Performability Models for Designing Disaster Tolerant Infrastructure-as-a-Service Cloud Computing Systems

Bruno Silva, Paulo Maciel
Center for Informatics (CIn)
Federal University of Pernambuco (UFPE)
Recife, Pernambuco
{bs,prmm}@cin.ufpe.br

Armin Zimmermann
System & Software Engineering
Ilmenau University of Technology
Ilmenau, Germany
Armin.Zimmermann@tu-ilmenau.de

***Abstract*—Providing high availability in a diverse client demand scenario is a key challenge of cloud computing providers. A possible approach to protect cloud systems from natural disasters corresponds to the utilization of redundant data centers located far enough apart. The time to migrate a virtual machine (VM) from a data center to another increases due to distance. As this time impacts the overall system behavior, performability evaluation considering VM migration time and different load balances is of utmost importance when considering the analysis of distributed cloud systems. This work presents performability models for evaluating distributed cloud computing systems deployed into multiple data centers considering disaster occurrence. Additionally, we present a case study which evaluates a set of scenarios with different client loads and distances between data centers.**

***Index Terms*—performability evaluation, IaaS systems, stochastic petri nets**

I. Introduction

Cloud computing has driven the new wave of Internet-based applications by providing computing as a service [1]. Nowadays, usual business applications (e.g., spreadsheets, text editors) are provided as cloud computing services, in the sense that they are often accessed using a web browser, and, their respective software/data reside on remote servers. This approach has affected all fields of the computational system, from users to hardware manufacturers [2].

Such paradigm is attractive for a number of reasons: (i) it frees users from installing, configuring and updating the software applications; (ii) it offers advantages in terms of mobility as well as collaboration; and (iii) updates and bug fixes can be deployed in minutes, simultaneously affecting all users around the globe [3]. An important type of cloud service is the Infrastructure-as-a-Service (IaaS), such as Amazon EC2 [4] and IBM Smart Business Cloud [5]. IaaS delivers, on-demand, computing resources in the form of virtual machines (VMs) deployed into the cloud provider's data center, satisfying user needs. User requests are provisioned depending on the data center capacity in terms of physical machines.

Considering the user point of view, performability measures are prominent indicators to assess provider's quality-of-service (QoS). These metrics take into account the effects of queuing and failure/recovery behavior of data center subsystems. For prominent IaaS providers, the quality level is regulated by adopting a Service Level Agreement (SLA), which specifies, for instance, the maximum downtime per year. Penalties may be applied if the defined quality level is not satisfied. Thus, to meet SLA requirements, IaaS providers need to evaluate the performability level of its environment, considering, also, the possibility of disasters.

A disaster recovery plan requires the utilization of different data centers located far enough apart to mitigate the effects of unforeseen disasters (e.g., earthquakes) [6]. If multiple data centers are located in different geographical locations (considering disaster independent places), the availability level of whole system will improve. On the other hand, VM migration time increases due to distance between data centers. Additionally, failures and overloads can lead to system downtime considering cloud computing systems. Consequently, performability evaluation considering VM migration time and different user loads levels is of utmost importance when considering the analysis of distributed cloud systems.

This work presents an approach to evaluate performability metrics in IaaS systems deployed into geographically distributed data centers as well as taking into account disaster occurrence. The proposed approach contemplates state-based models (SPN - Stochastic Petri Nets) to allow performability evaluation. Using the proposed approach, IaaS providers can evaluate the system distributed in different data centers and the impact of VM migration on performability metrics. In this paper, we extend our previous work from [7] and provide new models for performability evaluation, as well as present user and provider oriented metrics.

The paper is organized as follows. Section II highlights the related works. Section III describes the cloud computing system considered. Then, the performability models are introduced in Section IV and V, and Section VI presents a case study. Finally, Section VII concludes this paper and introduces future works.

II. Related Work

Over the last years, some authors have been devoting efforts to study dependability issues on cloud computing systems. Longo et al. [8] proposed an approach for availability analysis of cloud computing systems based on Petri nets and Markov chains. The authors also developed closed-form equations and demonstrated that their approach can scale for large systems.

