An Analytical Model for Availability Evaluation of Cloud Service Provisioning System

Fatimah M. Alturkistani
Information System Department
Imam Mohammed bin Saud University
Riyadh, Saudi Arabia

Saad S. Alaboodi
Information System Department
King Saud University
Riyadh, Saudi Arabia

***Abstract*—Cloud computing is a major technological trend that continues to evolve and flourish. With the advent of the cloud, high availability assurance of cloud service has become a critical issue for cloud service providers and customers. Several studies have considered the problem of cloud service availability modeling and analysis. However, the complexity of the cloud service provisioning system and the deep dependency stack of its layered architecture make it challenging to evaluate the availability of cloud services. In this paper, we propose a novel analytical model of cloud service provisioning systems availability. Further, we provide a detailed methodology for evaluating cloud service availability using series/parallel configurations and operational measures. The results of a case study using simulated cloud computing infrastructure illustrates the usability of the proposed model.**

Keywords—Cloud computing; availability evaluation; series and parallel configuration; infrastructure as service

I. Introduction

Infrastructure as service (IaaS) cloud providers, such as Amazon Web Service and Microsoft Azure, deliver on-demand computational resources from large pools of equipment installed in a cloud service provider's data centers. The requests submitted by the cloud customers are provisioned and released if the cloud has enough available resources. Conversely, customers expect cloud services to be available whenever they need them, just like electricity or telephone connectivity. This expectation requires cloud service providers to regularly assess their infrastructure for probable failures and reduce the amount of time needed to recover from such failures.

Typically, a cloud service provider offers a service level agreement (SLA) stipulating the service provider's performance and quality in several ways. For example, an SLA may include a metric specifying the availability of the cloud service. Before committing an SLA to the cloud customers, the service provider needs to carry out an availability assessment of the infrastructure on which the cloud service is hosted [1], [2]. Most of the cloud providers offer approximately 99.99% of availability in their SLA. However, real data shows that the actual value of the availability of these providers is much lower [3], [4].

Hence, to reduce the overall cloud downtime and to provide a reliable estimate of service availability, cloud service providers need to assess the availability characteristics of their data centers in responsible and dependable manner. This assessment can be done through controlled experiments, large-scale simulations, and via analytical models [5], [1]. In a massive system such as cloud computing, conducting repetitive experiments or simulations is likely to be costly and time-consuming. Although analytical models can be cost and time-effective, accurate analytical modeling must deal with a large number of system states, leading to the state space explosion problem [6].

The primary contribution of this study is to propose a novel analytical model for evaluating the availability of cloud service provisioning systems focusing on IaaS. The proposed model is architecture-based; it relies on National Institute of Standards and Technology - Cloud Computing Reference Architecture (NIST-CCRA), the well-known cloud computing reference architecture [7]. NIST-CCRA provides an abstraction for cloud service provisioning system that can be used to model the logical interaction of failures within the system.

Consequently, availability is evaluated at two levels: the system-level and the component-level. At the system-level, reliability block diagrams (RBDs) are used to model the system's failures by considering series/parallel arrangements of the cloud components/subsystems. At component-level, availability is determined by probabilistic model and operational measures. Failure and repair data are modeled and analyzed using probability distributions and statistical inferences. Then, operational measures are derived and used to estimate component's availability.

A simulation approach is used to develop and verify the proposed analytical model. CloudSim [8] is used to simulate cloud infrastructure and the underlying components, while FTCloudSim [9], an extension of CloudSim, is used to simulate different failure scenarios using the fault injection technique. Also, BlockSim [10] and Weibull++ [11] are used for availability analysis and interpretation of results.

The rest of the paper is structured as: Section 2 presents relevant background information. Section 3 describes the proposed analytical model, and Section 4 presents conclusions and suggested future work.

II. Background

A. NIST-based Cloud Service Provisioning System

According to NIST-CCRA, there are explicit processes and activities that cloud service providers need to perform to ensure reliable cloud service provisioning. Through service orchestration, a cloud service provider operates the underlying cloud service infrastructure that supports its customers. The NIST defines service orchestration as “the composition of system components to support the cloud provider activities in arrangement, coordination, and management of computing resources in order to provide cloud services to cloud consumers” [11].

Service orchestration has three main components, which are arranged in layers: 1) the service layer (SL); 2) the resource abstraction and control layer (RACL); and 3) the physical layer (PL). The horizontal positioning of these layers reflects the relationships between them; upper-layer components depend on adjacent lower-layer components to provide a service. For instance, the RACL provides virtual cloud resources on top of the PL and supports the SL.

Likewise, in the SL, services can be modeled as three-layered components representing three types of services that have been universally accepted: 1) software as a service (SaaS); 2) platform as a service (PaaS); and 2) IaaS. A cloud service provider may define interface points in all three service models or just a subset. For instance, the platform component (i.e., PaaS) can be built upon the infrastructure component (i.e., IaaS) to support the software component (i.e., SaaS) where cloud service interfaces are exposed.

