RC25247 (W1112-048) December 15, 2011
Computer Science

IBM Research Report

Availability Analysis of IaaS Cloud Using Analytic Models

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

[1]Dipartimento di Matematica
Università degli Studi di Messina
Italy

[2]Department of Electrical and Computer Engineering
Duke University
USA

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

Availability Analysis of IaaS Cloud Using Analytic Models

Francesco Longo, Dipartimento di Matematica, Università degli Studi di Messina, Italia

Rahul Ghosh, Department of Electrical and Computer Engineering, Duke University, USA

Vijay K. Naik, IBM T. J. Watson Research Center, USA

Kishor S. Trivedi, Department of Electrical and Computer Engineering, Duke University, USA

ABSTRACT

Cloud based systems are inherently large scale. Failures in such a large distributed environment are quite common phenomena. To reduce the overall Cloud downtime and to provide a seamless service, providers need to assess the availability characteristics of their data centers. Such assessments can be done through controlled experimentations, large scale simulations and via analytic models. In the scale of Cloud, conducting repetitive experimentations or simulations might be costly and time consuming. Analytic models, on the other hand, can be used as a complement to small scale measurements and simulations since the analytic results can be obtained quickly. However, accurate analytic modeling requires dealing with large number of system states, leading to state-space explosion problem. To reduce the complexity of analysis, novel analytic methods are required. In this chapter, we introduce the reader with a novel approach using interacting analytic sub-models and we show how such approach can deal with large scale Cloud availability analysis. We put our work in perspective of other existing and ongoing research in this area, describe how such approach can be useful to Cloud providers, especially in the case of federated scenarios, and summarize the open research questions that are yet to be solved.

INTRODUCTION

Cloud computing is a model of Internet-based computing. An IaaS Cloud, such as Amazon EC2 and IBM SmatCloud Enterprise (Amazon EC2: http://aws.amazon.com/ec2, 2011; IBM SmatCloud Enterprise: www.ibm.com/services/us/en/cloud-enterprise/, 2011) delivers, on-demand, operating system (OS) instances provisioning computational resources in the form of VMs deployed in the Cloud provider's data center. Requests submitted by the users are provisioned and served if the Cloud has enough available capacity in terms of physical machines (PMs). Large Cloud service providers such as IBM provide service level agreements (SLAs) regulating the availability of the Cloud service. Before committing an SLA to the customers of a Cloud, the service provider needs to carry out availability analysis of the infrastructure on which the Cloud service is hosted. In this chapter, we show how stochastic analytic models can be utilized for Cloud service availability analysis. We first provide a background on the subject describing how the problem is faced in the current literature. Then, we propose an example of one-level monolithic model that can be used to analyze the availability of an IaaS Cloud. However, such monolithic models become intractable as the size of Cloud increases. To overcome this difficulty, we illustrate the use of an interacting sub-models approach. Overall model solution is obtained by iteration over individual sub-model solutions. Comparison of the results with monolithic model shows that errors introduced by model decomposition are negligible. We also show how closed form solutions of the sub-models can be obtained and demonstrate that the approach can scale for large size Clouds. The presence of three pools of PMs and the migration of them from one pool to another caused by failure events makes the model both novel, interesting and particularly suitable in federated environments. In order to automate the construction and solution of underlying Markov models, we use a variant of stochastic Petri net (SPN) called stochastic reward net (SRN). This paradigm is supported by SHARPE (Trivedi & Sahner, 2009) and Stochastic Petri Net Package (SPNP) (Hirel, Tuffin, &

Trivedi, 2000) software packages.
Rest of the chapter is organized as follows. Section II gives a background on the subject illustrating the state of the art. Section III provides an introduction to the formalism that will be used in the following to model the considered scenario. Section IV describes Cloud system model, assumptions and problem formulation. Section V, presents the monolithic SRN model. Interacting SRN sub-models are described in Section VI and their closed form solutions are presented in Section VII. Fixed point iteration among the interacting sub-models and proof of existence of a solution is shown in Section VIII. Results obtained from monolithic approach and interacting sub-models approach are compared in Section IX. Sections X and XI discuss how the approach can be used by Cloud providers, point out future challenges and highlight the benefit of decomposed models in the analysis of federation scenarios. We conclude the chapter in Section XII.