In [9], a performability analysis for cloud systems is presented. The authors quantify the effects of variations in workload, failure rate and system capacity on service quality. In [10], the authors investigate the aging effects on the Eucalyptus framework [11], and they also propose a strategy to mitigate such issues during system execution.

[12] describes a system design approach for supporting transparent migration of virtual machines that adopt local storage for their persistent state. The approach is transparent to the migrated VM, and it does not interrupt open network connections during VM migration. In [13], the authors present a case study that quantifies the effect of VM live migrations in the performance of an Internet application. Such study helps data center designers to plan environments in which metrics, such as service availability and responsiveness, are driven by Service Level Agreements.

Dantas et al. [14] presents a study of warm-standby mechanisms in Eucalyptus framework. Their results demonstrate that replacing machines by more reliable counterparts would not produce improvements in system availability, whereas some techniques of fault-tolerance can indeed increase dependability levels.

Unlike previous works, this paper proposes performability models for evaluating cloud computing systems deployed into geographically distributed data centers, considering VM migration, disasters occurrence and different user loads. Moreover, performability metrics taking into account user and provider perspectives are adopted by this work.

III. System Architecture of Reliable Distributed Data Centers

This section presents an overview of the cloud computing system considered in this work, which contemplates a set of components, distributed over distinct data centers (Figure 1). The system is composed of d data centers, each with two set of machines, namely, hot and warm pools. The hot pool is composed of n physical machines (PM), which are active and running virtual machines (VM). The warm pool consists of m PMs that are active, but without running VMs. Thus, the number of PMs in a data center is $t = m + n$.

Depending on the capacity of each PM, it is possible to run multiple VMs in the same host. In this study, we assume all physical machines are identical, in the sense that they adopt the same services and hardware/software components. PMs may share a common network attached storage (NAS) or a storage area network (SAN) to provide distributed storage and to allow the migration of a virtual machine from one server to another in the same data center [15]. In case of failure, a VM must be instantiated in another physical machine. If there is no available PM, the VM image is migrated to another data center.

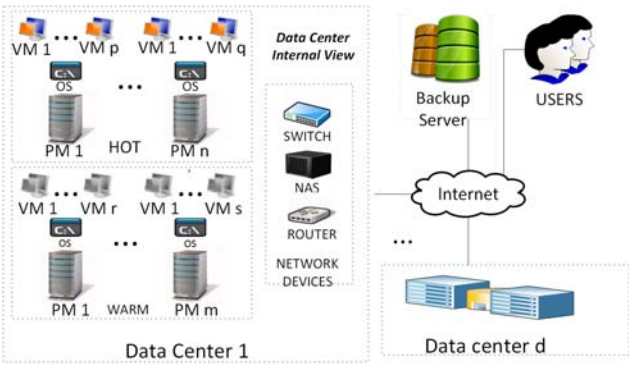

Fig. 1. Distributed Cloud System Example

Furthermore, a Backup Server (BS) is assumed to provide backup of VM data. This component receives a copy of each VM image during data center operation. Hence, whenever a disaster makes one data center unavailable, BS sends VM copies to an operational data center. In this work, the number of running VMs (w) is compared with a threshold (k) to evaluate the availability of cloud computing system. Hence, if $w \geq k$ the system is assumed operational.

IV. Scalable Cloud System Models

This section presents the adopted hierarchical modeling to evaluate the system. The adopted modeling process considers first the evaluation of lower-level submodels, then the respective results are applied to higher-level models. Dependability results estimated from the RBD models are associated to SPN models. After that, the basic SPN models are configured and combined to create the SPN final model. As a future work we will create a software tool to perform this process automatically.

For further details about the hierarchical modelling approach the reader is referred to [7]. The reader should refer to [16] and [17] for details about performability concepts. Henceforth, the following operators are adopted for assessing dependability metrics: $P\{exp\}$ estimates the probability of the inner expression (exp); $\#p$ denotes the number of tokens in place p and $E\{\#p\}$ estimates its expected number.