Although NIST-CCRA does not represent the system architecture for a particular cloud system, a specific cloud service provisioning system such as an IaaS provider or an IaaS broker can be modeled using NIST-CCRA [12]. Pereira, *et al*. [13] used NIST-CCRA to design a cloud-based architecture by refining the system’s logical architecture. The suggested method involves 1) the selection of the NIST architectural component for which the respective coverage in the system’s logical architecture needs to be analyzed; 2) analysis of the system’s components into logical architecture including the respective architectural elements (AEs); and 3) the refinement and development of a new logical architecture in the cloud context by mapping the system’s AEs to NIST-CCRA AEs.

B. Availability Evaluation in Cloud Computing

Cloud architecture has been studied using various techniques from reliability theory including RBDs, stochastic Petri nets (SPNs), fault trees, and Markov chains [13]-[17]. The availability of cloud computing architecture has been modeled in various ways using RBD techniques. In addition, analytical modeling has been used to estimate the availability of cloud system architectures including virtualized simple architecture and virtualized redundant architecture [18].

By considering the virtualization in the cloud, RBDs can also be applied to full virtualization, OS virtualization, and paravirtualization (see Fig. 1).

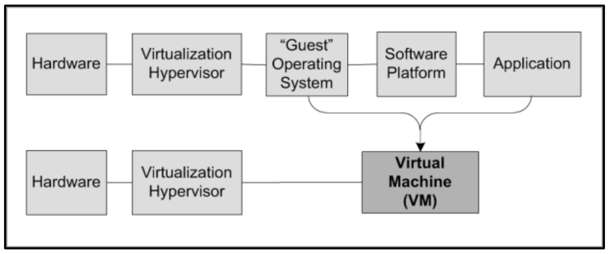

Fig. 1. Canonical virtualization *RBD.*

However, the dynamic nature of cloud computing requires the use of more rigorous modeling such as Markov modeling. Thus further analysis of availability in the context of system-level virtualization is needed.

Therefore, Dantas, *et al*. [19] used a hierarchical heterogeneous model based on RBD and a Markov reward model to describe non-redundant and redundant Eucalyptus architectures. Consequently, closed-form equations are obtained to compute the availability of those systems according to the rule of composition of series and parallel components. With respects to virtualization, availability model of a non-virtualized and virtualized system is presented using a hierarchical analytic model in which fault tree is used in the upper-level and homogeneous continuous-time Markov chains are used in the lower-level [20].

In another study, Silva, *et al*. adopted a hybrid modeling approach to deal with the complexity of the cloud system; RBDs are used for system-level dependability, whereas operational measures, such as mean time to failure (MTTF) or mean time between failure (MTBF), and mean time to repair (MTTR), are obtained for subsystem and component-level dependability.

III. Modelling Formalism

A. System Representation and Basic Assumptions

Based on NIST-CCRA, let us consider the following two cloud implementation scenarios that can be used by a cloud service provider. In the first scenario, a cloud service provider may implement a high-level service model (i.e., SaaS) by using the interface points defined in the lower layers. For example, SaaS applications can be built on top of PaaS components, and PaaS components can be built on top of IaaS components. A real-world example of this is Google’s cloud offerings. They offer a variety of SaaS products (e.g., Gmail, Google Search, Google Maps, Google Apps) using PaaS components (Google App Engine) that are run operationally on Google’s cloud IaaS (Google’s cloud platform) [7], [21]. As per NIST-CCRA, the dependency relationships among SaaS, PaaS, and IaaS components are represented graphically as components stacked on top of each other (see Fig. 2 (a)). A stack is a clearly defined structure that implies a series of interconnected systems that transport data between each other to provide a certain function or service [22]. Therefore, similar to modeling large-scale distributed systems [23], [24] and other cloud platforms [19], [25], [26] the cloud service provisioning system is represented as a simple series system using an RBD (see Fig. 2 (b)).

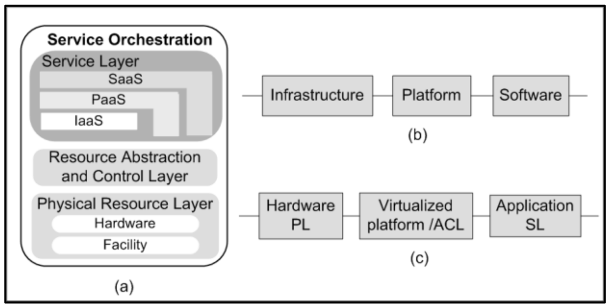

Fig. 2. NIST-CCRA service orchestration model.

In the second scenario, a cloud service provider may choose to provide an SL without the support of the lower-layer interface points. For example, a SaaS application can be implemented and hosted directly on top of cloud resources rather than using an IaaS virtual machine. A real-world example is salesforce.com, which provides both SaaS and PaaS products. The SaaS layer is built using the well-defined interface components from the PaaS. However, in this case, no IaaS layer is offered. SaaS is run directly on the resource abstraction layer with no explicit IaaS components. As per NIST-CCRA, the angling of the components indicates that each of the service components can stand alone and can be implemented directly on top of the cloud RACL and PL [7]. Hence, the cloud service provisioning system is represented as a simple series system using an RBD (see Fig. 2 (c)).

B. Cloud Service Provisioning Availability Model

In this model, the availability of the system is specified concerning the availability of the various components. Following a bottom-up approach, the availability at the component-level is determined using operational measures (i.e., MTBF and MTTR).

The logical relationship between individual components is considered to estimate the system-level availability, and it is expressed graphically using RBD. Table 1 set the definition of the notations that have been used in the availability modeling.

