Capacity-Oriented Availability Model for Resources Estimation on Private Cloud Infrastructure

Carlos Melo*, Rubens Matos*, Jamilson Dantas* and Paulo Maciel*
*Informatics Center, Federal University of Pernambuco, Recife, Brazil
{casm3, rsmj, jrd, prmm}@cin.ufpe.br*,

***Abstract*—Predicting the amount of resources available to system's users has become a task of interest to services providers even with the advent of elastic cloud computing, because the number of resources is finite despite being virtually infinite on the customer view. This paper proposes a model to evaluate node's capacity in a cloud computing environment based on the amount of available hardware resources. By combining models to availability evaluation, such as reliability block diagrams, representing the operational infrastructure mode, and stochastic Petri net for capacity-oriented availability evaluation, we can determine the real amount of resources available at a predetermined time interval. Sensitivity analysis is also used to determine the component with highest impact in the metric of interest. The models, methods, and results of this research shall aid companies to plan the deployment and configuration of their services and cloud computing infrastructure.**

Keywords*—*Availability, Cloud Computing, Eucalyptus Platform, Stochastic Petri Net, Sensitivity Analysis, Capacity-oriented Availability.

I. Introduction

Cloud computing is a model increasingly adopted by companies around the globe which seek to reduce the high expenses of maintainability of the infrastructure itself [1].

The amount of services and platforms hosted in cloud computing environments had significant growth over the last few years, and with the elastic cloud computing, or Amazon EC2, turns the cloud computing model in an abstraction of hardware and software resources by those who consume them [2]. Resources in a cloud computing environment are dynamically allocated, virtualized, and enable high management control and flexibility. Storage, software platforms, and network services are provided and configured on demand over the Internet [3], what is a great benefit for companies that offer any service through the Internet, but these resources are only unlimited from the user point of view.

There are many open source tools and frameworks for cloud computing infrastructures deployment. They are particularly useful for companies that can afford the implementation costs of their own infrastructure, keeping full control over privacy, security, and accessibility of the stored information. A problem that still needs to be solved is to establish a relationship between the amount of hardware resources and deliverable resources to system users, such as virtual machines that are actually available to be allocated for providing a service through a cloud computing environment.

978-1-5090-5652-1/17/$31.00 ©2017 IEEE

This paper evaluates the capacity-oriented availability (COA) of nodes on a private cloud computing platform, by pointing out how many virtual machines will be available at a node, and describing the amount of resources that will be lost due to failures and repair routines. We also propose analytical models comprising Reliability Block Diagrams (RBD) and Stochastic Petri Nets (SPN). The RBD models enable computing metrics such as steady-state availability and annual downtime, whereas the SPN model provides means for capacity-oriented availability calculation. Using sensitivity analysis, we point out the component with highest impact on the COA metric. The modeling artifacts, methods, and results of this research shall be useful for people and organizations that want to offer services through the Internet. They can employ the proposed approach to perform capacity planning and define service level agreements (SLAs).

The remainder of this paper is organized as follows. Section II presents the works that are most related to this research. Section III shows the background used as a basis for the development of this study. Section IV shows the evaluation results for the proposed architecture. Section VI provides the case study considering capacity-oriented availability. Finally, Section VII presents the final remarks and future works.

II. Related Works

Some related research works propose analytical models for systems empowered by cloud computing, such as [4] and [5]; the authors provide models for evaluating a video streaming service hosted in private clouds. They also apply sensitivity analysis techniques to find the main components of the infrastructure and to measure the impact of warm standby redundancy in the system availability.

Availability models for the Eucalyptus cloud computing platform are proposed in [6] and [7], which enable computation of downtime, mean time to system failure and other similar metrics for distinct private cloud architectures. Later in [8] a study was carried out to a cost analysis, energy consumption, proposition of hierarchical models for the implementation of a public cloud with Eucalyptus platform.