A. SPN block: generic component

The generic component (Figure 2) is adopted to represent components that have no redundancy and might be in two states, either functioning or failed. In order to compute its availability, mean time to failure (*MTTF*) and mean time to repair (*MTTR*) are the only parameters needed for computing its availability. The respective SPN model of this model is shown in Figure 2. Both transitions are exponentially distributed (exp) and have single server (ss) semantic [18].

Places *X_ON* and *X_OFF* are the model component's activity and inactivity states, respectively. Label "X" is instantiated according to the component name, for instance, *DC_UP1* and

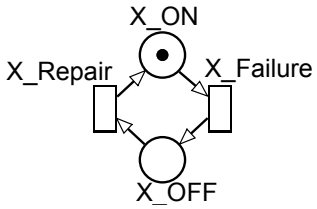

Fig. 2. Generic component SPN model

DC_DOWN1 (Figure 5). A component is operational only if the number of tokens in place *X_ON* is greater than 0.

B. SPN block: VM performability component

VM performability component represents VM requests on a single physical machine considering failures and repairs on the underlying infrastructure. Whenever a user request is performed, a new VM is started (considering that the infrastructure is operational) and becomes available for some time. If the external infrastructure or the VM fail during the virtual machine execution, the VM should be migrated to another physical machine. If there is no available physical machine in the system, the task is rejected and a new request must be performed.

This component interacts with three generic components: (i) one representing the occurrence of disasters (*DC*); (ii) the network infrastructure (*NAS_NET*); and (iii) the physical machine (*OSPM*). Figure 3 presents the VM performability model, which is composed of three main parts: (i) *VM_PART* represents the VM behavior running on a single machine; (ii) *DC_PART* which express the incoming requests to data center; and (iii) *CLOUD_PART* that models the requests generation.

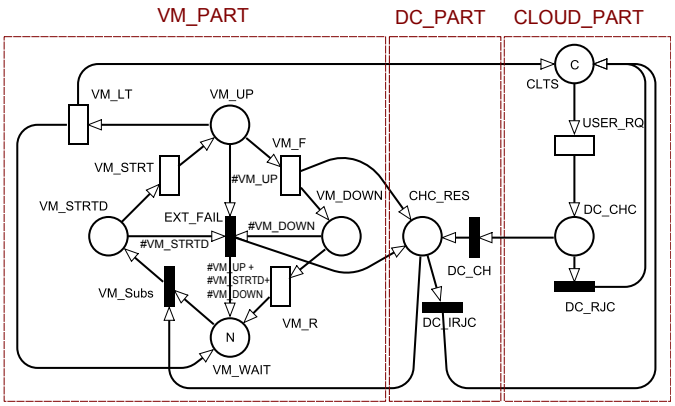

Fig. 3. VM performability model

Considering the *VM_PART*, places *VM_UP*, *VM_DOWN*, *VM_STRTD* and *VM_WAIT* denote, respectively, the amount of VMs in states operational, failed, starting, and waiting request. The exponential transition *VM_LT* represents the period in which the VM is operational. Transitions *VM_F*, *VM_R* and *VM_STRT* represent the failure, repair and starting activities related to the virtual machines. The before mentioned exponential transition have infinite server (is) semantics.

The association with the underlying infrastructure is carried out by immediate transitions *EXT_FAIL*, *VM_Subs* and the respective guard conditions (see Table I). *EXT_FAIL* transition verifies disaster occurrence as well as the failures on the physical machine and network devices. A VM fails whenever the respective infrastructure is not capable to provide the service. Transition *VM_Subs* denotes the opposite idea, in the sense that virtual machines start only if the required infrastructure is operational.

EXT_FAIL presents input arcs from *VM_UP*, *VM_DOWN* and *VM_STRTD*. Therefore, if there were not arc multiplicities, the transition would fire if the respective guard expression was evaluated to true and all input places had tokens. However, the multiplicity of input and output arcs associated with *EXT_FAIL* allows a different behavior. In this case, the transition fires if at least one input place has tokens and the transition guard is evaluated as true. Whenever a external failure occur, the tokens of input places are instantaneously eliminated and the sum of removed tokens is inserted in *VM_WAIT*.