Let us consider a cloud service provisioning system denoted by CSP that consists of a set of subsystems $C = \{C_i, i = 1,2,3\}$ in which CSP success depends on the success of every subsystem C_i. Given the serial configuration as shown in Fig. 2, the availability of CSP denoted by A_{CSP} is written by [27]

$$A_{CSP} = \prod_{i=1}^{3} A_i, \tag{1}$$

Where, A_i is the availability of subsystem i. Recall that in NIST-CCRA, a cloud service provisioning system consists of three ordered layers, PL, RCAL, and SL.

Likewise, each subsystem C_i consists of a set of components $C_i = \{C_{i,j}, i = 1,2,3, j = 1,..,n_i\}$, where $C_{i,j}$ denotes the j^{th} component of the i^{th} subsystem, and n_i denotes the total number of components in subsystem i. Let us assume that the success of each subsystem C_i depends on the success of every individual component $C_{i,j}$.

TABLE. I. DEFINITION OF NOTATIONS

Notation	Definition
$_{SP}$	Cloud service provisioning system's overall availability value
$_i$	Availability of subsystem i
$_j$	Availability of component $C_{i,j}$
$_j$	Component j at subsystem i
$_i$	Total number of components at subsystem i
	$= \{C_{i,j}, i = 1,2,3, j = 1,..,n_i\}$, set of components for subsystem i
$X_{i,j}$	A random variable representing the time to failure of the j^{th}component of the i^{th} subsystem, $i = 1,2,3$ and $j = 1,..,n_i$
(t)	Availability of component $C_{i,j}$as a function over time
(t)	Failure density function for component $C_{i,j}$
$F_{i,j}$	Mean time to failure for component $C_{i,j}$
$R_{i,j}$	Mean time to repair for component $C_{i,j}$
$_j$	A random variable representing the time to repair of the j^{th}component at the i^{th} subsystem, $i = 1,2,3$ and $j = 1,..,n_i$
(t)	Repair density function for component $C_{i,j}$

Given this serial configuration, the availability of subsystem C_i denoted by A_i is given by

$$A_i = \prod_{j=1}^{n_i} A_{i,j}. \tag{2}$$

At the component-level, probability distributions are used for modeling operational data such as time to failure (TTF) and time to repair (TTR). Failure data can be used to make statements about the probability model, either in terms of the probability distribution itself or in terms of its parameters or some other characteristics.

Availability is the probability of a system/component being up (i.e., providing the service) at a specific instant of time t [24].

It is often expressed using (3), with many different variants [28], [29].

$$A = \frac{Uptime}{Uptime + Downtime}, \tag{3}$$

Where, *Uptime* refer to a capability to perform the task and *Downtime* refer to not being able to perform the task. However, the classification of availability is somewhat flexible and is largely based on the types of downtimes used in the computation and on the relationship with time. This study focused on inherent availability to determine the component-level availability. Inherent availability is the steady-state availability when considering only the corrective maintenance downtime of the system. Usually, this is the type of availability that companies use to report the availability of their products (e.g., computer servers) because they see downtime other than actual repair time as out of their control and too unpredictable.

Inherent availability used some operational measures from reliability theory: MTTF or MTBF and MTTR [18].

Now, let us assume that components failure data (i.e., TTF and TTR) is collected and preliminary analysis is performed using descriptive statistics and statistical inferences. Statistical inferences aim to draw inferences from the collected data in a meaningful way concerning some characteristics failure rates, MTTF, MTTR and related quantities. As probabilistic assumptions regarding the failure data play an important role in reliability and availability analysis [30], failure data (or at least assume the means of the sample data) usually assumed to follow well-known distributions (e.g., exponential, Weibull, lognormal).

Let $MTTF_{i,j}$ be a mean time to failure of the j^{th}component at the i^{th} subsystem, $i = 1,2,3$ and $j = 1,..,n_i$, $X_{i,j}$ is the random variable that represents the TTF of that component, and $f_{i,j}(t)$ is the probability density function of the component's failure time, then $MTTF_{i,j}$ is defined as the expected value of the random variable $X_{i,j}$ such that [24], [29]

$$MTTF_{i,j} = E\left[X_{i,j}\right] = \int_0^\infty t\, f_{i,j}(t)dt. \tag{4}$$

For a repairable component, $MTBF_{i,j}$ is used rather than $MTTF_{i,j}$ and defined similarly.

On the other hand, MTTR is used to measure the amount of time it takes to get a component running again after a failure [18]. Let $Y_{i,j}$ is the random variable that represents the TTR of the j^{th} component at the i^{th} subsystem, $i = 1,2,3$ and $j = 1,..,n_i$, and $g_{i,j}(t)$ is the probability density function of the component's repair time, then the component's $MTTR_{i,j}$ can be defined as [24]

$$MTTR_{i,j} = E\left[Y_{i,j}\right] = \int_0^\infty t g_{i,j}(t)dt. \tag{5}$$

Now, let us consider $C_{i,j}$ which represent the j^{th} component at i^{th} subsystem, $i = 1,2,3$ and $j = 1,..,n_i$, the availability of component $C_{i,j}$ denoted by $A_{i,j}$ is given by [18], [31]

$$A_{i,j} = \frac{MTTF_{i,j}}{MTTF_{i,j} + MTTR_{i,j}}. \tag{6}$$

C. *Modeling Cloud System Availability with Redundant Components*

Redundancy in cloud service provision system (e.g., hardware redundancy, software redundancy, and application redundancy) can also be modeled using RBD. The simplest example of redundancy could be achieved by combining two components in a parallel subsystem (i.e., server, storage, and virtual machine). The subsystem only fails if both components fail.