Meanwhile, [9] uses hierarchical models, such like Markov chains with rewards in some places, to evaluate besides the steady-state availability, the capacity-oriented availability (COA). The main difference from this work to the one proposed here is that our COA model is an SPN general model that can be used to estimate any resource running on a machine, by replicating the SPN for each new machine in the infrastructure we are also able to identify which one have a resource in a failure state.

Our approach differs from [6], [7], [4], [5], and [8] by proposing a stochastic model to estimate the amount of resources available at cloud computing environment, and the impact of system failures in the delivery of these resources. A sensitivity analysis technique is also used to identify the parameters that deserve the greatest attention for the improvement of capacity-oriented availability metric.

III. Background

This section presents concepts regarding dependability, capacity-oriented availability, stochastic modeling, and sensitivity analysis, which are essential for a proper understanding of this paper.

A. Dependability

Dependability is the ability of a computer system to deliver a service that can be justifiably trusted [10]. This definition considers the user perception and the system behavior. Evaluating the dependability of a system is usually the result of evaluating at least one of its five attributes: reliability, availability, maintainability, integrity and safety [11].

The instantaneous availability is the probability that the system is operational at *t*, that is, $A(t) = P\left\{Z(t) = UP\right\} = E\left\{Z(t)\right\}, t \geq 0$, where $Z(t) = 1$ when the system is operational, and $Z(t) = 0$ when the system is down, as shown in Figure 1.

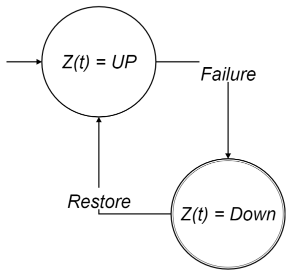

Fig. 1. System's States

Whenever the system fails ($Z(t) = 0$), a set of activities are conducted in order to allow the restoring process, these activities are generally made by a maintenance team and the time take by this team to repair the system is the time to restore (TTR). [12].

Steady state availability also called the long-run availability, is the limit of the availability function as time tends to infinity (Equation 1).

$$A = \lim_{t \to \infty} A(t), t \geq 0 \qquad (1)$$

This probability is also represented by a ratio between the Mean Time To Failure (MTTF) and Mean Time To Repair (MTTR) of the system (Equation 2), or even through system's Uptime and Downtime, as in (Equation 3).

$$A = \frac{MTTF}{MTTF + MTTR} \qquad (2)$$

$$A = \frac{E[Uptime]}{E[Uptime] + E[Downtime]} \qquad (3)$$

The MTTF for a system may be computed from Equation 4, where **R(t)** is the reliability for that system as a function of elapsed time. The Equation 5 provides a way of computing the MTTR from the values of MTTF, availability (A), and unavailability (UA = $1 - A$).

$$MTTF = \int_0^{\infty} R(t)\partial t \qquad (4)$$

$$MTTR = MTTF \times (\frac{UA}{A}), \qquad (5)$$

System downtime is the period when the service has been unavailable, and involves the time taken by maintenance team to receive the order and reach the site, also known as non-repair time (NRT), and the time taken to repair. Usually, the downtime is expressed by $Downtime = NRT + TTR$, while the system's uptime is the period when the system has been available to its users. Stochastic models can be used to analyze the presented metrics and the behavior of the system, which includes its operational modes. The next subsection shows some kinds of stochastic models that are employed for such purposes.

B. Stochastic Models

Stochastic models for reliability and availability evaluation are classified into combinatorial (or non-state space) and state-space models. Combinatorial models capture conditions that make a system fail (or to be working) regarding structural relationships between the system components. State-space models represent the system behavior (failures and repair activities) by its states and event occurrence expressed as rates or distribution functions. These models allow describing more complex relations between components of the system than combinatorial models do. For instance, state-space model support representing system's resource constraints, and the impact of component's failure in other components behavior. On the other hand, combinatorial usually demands less computation power to evaluate.

Reliability Block Diagram (RBD) and Fault Tree are among the most prominent combinatorial modeling formalisms, whereas Stochastic Petri Nets (SPN), Continuous Time Markov Chain (CTMC), and Stochastic Automata Networks are widely used state-space modeling formalisms [13], [14], [15], [12], [5], and [16].