DC_PART models the incoming requests to data center and the failed VMs that must be migrated to another machine. The *DC_PART* of different VM performability components are merged if the respective physical machines are located in the same data center. In other words, just one *DC_PART* is represented for each data center (see Figure 5).

Place *CHC_RES* represent the incoming requests to the current data center. In case of no available machine in the system, the request is cancelled. The task cancellation is represented by the transition *DC_IRJC*. Furthermore, *DC_CH* symbolizes the input of new requests to data center. The guard expressions related to these transitions are represented in Table I.

TABLE I
GUARD EXPRESSIONS FOR VM PERFORMABILITY COMPONENT.

Transition	Condition	Description
EXT_FAIL	*(#OSPM_UP=0) OR (#NAS_NET_UP=0) OR (#DC_UP=0) AND* ((#VM_UP + #VM_DOWN + #VM_STRTD) $>$ 0)	Failure of physical machine or infrastructure
VM_Subs	*(#OSPM_UP>0) AND (#NAS_NET_UP>0) AND (#DC_UP>0)*	Physical machine and infrastructure working
DC_JRJC	*(#NAS_NET_UP=0) OR (#DC_UP=0) OR (#VM_WAIT=0) OR (#OSPM_UP=0)*	task rejection
DC_CH	*(#NAS_NET_UP>0) AND (#DC_UP>0) AND (#VM_WAIT>0) AND (#OSPM_UP>0)*	request acceptance to data center
DC_RJC	*(#NAS_NET_UP=0) OR (#DC_UP=0) OR (#VM_WAIT=0) OR (#OSPM_UP=0)*	task rejection

CLOUD_PART represent the load generation and the rejection of requests when all data centers are unavailable. All the *CLOUD_PART* of different VM performability submodels are merged. In sense that, just one *CLOUD_PART* is represented for the whole system (see Figure 5).

Places *CLTS* and *DC_CH* models the clients that are about to perform requests and the requests that entered into the system, respectively. It is assumed that each client requests a single VM. Exponential transition *USER_RQ* symbolizes the request process. Finally, *DC_RJC* denotes the request rejection when all data centers are unavailable. Both transitions have single server semantics (ss). It is assumed that the data center

is not available if the underlying infrastructure is broken or the servers are full.

It is important to stress that the guard expressions of *DC_IRJC*, *DC_CH* and *DC_RJC* can vary depending on number of physical machines and the number of data centers. In this case, we assume just one physical machine in one data center. However, this approach is generic enough to consider several machines in multiple data centers.

C. SPN block: transmission component

A VM should migrate to another data center whenever the current data center is full or the underlying infrastructure is failed. Moreover, Backup Server is responsible for migrating the VM image in case of disaster or network error.

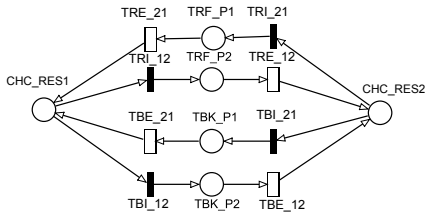

Fig. 4. Transmission component SPN model

Transmission component (Figure 4) has four exponentially distributed transitions that represent the VM data transfer and four immediate transitions that depict the enabling of the VM migration. *TRE_21* represents the data transfer from Data Center 2 to Data Center 1; *TRE_12* characterizes the migration from Data Center 1 to Data Center 2; *TBK_21* corresponds to data transfer from Backup Server to Data Center 1, and *TBK_12* characterizes the data transfer from Backup Server to Data Center 2. Although this component presents the same structure of that transmission component showed in the previous work [7], the guard expressions of these models are different.

TABLE II
GUARD EXPRESSIONS FOR TRANSMISSION COMPONENT.