Let us consider n_i components in a cloud subsystem i with parallel composition, the subsystem availability A_i can be computed as follows

$$A_i = 1 - \prod_{j=1}^{n_i}(1 - A_{i,j}), \tag{7}$$

Where, $A_{i,j}$ is the availability of j^{th} individual component within the subsystem i . Further, the availability of more complex configuration (e.g., series-parallel configuration) can be obtained by combining the rules defined for series and parallel configuration [25].

D. *Numerical example*

To demonstrate NIST-based availability modeling and analysis numerically, let us consider the RBD of the IaaS provisioning system (IPS) depicted in Fig. 3. The objective is to obtain the average availability of the system after one year of operation (i.e., 8,760 hours).

The availability of the IPS, denoted by A_{IPS}, is the product of the availabilities of its subsystem i such that

$$A_{IPS} = A_1 \times A_2 \times A_3. \tag{8}$$

Where, A_1, A_2, and A_3 represent the availability of the hardware, virtualized platform, and application subsystems, respectively.

The availability of the hardware subsystem, denoted by A_1, is determined by the availability of its constituent components: power, network, storage, processor, and memory. Hence, A_1 is written as

$$A_1 = A_{1,1} \times A_{1,2} \times A_{1,3} \times A_{1,4} \times A_{1,5}. \tag{9}$$

Where, $A_{1,1}$, $A_{1,2}$, $A_{1,3}$, $A_{1,4}$, and $A_{1,5}$ represent the availability of power, network, storage, processor, and memory, respectively.

Likewise, the availability of virtualized platform A_2 is given by

$$A_2 = A_{2,1} \times A_{2,2} \times A_{2,3} \times A_{2,4}. \tag{10}$$

Where, $A_{2,1}$, $A_{2,2}$, $A_{2,3}$, and $A_{2,4}$ represent the availability of the hypervisor, VM, virtualized operating system, and middleware.

At component-level, let us assume that probability distributions model of failure data (i.e., TTF and TTR) are used to estimate the MTBF and MTTR for each component using (4) and (5), respectively to be shown in Fig. 4. Consequently, component availability is determined by substituting the values of MTBF and MTTR for the component in (6). Then, at the subsystem level, availability is analyzed based on system RBDs as follow:

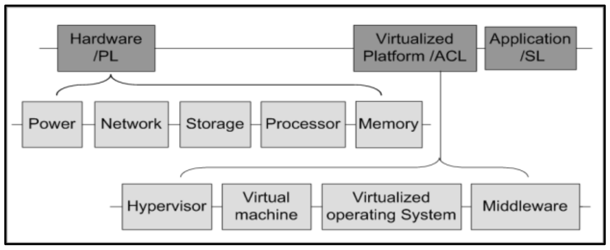

Fig. 3. IaaS provisioning system RBD.

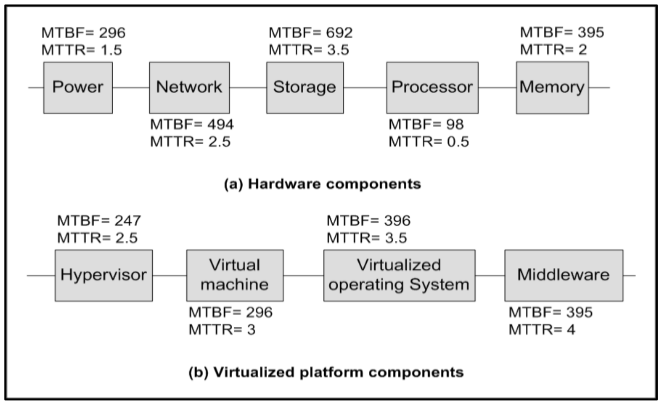

Fig. 4. MTBF and MTTR for IPS components.

Hardware availability A_1 is determined by substituting the values of its constituent component availabilities in (9), leading to

$$\begin{aligned} A_1 &= [MTBF_{1,1}/\left(MTBF_{1,1} + MTTR_{1,1}\right)] \\ &\times [MTBF_{1,2}/\left(MTBF_{1,2} + MTTR_{1,2}\right)] \\ &\times [MTBF_{1,3}/\left(MTBF_{1,3} + MTTR_{1,3}\right)] \\ &\times [MTBF_{1,4}/\left(MTBF_{1,4} + MTTR_{1,4}\right)] \\ &\times [MTBF_{1,5}/\left(MTBF_{1,5} + MTTR_{1,5}\right)] \end{aligned}$$

$$\begin{aligned} &= [296/(296 + 1.5)] \\ &\times [494/(494 + 2.5)] \\ &\times [692/(692 + 3.5)] \\ &\times [98/(98 + 0.5)] \\ &\times [395/(395 + 2)] \end{aligned}$$