C. Capacity-Oriented Availability

Capacity-oriented availability (COA) assesses how the system is delivered, therefore, does not consider only states of availability or unavailability, but the real impact of these factors in service delivery. The COA calculation considers pc_i as the operational processing capacity or the amount of resources available at any state s_i, while π_i describes the steady state availability for the s_i state, a set with all available states know as *US* it is also considered and the maximum processing capacity of the system *MPC*. Thus, we can calculate the capacity-oriented availability by Equation 6 [12]:

$$COA = \frac{\sum_{s_i \in US} pc_i \times \pi_i}{MPC} \tag{6}$$

D. Sensitivity Analysis

Sensitivity analysis techniques aim at determining the most influential factors in a system [17], [18]. With sensitivity analysis techniques we can detect the most critical system's components, where the manager or administrator should invest with the highest priority to improve the results of metrics of interest.

Some techniques for sensitivity analysis have been developed and reported in the literature, such as partial derivatives of the output metric function and analysis of variance for design of experiments (DoE). In this paper we employ a percentage difference technique for computing the sensitivity index $S_\theta(Y)$, which indicates the impact on a given measure Y caused by variations in an input parameter θ. Equation 7 presents how the percentage difference index is computed for a metric $Y(\theta)$, where $max\,\{Y(\theta)\}$ and $min\,\{Y(\theta)\}$ are the maximum and minimum output values, respectively, computed when varying the parameter θ over the range of its *n* possible values of interest. If $Y(\theta)$ is known to vary monotonically, so only the extreme values of θ (i.e., θ_1 and θ_n) may be used to compute $max\,\{Y(\theta)\}$; $min\,\{Y(\theta)\}$, and subsequently $S_\theta(Y(\theta))$ [18].

$$S_\theta(Y) = \frac{max\,\{Y(\theta)\} - min\,\{Y(\theta)\}}{max\,\{Y(\theta)\}} \tag{7}$$

While computing $S_\theta(Y)$, we fix the values of the other model parameters. The same process is carried out for every input parameter of the target model until analyzing all parameters and build a sensitivity ranking. The parameters with largest indices on that ranking have the most significant impact on model results. Such information may be used to establish priority for efforts that improve top-ranked parameters, as an efficient means to improve the whole system. Parameters that lie on last positions of the sensitivity ranking might be removed from the model to reduce its size or complexity.

IV. Cloud Computing Platform

For our analysis, we consider a generic cloud computing platform. Table I describes the main components of the cloud infrastructure that composes our platform.

TABLE I. Cloud Computing Platform Components

Component	Meaning
Platform Manager (PM)	Monitors and defines which cluster will provide new instances
Cluster Manager (CM)	Defines which physical machine belongs to a cluster
Block-based Storage (BBS)	Provides block storage for virtual machines
File-based Storage (FBS)	Object storage that allows users to exchange information about virtual machines
Instance Manager (IM)	Maintain virtual machines life cycle

Those cloud infrastructure components are present in the architecture that we evaluate. An architecture consisting of a frontend and a node, nodes on a cloud computing environment are the machines responsible for providing virtual machines and hosted services to system users, while the frontend machine is the responsible for access management. The following components run on the frontend machine: hardware (HW), operating system (OS), Platform Manager (PM), Cluster Manager (CM), Block-based Storage (BBS), and File-based Storage (FBS). HW and OS components are also present on the nodes, which also contain the hypervisor – responsible for management and allocation of resources for Virtual Machines (VM)–, the Instance Manager (IM).

The Figure 2 show how the cloud computing platform components are organized between the frontend and node in a baseline architecture.