Transition	Condition
TRI_12	(#VM_WAIT1=0 OR #OSPM_UP1=0) AND (#VM_WAIT2=0 OR #OSPM_UP2=0) AND NOT(((#VM_WAIT3=0 OR #OSPM_UP3=0) AND (#VM_WAIT4=0 OR #OSPM_UP4=0)) OR #NAS_NET_UP2=0 OR #DC_UP2=0)
TRI_21	(#VM_WAIT3=0 OR #OSPM_UP3=0) AND (#VM_WAIT4=0 OR #OSPM_UP4=0) AND NOT(((#VM_WAIT1=0 OR #OSPM_UP1=0) AND (#VM_WAIT2=0 OR #OSPM_UP2=0)) OR #NAS_NET_UP1=0 OR #DC_UP1=0)
TBI_12	(#BKP_UP=1 AND #NAS_NET_UP1=0 OR #DC_UP1=0) AND NOT(((#VM_WAIT3=0 OR #OSPM_UP3=0) AND (#VM_WAIT4=0 OR #OSPM_UP4=0)) OR #NAS_NET_UP2=0 OR #DC_UP2=0)
TBI_21	(#BKP_UP=1 AND #NAS_NET_UP2=0 OR #DC_UP2=0) AND NOT(((#VM_WAIT1=0 OR #OSPM_UP1=0) AND (#VM_WAIT2=0 OR #OSPM_UP2=0)) OR #NAS_NET_UP1=0 OR #DC_UP1=0)

Table II presents the guard expressions of transmission component. It is assumed that Data Center 1 contains the physical machines *OSPM_UP1* and *OSPM_UP2*, and, Data Center 2 includes *OSPM_UP3* and *OSPM_UP4*. The mean time to transmit *(MTT)* symbolizes the mean time to transmit one virtual machine from one location to another. The *MTT* depends on the physical link speed, the distance between the data centers and the VM size [19]. In this block, there are three *MTTs*: mean time to transmit a VM from the data center to another (*MTT_DCS*) and the mean times to transfer the VM image from Backup Server to Data Centers 1 and 2 (*MTT_BK1 and MTT_BK2*). The *MTT_DCS* parameter is associated to transitions *TRE_12* and *TRE_21*, while *MTT_BK1* and *MTT_BK2* are related to TBK_21 and TBK_12, respectively. All transitions mentioned before have single server semantics (ss).

V. CLOUD SYSTEM EXAMPLE WITH MULTIPLE DATA CENTERS

This section presents an architecture following Section III that assumes two data centers with two PMs each and up to N VMs per machine. Figure 5 presents the model, which is composed of four VM performability models, one transmission component as well several generic components. *OSPM_1* and *OSPM_2* represent the physical machines of Data Center 1, and *OSPM_3* as well as *OSPM_4* are the models related to PMs of Data Center 2. *DISASTER1* and *DISASTER2* models disasters in Data Centers 1 and 2, respectively. *NAS_NET_1* and *NAS_NET_2* corresponds to network devices of Data Center 1 and 2.

In this model, the dynamic behavior of the virtual machines is modelled by a transmission component and VM performability components. Table III presents the special guard expressions for this particular model.

TABLE III
CONFIGURATION DEPENDENT GUARD EXPRESSIONS OF FIGURE 5.

Transition	Condition	Description
DC_CH1	*((#NAS_NET_UP1>0 AND #DC_UP1>0) AND ((#VM_WAIT1>0 AND #OSPM_UP1>0) OR (#VM_WAIT2>0 AND #OSPM_UP2>0)))*	the current data center has operational infrastructure and an available server
DC_CH2	*((#NAS_NET_UP2>0 AND #DC_UP2>0) AND ((#VM_WAIT3>0 AND #OSPM_UP3>0) OR (#VM_WAIT4>0 AND #OSPM_UP4>0)))*	same description of *DC_CH1*
DC_RJC, *DC_IRJC1*, *DC_IRJC2*	NOT(*DC_CH1's* guard expression) AND NOT(*DC_CH2's* guard expression)	both data centers are unavailable

VI. CASE STUDY

To illustrate the feasibility of the proposed approach, we present a case study considering a set of cloud system scenarios in which the systems are deployed into two different data centers. We have conducted an availability evaluation considering distance between data centers and disaster mean time. We assume data centers located in the following pairs of cities: Rio de Janeiro (Brazil)-Brasilia (Brazil), Rio de Janeiro-Recife (Brazil), Rio de Janeiro-NewYork (USA), Rio de Janeiro-Calcutta (India) and Rio de Janeiro-Tokio (Japan). We assume that the Backup Server is located in São Paulo (Brazil).