$$=0.975.$$

Virtualized platform availability A_2 is determined by substituting the values of components availabilities in (10), leading to

$$\begin{aligned} A_2 &= [MTBF_{2,1}/\left(MTBF_{2,1} + MTTR_{2,1}\right)] \\ &\times [MTBF_{2,2}/\left(MTBF_{2,2} + MTTR_{2,2}\right)] \\ &\times [MTBF_{2,3}/\left(MTBF_{2,3} + MTTR_{2,3}\right)] \\ &\times [MTBF_{2,4}/\left(MTBF_{2,4} + MTTR_{2,4}\right)] \end{aligned}$$

$$\begin{aligned} &= [247/(247 + 2.5)] \\ &\times [296/(296 + 3)] \\ &\times [396/(396 + 3.5)] \\ &\times [395/(395 + 4)] \end{aligned}$$

$$=0.961.$$

For application, let us assume that MTBF=329 and MTTR=5, then application availability A_3 is determined by substituting the values of the MTBF and MTTR of the application in (6), leading to

$$A_3 \;\; 29/(329 + 5)$$

$$.986.$$

Subsequently, by substituting the values of A_1, A_2, and A_3, in (8), the availability of IPS is given by

$$A_{IPS} \;\; .975 \times 0.961 \times 0.986$$

$$.924.$$

Using the RBD model and all the failure and repair characteristics, the IPS is simulated for 30,000 hours of operation (using BlockSim). After running the simulation for 30,000 hours, the relevant metrics are obtained. The point availability after one year of operation (i.e., $t = 8{,}670$) is estimated to be 93.6000%, whereas the mean availability after one year of operation is estimated to be 92.3134% (see Fig. 5), which corresponds to the analytical result obtained for mean system availability (i.e., $A_{IPS} = 0.924$). The subsystems mean availabilities are estimated to be 98.47% for application, 79.50% for hardware, and 69.19% for virtualized platform; these results correspond with those obtained using the analytical method (see Fig. 6).

Modeling and analyzing the IPS availability often carries significant value in boosting the efforts to improve availability, performing a trade-off analysis in system design or suggesting the most efficient way to operate and maintain the system.

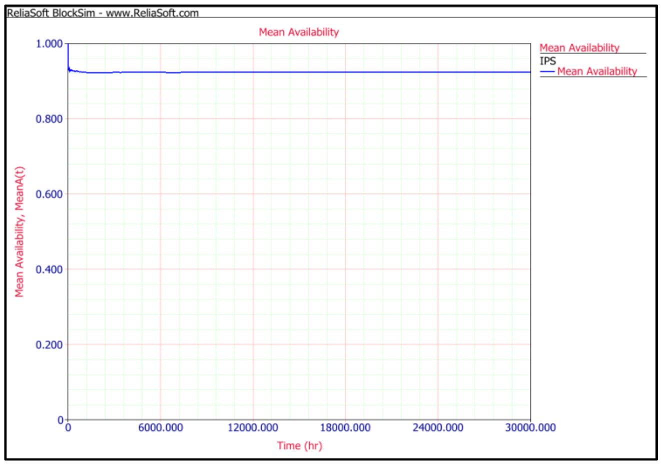

Fig. 5. Simulation results of IPS mean system availability.

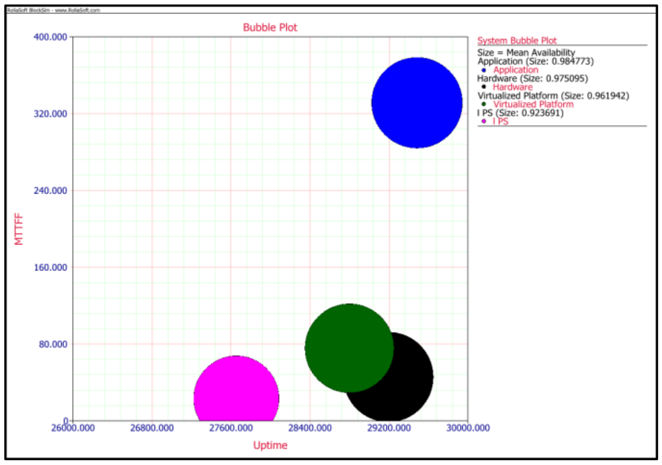

Fig. 6. Bubble plot of IPS subsystems mean availability with respects to mean time to first failure (MTTFF) and uptime.

IV. Testing and Validation

A. Approach

Because obtaining real-life failure data is extremely difficult as a result of the sensitive nature of these data, a simulation approach is used to test and validate the proposed model (Fig. 7). First, CloudSim is used to create a computerized duplication of real cloud infrastructure that is suitable for modeling probabilistic systems. Host, virtual machine (VM), and cloudlet are considered to be the infrastructure components that constituted the cloud service provisioning system.

For the experimental simulation, one data center is considered with different numbers of hosts using a 16-port fat-tree data center network and a corresponding number of VMs. Simultaneously, a Linux environment with x86 architecture is used as the operating environment and Xen as the virtual machine manager (VMM).