BACKGROUND

In this section, we highlight key analytic approaches for system availability assessment. There are four main types of analytic modeling techniques (Nicol, Sanders, & Trivedi, 2004; Trivedi, 2001; Trivedi, Kim, Roy, & Medhi, 2009) that can be applied for availability analysis: non-state-space models, state-space space models, hierarchical, and fixed-point iterative models (Haring, Marie, Puigjaner, & Trivedi, 2001; Longo, Ghosh, Naik, & Trivedi, 2011; Mainkar & Trivedi, 1996; Tomek & Trivedi, 1991). Reliability block diagram (RBD), reliability graph (Relgraph), fault tree (FT) are examples of non-state-space models. Such models can be easily developed assuming statistical independence between system components and thus allowing a fast solution for system reliability, system MTTF and system availability. However, non-state space models cannot easily handle failure/repair dependencies (e.g., shared repair, warm/cold spares, imperfect coverage, etc.). In order to model complex interactions between system components, Markov chains or more generally state-space models can be used (Trivedi, 2001). Markov chains consist of state(s) and state transition(s). In Discrete Time Markov Chains (DTMC), all transition labels are probabilities. In Continuous Time Markov Chains (CTMC), all transition labels are rates. If the rates are time dependent then we have a non-homogeneous CTMC. If each label is a distribution function (Yin, Fricks, & Trivedi, 2002) then we have a Semi-Markov Process (SMP) or a Markov Regenerative Process (MRGP) (Kulkarni, 2010). Markovian and non-Markovian models may suffer from the state-space explosion problem (or largeness problem). SPNs (Trivedi, 2001) or SRNs can be used for easy specification and automated generation/solution of underlying Markov models to tolerate largeness. To avoid largeness problems, hierarchical models (Chen, Dharmaraja, Chen, Li, Trivedi, Some, & Nikora, 2002; Kim, Machida, & Trivedi, 2009; Smith, Trivedi, Tomek, & Ackaret, 2008; Tomek, Muppala, & Trivedi, 1993; Trivedi, Vasireddy, Trindade, Nathan, & Castro, 2006; Trivedi, Wang, Hunt, Rindos, Smith, & Vashaw, 2008) or interacting sub-models can be used. In interacting sub-models approach (Longo, Ghosh, Naik, & Trivedi, 2011), dependencies among the sub-models are usually resolved by fixed-point iterations (Haring, Marie, Puigjaner, & Trivedi, 2001; Mainkar & Trivedi, 1996; Tomek & Trivedi, 1991). Such interacting sub-models can be used for analysis of large scale Cloud infrastructure.
In (Vishwanath, & Nagappan, 2010), Vishwanath et al. investigated failure characteristics of servers in large Cloud data centers. They explored the relationship between the failure of a PM and other factors (e.g., age of the PM, number of disks on a PM etc.), tried to quantify the relationships between successive failures on same PM by analyzing experimental data and finally empirically compute reliability. The analytic approach described in this chapter can be complementary to this work since it takes into consideration multiple classes of PMs and consider their failure and repair. In (Yang, Tan, Dai, & Guo, 2009), Yang et al. investigated the failure of workloads on Cloud service performance. They considered response time as the performance metric. Although, such joint analysis of availability and performance is important, the authors do not address the scalability issues. In this chapter, we restrict ourselves only to the scalability of availability models. In (Bonvin, Papaioannou, & Aberer, 2009), Bonvin et al. designed a reliable and cost-effective storage system that maintains high availability guarantees despite failures of servers. The authors address interesting cost-optimization questions. Similar optimization problems can be formulated using the models and approach we describe in this chapter. In (Joshi, Bunker, Jahanian, Moorsel, & Weinman, 2009), Joshi et al. discussed the key challenges and opportunities in achieving high availability in large scale Cloud services. They outline different security threats, privacy issues and compliance requirements for a highly available Cloud. Although, the authors do not solve any specific problem, their work can provide interesting open research issues in this area.
There are limited research efforts which investigated availability in large scale infrastructure. In (Tan, Gu, & Wang, 2010), Tan et al. designed and implemented a prediction system to achieve robust hosting for production hosting infrastructure. Specifically, they designed a system called ALERT to predict anomalies in