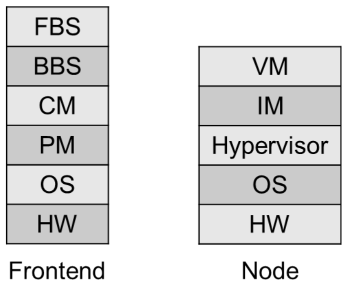

Fig. 2. Software Stack

V. Availability Models

This section describes the models proposed to evaluate system's availability. We represented the relationship between the cloud computing components using RBDs which follow the models proposed in [7] and [8]. RBDs were the models chosen to represent the availability in this work because there is no high dependence between the components, as techniques such as virtual machine migration and automated repair components are not considered here.

The values for system's failure (λ) and repair (μ) for each RBD block follows an exponencial distribution, and is equivalent to the inverse of time $\frac{1}{\lambda}$ and $\frac{1}{\mu}$ respectively.

The Figure 3 represents the architecture composed of frontend and node subsystems, and the virtual machine.

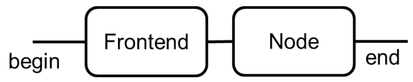

Fig. 3. RBD for entire system

The general availability for the system is given by equation: $A_{system} = A_{Frontend} \times A_{Node} \times A_{VM}$. The frontend's subsystem is presented in Figure 4.

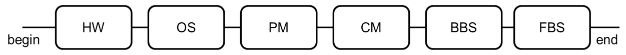

Fig. 4. RBD for frontend

The frontend's subsystem is composed by: hardware (HW), operating system (OS), and the other platform components:

PM, CM, BBS and FBS; And the closed expression for frontend's RBD subsystem is: $A_{Frontend} = A_{HW} \times A_{OS} \times A_{PM} \times A_{CM} \times A_{BBS} \times A_{FBS}$.

The node's RBD subsystem is presented in Figure 5. This RBD consists of hardware (HW), operating system (OS), the hypervisor (responsible for virtual machine management) and IM.

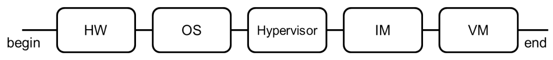

Fig. 5. RBD for a single node

The availability expression for node's RBD subsystem is given by: $A_{Node} = A_{HW} \times A_{OS} \times A_{Hypervisor} \times A_{IM}$.

VI. Case Study

This section provides a case study to evaluate the capacity-oriented availability of the proposed architecture. First, some input values are needed to evaluate our models. Those values used for the components of the platform and other software applications, such as failure and repair rate of database service, Apache and reboot time, are extracted from [7], [19], [5], and [20]. The availability evaluation period for the infrastructure that will provide virtual machines corresponds to 365 days or 8760 hours. The input values for all components are shown in Table II.

TABLE II. Input values for proposed scenarios

Component	MTTF	MTTR
HW	8760 h	100 min
OS	2893 h	15 min
CM	788.4 h	1 h
PM	788.4 h	1 h
BBS	788.4 h	1 h
FBS	788.4 h	1 h
Hypervisor	2990 h	1 h
IM	788.4 h	1 h
Virtual Machine	2880 h	69 s

The repair rate or virtual machine MTTR value is, in fact, the mean time to restart the virtual machine or instantiate a new one, which is equal to 69 seconds [5].

Based on the input parameters, we could extract as output values related to the availability of the RBDs representing frontend (Figure 4) and node (Figure 5), Table III summarizes the results of the evaluation.

TABLE III. Availability Results for Frontend and Node

Metric	Frontend	Node
Availability (%)	99.46	99.81
Annual Uptime (h)	8713.22	8743.53
Annual Downtime (h)	46.78	16.47

For the RBD representing the frontend subsystem (Figure 4) a value corresponding to 99.46% availability in the course of a year is presented, and an annual downtime of 46.78 hours. As for the subsystem represented by the node RBD (Figure 5), this value reaches 99.81%, ensuring an annual downtime of sixteen hours, almost a third of RBD presented by the frontend and less than a day of downtime; this is because of the number of applications running on each component. As a result of general availability, taking into account the model of Figure 3 the reach value was 99.27%.

The availability values for each state enable us to determine the capacity-oriented availability (COA), i.e., we can determine the real amount of available resources that is delivered to system's users. For such a purpose, first we need to know how many virtual machines our platform can handle, and what are our limitations to provide the virtual machines.