By investigating the architectural model of the cloud service provisioning system in CloudSim, the availability of IaaS requires an available host, VM, and cloudlet. Thus, the simulated IaaS cloud system can be modeled in a simple series system. Second, based on the previous study conducted by Zhou, *et al*. [8], multiple host failures are injected into the simulated system. Likewise, by considering some failure scenarios described by Nita, *et al*. [31], VM and cloudlet failures are introduced, and then failure and repair data are collected for availability analysis. Next, for the purpose of component availability modeling, Weibull++ is used to model component failure and repair data (i.e., TTF and TTR). A goodness-of-fit test is used to determine the corresponding distribution and estimate its parameters.

BlockSim is then used to model the cloud infrastructure availability at the system-level using an RBD. The cloud infrastructure system is modeled as a simple series diagram, which referred to as the base model (see Fig. 8). Failure and repair distributions are fed into BlockSim, and Monte Carlo simulations are done for 300,000-time units. Moreover, three more models representing different scenarios are created for comparison with the base model, and the model was showing the greatest availability is selected.

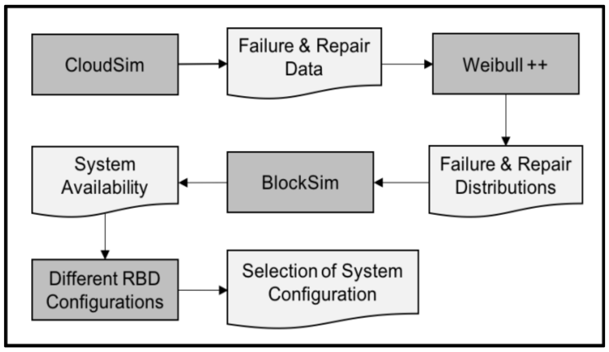
CloudSim
Failure & Repair Data
Weibull ++
Failure & Repair Distributions
BlockSim
System Availability
Different RBD Configurations
Selection of System Configuration

Fig. 7. Testing and validation approach.

B. Results and Discussion

The simulations show that failure data at the component-level (host, VM, and cloudlet) were successfully fitted to a Weibull distribution. The parameters of each distribution were estimated using regression analysis. For instance, Fig. 9 shows the Weibull probability plot for host failure data. In the probability plot, the shape parameter (beta) is estimated based on the fitted-line slope. The scale parameter (eta) is the time at which a specific percentage of the population has failed. The correlation coefficient (rho) is a measure of how well the linear regression model fits the data. Furthermore, VM repair data were the best fit with the lognormal distribution and a two-sided confidence level of 90% (see Fig. 10). In contrast, both the host and cloudlet have constant values for repair that are 10,800 hours and 300 hours, respectively.

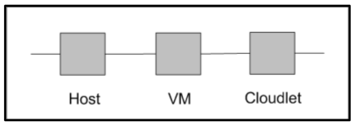

Fig. 8. Base model RBD.

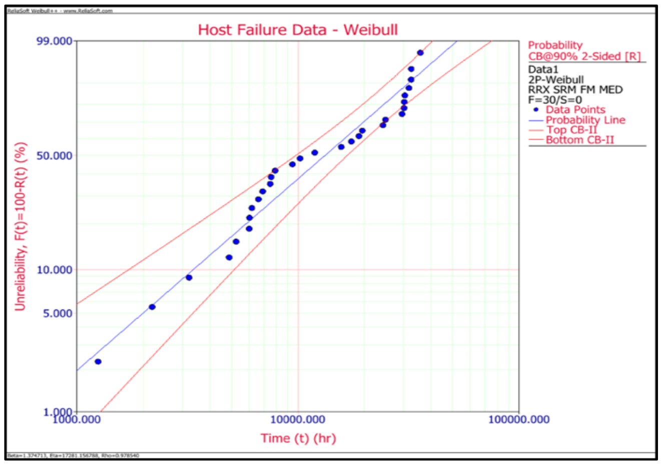

Fig. 9. Host failure data Weibull probability plot.

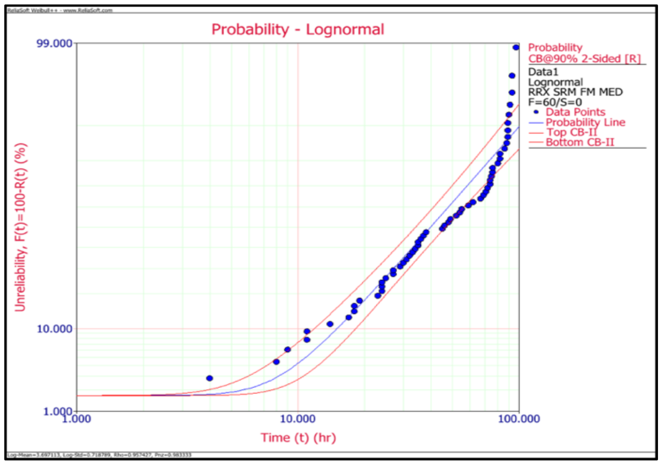

Fig. 10. VM repair data lognormal probability plot.

The failure and repair data distributions with associated parameters were used to feed several simulation models that were built using BlockSim and configured in four different models.

A base model was built in simple series and configured using the data provided by Weibull++. Table 2 shows the base model block configurations detailing the inputs for each block (i.e., host, VM, and cloudlet) on the availability model and the corrective task.

