtm}] + \#[P_{vtp}] > AWCapacity)$ **and** $(\#[P_{sleep_i} > 0)\ ?\ return 1 : 0;$
IG_{sle_i}	$\left(\sum_{j=1, j \neq i}^{M} \#[P_{active_j}] \cdot \left(C - PMLoad(j)\right) \geq \#[P_{vtm}] + \#[P_{vtp}] + \#[P_{vtm_i}] + \#[P_{vtp_i}]\right)$ **and** $\left(\#[P_{active_i}] > 0\right.$ **and** $\left.\left(\#[P_{vms_i}] = 0\right)\right)\ ?\ return 1 : return 0;$

TABLE II
OUTPUT FUNCTIONS

Gate	Code
OG_{sle_i}	$\#[P_{active_i}] - -; \quad \#[P_{sleep_i}] + +;$ $\#[P_{vtm}] + = \#[P_{vtm_i}]; \quad \#[P_{vtm_i}] = 0;$ $\#[P_{vtp}] + = \#[P_{vtp_i}]; \quad \#[P_{vtp_i}] = 0;$
OG_{afa_i}	$\#[P_{active_i}] - -; \quad \#[P_{failed_i}] + +;$ $\#[P_{vtm}] + = \#[P_{vtm_i}]; \quad \#[P_{vtm_i}] = 0;$ $\#[P_{vtp}] + = \#[P_{vtp_i}] + \#[P_{vms_i}]; \quad \#[P_{vtp_i}] = 0; \quad \#[P_{vms_i}] = 0;$

TABLE III
DEFAULT VALUES FOR RATES

Rate	**Value**
λ_{arr}	4 h^{-1}
λ_{vto}	1 h^{-1}
λ_{mig}	1200 h^{-1}
λ_{ini}	120 h^{-1}
λ_{afa}	0.00139 h^{-1}
λ_{sfa}	0.000139 h^{-1}
λ_{wfa}	0.00139 h^{-1}
λ_{rep}	2 h^{-1}
λ_{wak}	1440 h^{-1}

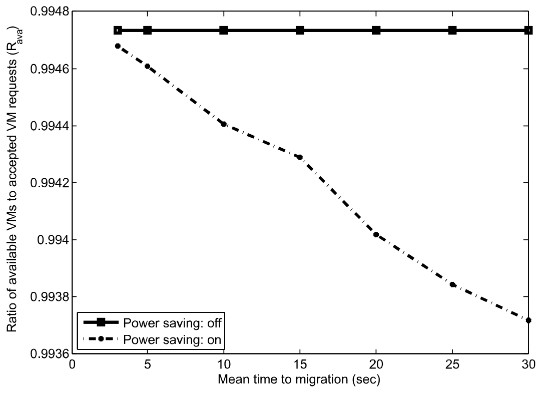

Fig. 2. R_{ava} vs. mean time to migration

phenomenon is that when the power saving policy is applied, PMs switch to sleep mode more frequently.

Fig. 4 shows that, with a fixed request arrival rate, increasing mean service time of VM increases N_{PM} of the VDC. Using power saving policy in the VDC results in fewer number of PMs in active mode which leads to less power consumption.

VII. CONCLUSION AND FUTURE WORK

In this paper, an availability model for a VDC using SANs is proposed which can be used to analyze a VDC with various number of PMs. In addition to configuration and characteristics of VDCs, we have incorporated the details of virtualization mechanism including VM placement and VM migration into the model. We have studied the effect of two different placement policies on service availability. However, the proposed model can be used to study other placement algorithms. As a part of future work we plan to investigate the effects of different possible placement policies on the service availability. The proposed model can also be appropriately used for analyzing a VDC with different capacities. Therefore, as a part of future work, it is possible to carry out cost analysis and capacity planning using the proposed model.