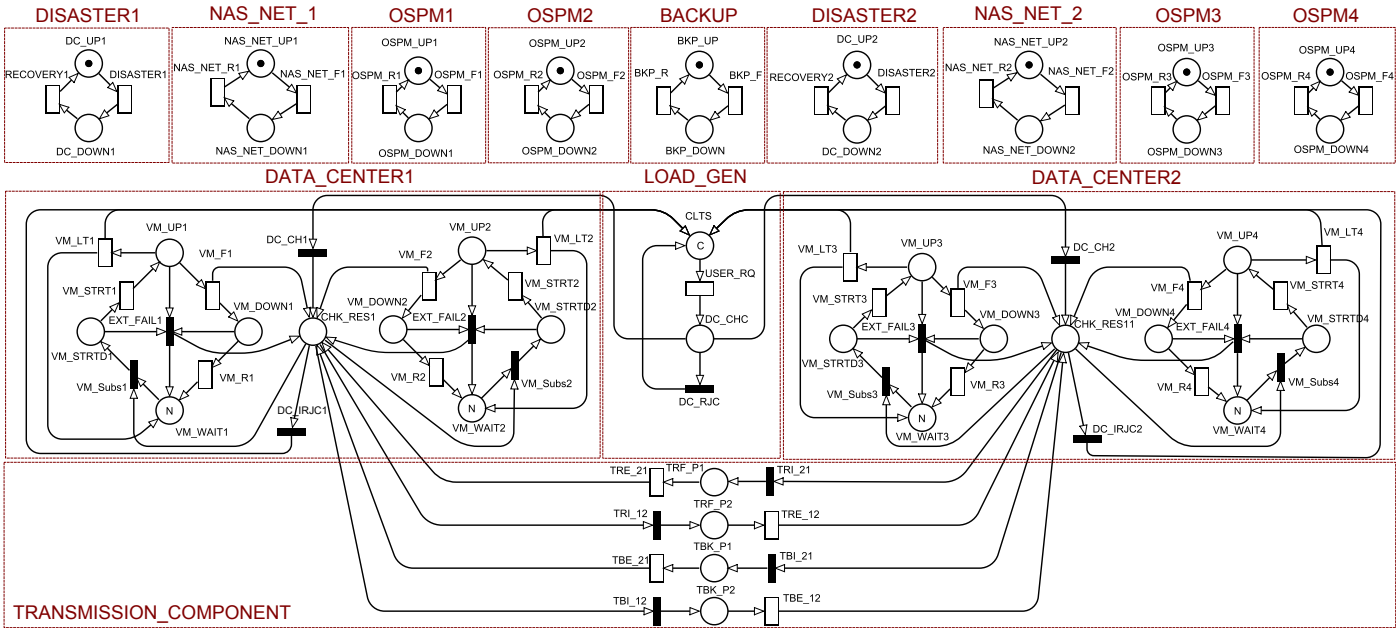

Fig. 5. SPN model representing four physical machines in two different data centers

To estimate the *MTT* value, we considered the approach presented in [20] that assesses network throughput based on the distance between the communication nodes. The equation associates a constant α with the network speed, which can vary from 0 (no connection) up to 1.0 (fastest connection). We have considered α as 0.45 and the size of VMs as 4GB.

We assume the mean time between disaster to be 100 years and the data center to take 30 days to be repaired. Moreover, a VM takes five minutes to start and the mean time between requests is half an hour. The mean time for using a VM is 720 hours. Previous work [7] presents the dependability parameters associated with the devices. Mercury-ASTRO [21] and TimeNET [22] tools have been adopted to perform the evaluation. To perform system evaluation we consider provider and client oriented metrics. The provider's oriented metric is the machine utilization in terms of maximum number of VMs (U) and the user's oriented metric is the probability of a task be completed without error (P). U is calculated as follows:

$$U = \frac{\sum_{j=1}^{M} E\{\#VM\_UPj\}}{M.N} \quad (1)$$

.