At the simulation end time (300,000-time units), the base model achieved an availability of 0.590767. Hence, to improve the system availability (i.e., to fulfill the customer's requirements), a sensitivity analysis is performed to study the impact of the repair rate and standby configuration (i.e., redundancy technique) on the overall system availability.

In the second model, host repair time was improved in the base model to determine its impact on overall system availability. The results showed that at the simulation end time (300,000-time units), availability is increased to 0.722.

In the third model, the base model was improved by using standby configurations. The base model was rebuilt with a standby container that included three base model systems; one system was active, and two were on standby (see Fig. 11). It is assumed that the switching reliability is 100%. The standby simulation results showed an improvement in overall system availability.

For instance, at the simulation end time (300,000-time units), system availability was 0.977. Also, a fourth model was created by applying a standby configuration to the second model in which the host repair time is improved. The results showed an improvement in overall system availability. At the simulation end time (300,000-time units), availability had increased to 0.996313. Fig. 12 shows the mean availability of the four models.

TABLE. II. BASE MODEL CONFIGURATIONS

Configuration	**Attribute**	**Host**	**VM**	**Cloudlet**
Reliability Model	Distribution	2P-Weibull	2P-Weibull	2P-Weibull
	Beta	1.37	1.43	1.46
	Eta	17281	17136	16260
Corrective Task	Distribution	Constant	Lognormal	Constant
	Parameter 1	10800	3.7 (log mean)	300
	Parameter 2	NA	0.72 (log-std)	NA
	Restoration	As good as new	As good as new	As good as new
	Task result	Bring item down	Bring item down	Bring item down

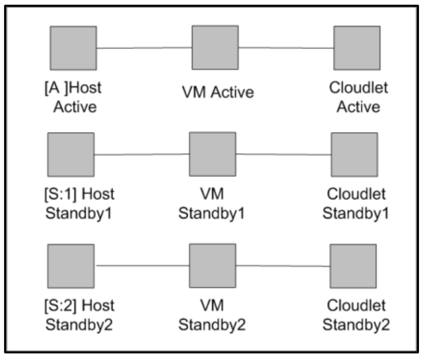

Fig. 11. Base model with standbys RBD.

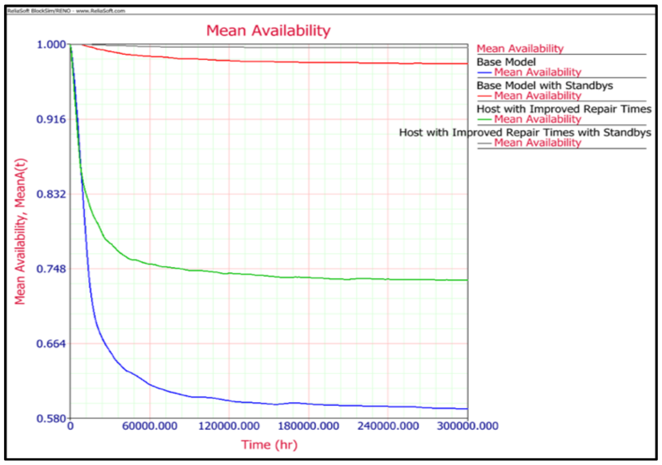

Fig. 12. Mean availability overlay plot for all models.

V. Conclusions and Future Work

This paper's primary contribution is a proposed model for evaluating the availability of cloud service provisioning systems. The model relies on NIST-CCRA, the well-known cloud computing reference architecture, to define cloud subsystems/components and the logical relationships among them. Using mature modeling techniques from reliability theory that can provide the operational measures that are so desirable today, we were able to quantify component-level availability. Furthermore, by considering series/parallel arrangements of the cloud system components, RBD was used to model system-level availability. The proposed model has some limitations imposed by the characteristics of RBDs. In future research, a dynamic RBD [30] will be adopted to consider the dynamic behavior of a cloud system. Other cloud scenarios such as cloud federation will also be modeled in future studies.

References

[1] F. Longo, R. Ghosh, V. K. Naik, and K. S. Trivedi, "Availability Analysis of IaaS Cloud Using Analytic Models," Achieve. Fed. Self-Manageable Cloud Infrastructures Theory Pract. IGI Glob., pp. 134–136, 2012.

[2] A. Undheim, A. Chilwan, and P. Heegaard, "Differentiated Availability in Cloud Computing SLAs," 2011 IEEE/ACM 12th Int. Conf. Grid Comput., pp. 129–136, Sep. 2011.

[3] A. Croll, Cloud performance from the end user perspective. 2011.

[4] G. P. Gibilisco, "Model-Based Availability Evaluation of Multi-Cloud Applications," University of Illinois at Chicago, 2013.

[5] F. Longo, R. Ghosh, V. K. Naik, K. S. Trivedi, and I. B. M. T. J. Watson, "A Scalable Availability Model for Infrastructure-as-a-Service Cloud," in Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on. IEEE, 2011.

[6] R. Ghosh, F. Longo, F. Frattini, S. Russo, and K. S. Trivedi, "Scalable Analytics for IaaS Cloud Availability," IEEE Trans. CLOUD Comput. VOL.2, vol. 2, no. X, pp. 1–14, 2014.

[7] V. Lo Faso, "Understanding NIST ' s Cloud Computing Reference Architecture : Part II Understanding NIST ' s Cloud Computing Reference Architecture : Part II," Global Knowledge, 2014. .