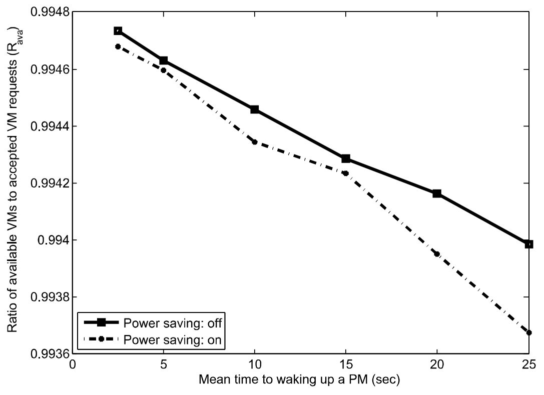

Fig. 3. R_{ava} vs. mean time to waking up a PM

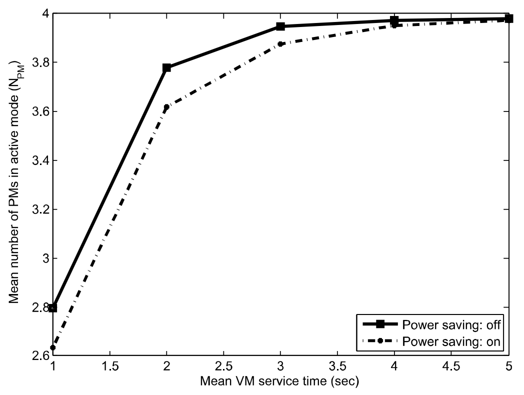

Fig. 4. N_{PM} vs. mean service time of a VM

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] B. Wei, C. Lin, and X. Kong, "Dependability modeling and analysis for the virtual data center of cloud computing," in *Proceedings of the 13th IEEE International Conference on High Performance Computing and Communications*, Banff, AB, Canada, September 2-4 2011, pp. 784–789.

[3] S. Graupner, V. Kotov, and H. Trinks, "Resource-sharing and service deployment in virtual data centers," in *Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops*, Vienna, Austria, July 2-5 2002, pp. 666–671.

[4] "Amazon elastic compute cloud (amazon ec2) [online]," *http://aws.amazon.com/ec2/.*

[5] "Ibm smartcloud enterprise [online]," *http://www.ibm.com/cloud-computing/us/en/.*

[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164–177, 2003.

[7] A. Movaghar and J. F. Meyer, "Performability modeling with stochastic activity networks," in *Proceedings of the Real-Time Systems Symposium*, Austin, TX, USA, December 7-9 1984, pp. 215–224.

[8] W. H. Sanders and J. F. Meyer, "Stochastic activity networks: Formal definitions and concepts," in *Lectures on formal methods and performance analysis*, ser. Lecture Notes in Computer Science, E. Brinksma, H. Hermanns, and J.-P. Katoen, Eds., vol. 2090. The Netherlands: Springer, July 3-7 2001, pp. 315–343.

[9] X. Zhang, C. Lin, and X. Kong, "Model-driven dependability analysis of virtualization systems," in *Proceedings of the 8th IEEE/ACIS International Conference on Computer and Information Science*, Shanghai, China, June 1-3 2009, pp. 199–204.

[10] D. S. Kim, F. Machida, and K. S. Trivedi, "Availability modeling and analysis of a virtualized system," in *Proceedings of the 15th IEEE Pacific Rim International Symposium on Dependable Computing*, Shanghai, China, November 16-18 2009, pp. 365–371.

[11] 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 16th IEEE Pacific Rim International Symposium on Dependable Computing*, Tokyo, Japan, December 13-15 2010, pp. 125–132.

[12] F. Longo, R. Ghosh, V. K. Niak, and K. S. Trivedi, "A scalable availability model for infrastructure-as-a-service cloud," in *Proceedings of the 41st IEEE/IFIP International Conference on Dependable Systems and Networks*, Hong Kong, June 27-30 2011, pp. 335–346.

[13] T. Thein, S.-D. Chi, and J. S. Park, "Improving fault tolerance by virtualization and software rejuvenation," in *Proceedings of the 2nd Asia International Conference on Modeling and Simulation*, Kuala Lumpur, Malaysia, May 13-15 2008, pp. 855–860.

[14] A. Rezaei and M. Sharifi, "Rejuvenating high available virtualized systems," in *Proceedings of the International Conference on Availability, Reliability and Security*, Krakow, Poland, February 15-18 2010, pp. 289–294.

[15] F. Machida, D. S. Kim, and K. S. Trivedi, "Modeling and analysis of software rejuvenation in a server virtualized system," in *Proceedings of the 2nd IEEE International Workshop on Software Againg and Rejuvenation*, San Jose, CA, USA, November 2-2 2010, pp. 1–6.

[16] ——, "Modeling and analysis of software rejuvenation in a server virtualized system with live vm migration," *Performance Evaluation*, vol. 70, no. 3, pp. 212–230, 2012.

[17] A. Movaghar, "Stochastic activity networks: A new definition and some properties," *Scientia Iranica*, vol. 8, no. 4, pp. 303–311, 2001.

[18] M. A. Azgomi and A. Movaghar, "A modeling tool for a new definition of stochastic activity networks," *Iranian Journal of Science and Technology*, vol. 29, no. B1, pp. 79–92, 2005.

[19] D. Daly, D. Deavours, P. W. J. Doyle, , and W. H. Sanders, "Möbius: An extensible tool for performance and dependability modeling," in *Proceedings of the Computer Performance Evaluation. Modelling Techniques and Tools*, Schaunmburg, IL, USA, March 25-31 2000, pp. 332–336.