Where M corresponds to the number of PMs and N is the maximum number of VMs per physical machine. P corresponds to:

$$P = \frac{(\sum_{j=1}^{Pm} E\{\#VM\_UPj\}).(1/T)}{P\{\#CLTS > 0\}.(1/R)} \quad (2)$$

.

In which T and R correspond to the times associated with transitions *VM_LT* and *USER_REQ*. This expression divides the throughput of completed requests by the incoming requests throughput to compute the percentage of completed requests.

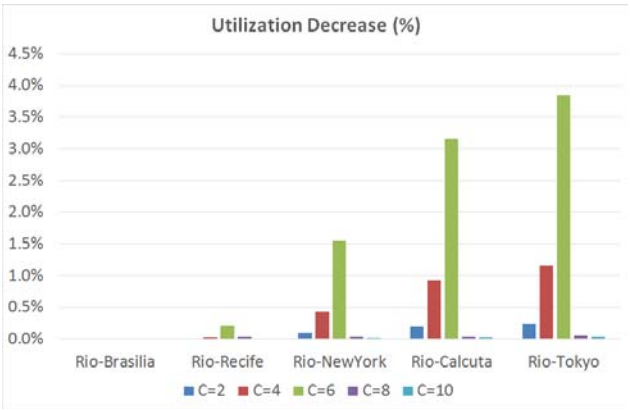

Fig. 6. Utilization decrease of different ditributed cloud configurations

Figure 6 shows utilization percentage decrease for each different pair of cities and number of clients (C). The results show that the utilization decreases with the distance if the system is not stressed. In case of high system load ($C > 6$), the effect of distance on the utilization is reversed.

A task rejection considering the proposed approach happens if the infrastructure is broken or the servers are full. Figure 7 presents the P percentage increase considering the same set of scenarios. It is possible to observe that, in this particular case, P rises with the distance between the data centers (until $C = 6$). With the growth of system load ($C > 6$), P decreases depending on the data centers distance. Therefore, for this particular system we can conclude that P is highly impacted by server utilization.

Table IV presents the values for U and P of the baseline architecture (Rio de Janeiro-Brasilia) considering different user loads.

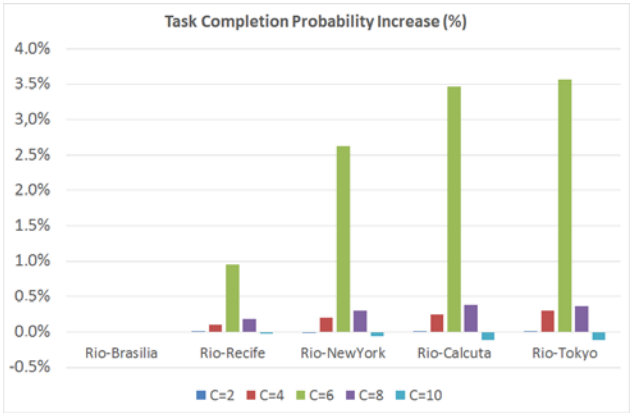

Fig. 7. P increase for different cloud configurations

TABLE IV
P AND U FOR THE BASELINE ARCHITECTURE (RIO DE JANEIRO-BRASILIA)

Client load	Utilization	P
C=2	0.2497989	0.9999299
C=4	0.4989833	0.9952341
C=6	0.7471487	0.9338905
C=8	0.9861435	0.4542866
C=10	0.9867157	0.2934411

VII. CONCLUSION

This work presented models for performability evaluation of cloud computing systems deployed into geographically distributed data centers taking into account disaster occurrence. The proposed technique allows the impact assessment of disaster occurrence, VM migration time and different client loads on system performability. Additionally, a case study is provided considering a set of data centers located in different places around the world.