We chose the types of virtual machines employed in the Eucalyptus cloud computing platform as an example for delimiting the capacity of virtual machines in our infrastructure. In Eucalyptus, virtual machines are divided into families, each one with its characteristics [21]: Applications with low resources demand, with higher processor demand and with a greater amount of RAM demand.

The next step for our COA analysis is determining how many VMs we can run on our cloud infrastructure. The proposed environment is based on Dell PowerEdge T320 servers, with Intel Xeon E5-2420, six physical cores, and 12 threads, 1 TB of storage and 24 GB RAM. Another important information is that Eucalyptus default hypervisor (KVM) supports a virtual core per physical core or per thread available, which means that we can create 12 virtual machines of m1.medium type, for example.

Figure 6 depicts the stochastic Petri net model employed for the COA analysis, considering the node and the virtual machines available. The place **nVM_ON** represents the number of available virtual machines, and **nVM_OFF** place represents the number of virtual machines that went into a failure state. The transition **MTTF_VM** represents the time between virtual machines failures and the **MTTR_VM** the time between the restoration of our resources. The place **NODE_ON** indicates the availability of the node, while **NODE_OFF** represents its unavailability. The transition **MTTF_VM** and have inhibitor arcs that prevent them from firing if there is a token in **NODE_OFF**, so a VM cannot fail or be repaired if their hosting node is down.

The input values for the transitions are the same as in Table II except by immediate transition **emptier**, which upon node failure removes all tokens that represent the numbers of available and unavailable virtual machines. The inhibitor arc that connects **NODE_ON** to emptier, prevents it from firing if the node is available. As soon as the node is repaired, denoted by firing of transition **MTTR_NODE**, the initial number of available VMs is restored to the place **nVM_ON**.

We chose SPN instead of CTMCs because at least 130 states would be required to represent a single node architecture with the CTMC formalism. The proposed SPN can be replicated as many times as the number of nodes existing in the infrastructure. After any failure of a virtual machine, the system is still available, but it works in a degraded way. The COA analysis will relate to the expected amount of VMs running, considering the amount of nodes available, which means that the COA will be expressed in Equation 8:

$$COA = \frac{\sum_{i=1}^{MNVM} (P\{m(nVM_{on}) = i\} \times \pi_i)}{MNVM} \quad (8)$$

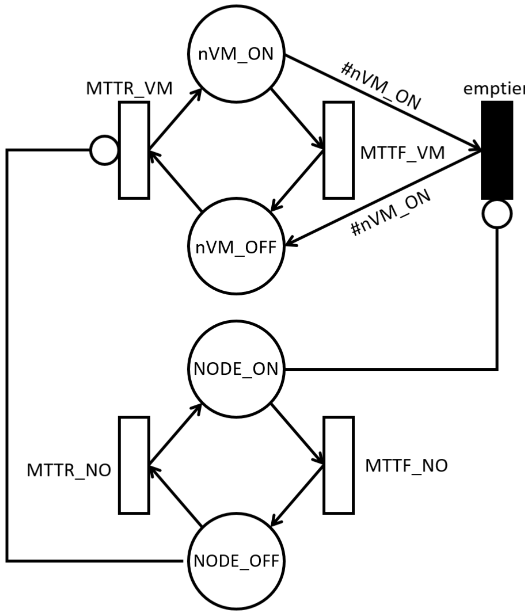

Fig. 6. SPN for COA Analysis considering one node

According to the expression, $m(nVM_{on})$ is the number of tokens in nVM_{on} place, which is the set of available virtual machines in the system in a s_i state, P is the probability of a number i of virtual machines be available, while $MNVM$ is the maximum number of virtual machines supported by the node.

For our case study, a node can handle 12 virtual machines of m1.medium type. The COA value obtained by the SPN is 99.78, which means that almost 0.22% of resources will be lost due to failures. Indicating that the failure events degrade the virtual machine provisioning in an extent that users cannot benefit from the storage and computing power provided by one VM on average, and only 11.97 virtual machines will be available to the users. Systems administrators should adapt their infrastructures for such an expected capacity loss, by oversizing the number of VMs and nodes about the number of expected users or providing alternative means for unavailability periods.