[8] R. N. Calheiros, R. Ranjan, A. Beloglazov, and A. F. De Rose, "CloudSim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software. Pract. Exp. 41.1, no. August 2010, pp. 23–50, 2011.

[9] A. Zhou, S. Wang, Q. Sun, H. Zou, and F. Yang, "FTCloudSim: A Simulation Tool for Cloud Service Reliability Enhancement Mechanisms," Proc. Demo Poster Track ACM/IFIP/USENIX Int. Middleware. Conf., p. 2:1--2:2, 2013.

[10] "BlockSim: System Reliability and Maintainability Analysis Software Tool," ReliaSoft Corporation. .

[11] ReliaSoft Corporation, "Weibull++: Life Data Analysis (Weibull Analysis) Software Tool," 2017. [Online]. Available: http://weibull.reliasoft.com.

[12] J. Teixeira, C. E. Salgado, and R. J. Machado, "Modeling an IaaS Broker Based on Two Cloud Computing Reference Models," 2016 IEEE Int. Conf. Cloud Eng. Work., pp. 166–171, 2016.

[13] M. Ribas, A. S. Lima, N. Souza, T. Engenharia, and S. Paulo, "Assessing Cloud Computing SaaS adoption for Enterprise Applications using a Petri net MCDM framework," Netw. Oper. Manag. Symp. (NOMS), IEEE., 2014.

[14] G. Callou, P. Maciel, D. Tutsch, and J. Araújo, "A Petri Net-Based Approach to the Quantification of Data Center Dependability," Petri Nets - Manuf. Comput. Sci., pp. 313–336, 2012.

[15] R. Jhawar, V. Piuri, and I. Universit, "Fault Tolerance Management in IaaS Clouds," 2012 IEEE First AESS Eur. Conf. Satell. Telecommun., pp. 1–6, 2012.

[16] D. S. Kim, F. Machida, and K. S. Trivedi, "Availability Modeling and Analysis of a Virtualized System," 2009 15th IEEE Pacific Rim Int. Symp. Dependable Comput., pp. 365–371, Nov. 2009.

[17] B. Silva, P. Maciel, E. Tavares, and A. Zimmermann, "Dependability Models for Designing Disaster Tolerant Cloud Computing Systems," in Dependable Systems and Networks (DSN), 43rd Annual IEEE/IFIP International Conference on. IEEE, 2013.

[18] R. Bauer, E., Adams, Reliability and Availability of Cloud Computing. 2012.

[19] J. Dantas, R. Matos, J. Araujo, and P. Maciel, "Models for Dependability Analysis of Cloud Computing Architectures for Eucalyptus Platform," Int. Trans. Syst. Sci. Appl., vol. 8, no. December, pp. 13–25, 2012.

[20] D. S. Kim, F. Machida, and K. S. Trivedi, "Availability modeling and analysis of a virtualized system," 2009 15th IEEE Pacific Rim Int. Symp. Dependable Comput. PRDC 2009, pp. 365–371, 2009.

[21] A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm, "What's inside the cloud? An architectural map of the cloud landscape," Proc. 2009 ICSE Work. Softw. Eng. Challenges Cloud Comput. CLOUD 2009, pp. 23–31, 2009.

[22] S. Kristopher, "Living in the Cloud Stack – Understanding SaaS, PaaS, and IaaS APIs," Nordic APIs, 2016. .

[23] K. Weyns and M. Höst, "Case Study on Risk Analysis for Critical Systems with Reliability Block Diagrams," in 10th International IT Systems for Crisis Response and Management (ISCRAM) Conference. ISCRAM, 2013, no. May, pp. 693–702.

[24] Gabriele Manno, "Reliability modelling of complex systems: an adaptive transition system approach to match accuracy and efficiency," University of Catania, 2012.

[25] B. Wei, C. Lin, and X. Kong, "Dependability Modeling and Analysis for the Virtual Data Center of Cloud Computing," 2011 IEEE Int. Conf. High Perform. Comput. Commun., pp. 784–789, Sep. 2011.

[26] T. Thanakornworakij, R. F. Nassar, and C. Leangsuksun, "A Reliability Model for Cloud Computing for High-Performance Computing Applications," in European Conference on Parallel Processing. Springer Berlin Heidelberg, 2013, pp. 474–483.

[27] G. Callou, P. Maciel, D. Tutsch, and J. Araújo, "A Petri Net-Based Approach to the Quantification of Data Center Dependability," Petri Nets-Manufacturing Comput. Sci. InTech, 2012.

[28] T. Humble, Availability, Reliability, Maintainability, and Capability. Triplex Chapter of the Vibrations Institute. Humble, TX: Barringer and Associated Inc, 1997.

[29] A. Alkasem and H. Liu, "Research Article A Survey of Fault-tolerance in Cloud Computing : Concepts and Practice," Sch. Comput. Sci. Technol. , Harbin Inst. Technol. China, vol. 11, no. 12, pp. 1365–1377, 2015.

[30] ReliaSoft, "Reliability Basics: Availability and the Different Ways to Calculate It." .

[31] V. K. Katukoori, "Standardizing Availability Definition," Univ. New Orleans, New orleans, La., USA, p. 21, 2007.