The results demonstrated the influence of distance, client load and disaster occurrence on performability metrics. User and provider oriented metrics are adopted to show the trade-off between distance and user loads in the case study. As future research, we intend to evaluate costs and environmental impacts considering the proposed infrastructure. Additionally, a software tool will be create to enable non-specialized users to adopt this approach.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010. [Online]. Available: http://doi.acm.org/10.1145/1721654.1721672

[2] D. A. Menasc and P. Ngo, "Understanding cloud computing: Experimentation and capacity planning," 2009.

[3] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, pp. 7–18, 2010. [Online]. Available: http://dx.doi.org/10.1007/s13174-010-0007-6

[4] Amazon ec2. [Online]. Available: http://aws.amazon.com/ec2

[5] IBM smart business cloud. [Online]. Available: http://www-935.ibm.com/services/us/igs/cloud-development/

[6] *Hyper-V Live Migration over Distance*. [Online]. Available: http://goo.gl/GzlkNk

[7] B. Silva, P. R. M. Maciel, and A. Zimmermann, "Dependabilitymodels for designing disaster tolerant cloud computing systems," in *The Third International Workshop on Dependability of Clouds, Data Centers and Virtual Machine Technology (DCDV)*, 2013.

[8] F. Longo, R. Ghosh, V. Naik, and K. Trivedi, "A scalable availability model for infrastructure-as-a-service cloud," in *Dependable Systems Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*, june 2011, pp. 335 –346.

[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 *Proceedings of the 2010 IEEE 16th Pacific Rim International Symposium on Dependable Computing*, ser. PRDC '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 125–132. [Online]. Available: http://dx.doi.org/10.1109/PRDC.2010.30

[10] J. Araujo, R. Matos, P. Maciel, R. Matias, and I. Beicker, "Experimental evaluation of software aging effects on the Eucalyptus cloud computing infrastructure," in *Proceedings of the Middleware 2011 Industry Track Workshop*, ser. Middleware '11. New York, NY, USA: ACM, 2011, pp. 4:1–4:7. [Online]. Available: http://doi.acm.org/10.1145/2090181.2090185

[11] "Open source private and hybrid clouds from Eucalyptus," http://www.eucalyptus.com.

[12] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg, "Live wide-area migration of virtual machines including local persistent state," in *Proceedings of the 3rd international conference on Virtual execution environments*, ser. VEE '07. New York, NY, USA: ACM, 2007, pp. 169–179. [Online]. Available: http://doi.acm.org/10.1145/1254810.1254834

[13] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: A performance evaluation," in *Proceedings of the 1st International Conference on Cloud Computing*, ser. CloudCom '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 254–265.

[14] J. Dantas, R. Matos, J. Araujo, and P. Maciel, "An availability model for eucalyptus platform: An analysis of warm-standy replication mechanism," in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, oct. 2012, pp. 1664 –1669.

[15] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, ser. NSDI'05. Berkeley, CA, USA: USENIX Association, 2005, pp. 273–286. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251203.1251223

[16] P. Maciel, K. S. Trivedi, R. Matias, and D. S. Kim, *Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions*, ser. Premier Reference Source. Igi Global, 2011, ch. Dependability Modeling.

[17] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, ser. Wiley Professional Computing. Wiley, 1991.

[18] R. German, *Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets*. New York, NY, USA: John Wiley & Sons, Inc., 2000.

[19] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the tcp congestion avoidance algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 3, pp. 67–82, Jul. 1997. [Online]. Available: http://doi.acm.org/10.1145/263932.264023

[20] W. M. Les Cottrell and C. Logg, "Tutorial on internet monitoring and pinger at SLAC," Tech. Rep., 1996. [Online]. Available: http://www.slac.stanford.edu/comp/net/wan-mon/tutorial.html

[21] B. Silva, G. Callou, E. Tavares, P. Maciel, J. Figueiredo, E. Sousa, C. Araujo, F. Magnani, and F. Neves, "Astro: An integrated environment for dependability and sustainability evaluation," *Sustainable Computing: Informatics and Systems*, 2012.

[22] R. German, C. Kelling, A. Zimmermann, and G. Hommel, "TimeNET: a toolkit for evaluating non-markovian stochastic petri nets," *Performance Evaluation*, vol. 24, no. 1-2, pp. 69 – 87.