Following the Eucalyptus table with the virtual machines family, an architecture with the presented model evaluation, we can show Table IV, which represents the COA analysis for each possible virtual machine type.

The node can handle more virtual machines of m1.small, m1.medium and t1.micro types, also m1.small and m1.medium can handle as many users as m1.large, m1.xlarge and m2.xlarge virtual machines, being a better choice for our case study.

The next and last step of our capacity-oriented availability are to determine which components have a higher impact on the showed results: Virtual Machine or Node. Another percentage difference sensitivity analysis was made, following the same rules, the results obtained are presented in Table V.

Figure 7 shows the relationship between the node mean time to failure and the capacity-oriented availability, the longer the node takes to fail, the greater is the percentage of available resources.

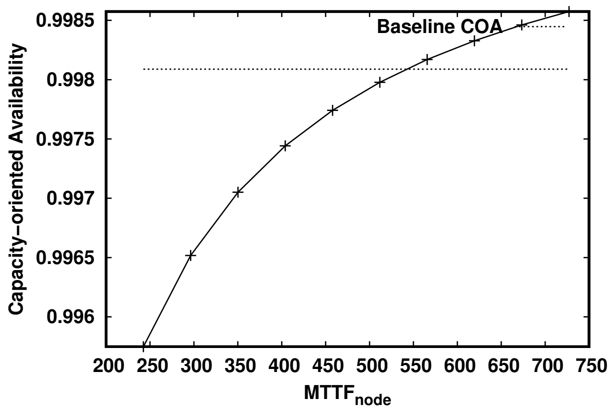

Fig. 7. Sensitivity Analysis for $MTTF_{node}$

Node's MTTF sensitivity index is followed closed by $MTTR_{node}$, which means that the higher the number of nodes, the larger the amount of resources available to users. The $MTTR_{VM}$ and $MTTF_{VM}$ have a lower impact on COA, but are no less important than other components, and must be considered.

VII. Final Remarks

This paper analyzed RBDs as availability planning artifacts for the ones who offers cloud computing infrastructures. The results showed an availability of 99.27% for the general system, and for the node subsystem, an annual downtime of 16 hours was found. Which enable us to realize a COA analysis, we were able to see a degradation in the provisioning of VMs on the node, losing capabilities due to failures and respective repairing periods. A sensitivity analysis was also made and identify that the node was the component with higher impact on COA value. As a future work, we shall analyze the trade-off between energy consumption and the COA of a cloud computing environment.

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, "Above the clouds: A berkeley view of cloud computing," UC Berkeley Reliable Adaptive Distributed Systems Laboratory, Tech. Rep. UCB/EECS-2009-28, Feb. 2009.

[2] L. M. Vaqueiro, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: Towards a cloud definition," *Computer Communication Review*, vol. 39, pp. 50–55, 2009.

[3] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop*, Austin, Nov. 2008.

[4] R. M. D. Melo, M. C. Bezerra, J. Dantas, R. Matos, I. J. D. M. Filho, and P. Maciel, "Redundant vod streaming service in a private cloud: Availability modeling and sensitivity analysis," *Mathematical Problems in Engineering*, vol. 2014, p. 14, 2014.

[5] M. C. Bezerra, R. Melo, J. Dantas, P. Maciel, and F. Vieira, "Availability modeling and analysis of a vod service for eucalyptus platform," in *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, Oct 2014, pp. 3779–3784.

[6] 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.

TABLE IV. EUCALYPTUS - COA

VM Types	Number of VM per Node	Number of VM lost due to failures	COA
m1.small	12	0.05	11.97
m1.medium	12	0.05	11.97
m1.large	6	0.03	5.98
m1.xlarge	6	0.03	5.98
m2.xlarge	3	0.02	2.99
m3.2xlarge	3	0.02	2.99
c1.medium	6	0.03	5.98
c1.xlarge	6	0.03	5.98
cc1.4xlarge	1.5	0.01	1.49
cc2.8xlarge	0.75	-	-
m2.xlarge	6	0.03	5.98
m2.2xlarge	6	0.03	5.98
m2.4xlarge	1.5	0.01	1.49
cr1.8xlarge	0.75	-	-
t1.micro	12	0.05	11.97

TABLE V. SENSITIVITY INDEX OF COA

Parameter	S(COA)
$MTTF_{NODE}$	2.83×10^{-3}
$MTTR_{NODE}$	1.66×10^{-3}
$MTTR_{VM}$	2.32×10^{-4}
$MTTF_{VM}$	8.60×10^{-6}

[7] ——, “Models for dependability analysis of cloud computing architectures for eucalyptus platform,” *International Transactions on Systems Science and Applications*, vol. 8, pp. 13–25, 2012.

[8] ——, “Eucalyptus-based private clouds: availability modeling and comparison to the cost of a public cloud,” *Computing*, vol. 97, no. 11, pp. 1121–1140, 2015. [Online]. Available: http://dx.doi.org/10.1007/s00607-015-0447-8

[9] D. S. Kim, F. Machida, and K. S. Trivedi, “Availability modeling and analysis of a virtualized system,” in *Dependable Computing, 2009. PRDC ’09. 15th IEEE Pacific Rim International Symposium on*, Nov 2009, pp. 365–371.

[10] A. Avizienis, J. Laprie, B. Randell, and C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing,” *IEEE Transactions on Dependable and Secure Computing*, vol. 1, pp. 11–33, 2004.

[11] A. Avižienis, J. Laprie, B. Randell, and U. of Newcastle upon Tyne. Computing Science, *Fundamental Concepts of Dependability*, ser. Technical report series. University of Newcastle upon Tyne, Computing Science, 2001. [Online]. Available: https://books.google.com.br/books?id=cDkmGwAACAAJ

[12] R. M. Paulo Maciel, Kishor Trivedi and D. Kim, “Dependability modeling,” in *Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions*, 2011.

[13] M. Malhotra and K. Trivedi, “Power-hierarchy of dependability-model types,” *Reliability, IEEE Transactions on*, vol. 43, no. 3, pp. 493–502, Sep 1994.

[14] K. S. Trivedi, S. Hunter, S. Garg, and R. Fricks, “Reliability analysis techniques explored through a communication network example,” 1996.

[15] I. Software, “Reliability block diagram,” http://www.reliabilityeducation.com/rbd.pdf, 2007, [Online; accessed 26-September-2015].

[16] S. Garg, P. A, T. M, and K. S. Trivedi, “Analysis of software rejuvenation using markov regenerative stochastic petri net,” in *Proc. In: Sixth International Symposium on Software Reliability Engineering, (ISSRE’95)*, Paderborn, 1995, pp. 180–187.

[17] P. M. Frank, *Introduction to Sensitivity Analysis*. Academic, 1978.

[18] R. Matos, J. Araujo, D. Oliveira, P. Maciel, and K. Trivedi, “Sensitivity analysis of a hierarchical model of mobile cloud computing,” *Elsevier Journal Simulation Modelling Practice and Theory*, vol. 50, pp. 151–164, Jan. 2015.

[19] I. Costa, J. Araujo, J. Dantas, E. Campos, F. A. Silva, and P. Maciel, “Availability evaluation and sensitivity analysis of a mobile backend-as-a-service platform,” *Journal Quality and Reliability Engineering International*, 2015.

[20] E. Campos, R. Matos, A. Pereira, F. Vieira, and P. Maciel, “Stochastic modeling of auto scaling mechanism in private clouds for supporting performance tunning,” in *Proceedings of the IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC’15)*, Hong Kong, China, 2015.

[21] HP, “Virtual machine types,” 2015, [Online; accessed 19-December-2015]. [Online]. Available: https://docs.hpcloud.com/eucalyptus/4.2.2/user-guide