the system. In the context of this chapter, anomalies in failure-repair behavior can be viewed as a sub-problem as addressed by the Tan et al. Moreover, the modeling approach described here can be complementary to such experimental work. In (Javadi, Kondo, Vincent, & Anderson, 2010), Javadi et al. show how statistical models can be useful to predict availability of an Internet distributed system. Analytic models described in this chapter can perhaps be combined with the statistical models proposed by Javadi et al. In (Uemura, Dohi, & Kaio, 2009), Uemura et al. used discrete time semi-Markov process to describe the stochastic behavior of a scalable intrusion tolerant system. In contrast, for Cloud availability analysis, we start with Petri net (PN) based models, to facilitate automated generation of Markov chains and subsequently decompose the large PN model into small PNs and eventually to Markov chains. In (Chen, Zhou, & Xiong, 2010) Chen et al. used a deterministic and stochastic PN method to illustrate the performance of producer/consumer based application models in Cloud context. Unlike the approach proposed here, they focus only on performance behavior of Cloud. In our previous work (Ghosh, Longo, Naik, & Trivedi, 2010), we showed an SRN modeling approach for resiliency analysis of IaaS Cloud but scalability of such approach needs to be investigated. The key challenges in developing an analytic model for Cloud are: (i) developed models should be accurate by taking into account different system details, and (ii) the models should be scalable and tractable i.e., their solution times should be negligible, especially in the case of federated environments. In the subsequent sections, we describe how such an analytic model can be developed.

STOCHASTIC PETRI NETS AND REWARD NETS

This section presents an overview of SPNs and SRNs. A PN can be formally defined as a 4-tuple: *PN = (P, T, A, M)*, where *P* is the finite set of places (represented by circles), *T* is the finite set of transitions (represented by bars), *A* is the set of arcs (connecting elements of *P* and *T*) and *M* is the set of markings each of which denotes the number of tokens in the places of the net. In SPN, exponentially distributed firing times can be associated to the net transitions so that the stochastic process underlying a SPN is a CTMC. In generalized SPNs (GSPN) (Marsan, Balbo, & Conte, 1984), transitions are allowed to be either timed (exponentially distributed firing time, drawn as rectangular boxes) or immediate (zero firing time, represented by thin black bars). Immediate transitions always have priority over timed transitions and if both timed and immediate transitions are enabled in a marking then timed transitions are treated as if they were not enabled. If several immediate transitions compete for firing, a specified probability mass function is used to break the tie. A marking of a GSPN is called vanishing if at least one immediate transition is enabled in it. A marking is called tangible otherwise. GSPN also introduces the concept of inhibitor arc (represented by a small hollow circle at the end of the arc) which connects a place to a transition. A transition with an inhibitor arc cannot fire if the input place of the inhibitor arc contains more tokens than the multiplicity of the arc.
SRNs (Ciardo, Blakemore, Chimento, Muppala, & Trivedi, 1993) are extensions of GSPNs. In SRNs, every tangible marking can be associated with a reward rate thus facilitating the computation of a variety of performance measures. Key features of SRNs are:

1. each transition may have an enabling function (also called a guard) so that a transition is enabled only if its marking-dependent enabling function is true;
2. marking dependent arc multiplicities are allowed;
3. marking dependent firing rates are allowed;
4. transitions can be assigned different priorities;
5. besides traditional output measures obtained from a GSPN, such as throughput of a transition and mean number of tokens in a place, more complex measures can be computed by using reward functions.

PROBLEM DEFINITION

System model and assumptions

In IaaS Cloud, when a request is processed, a pre-built image is used to create one or more VM instances. When the VM instances are deployed, they are provisioned with request specific CPU, RAM, and disk capacity. VMs are deployed on PMs each of which may be shared by multiple VMs. To reduce overall VM provisioning delays and operational costs, it is a common practice to group the PMs into a certain number of tiered pools. Here, we suppose that PMs are grouped into three pools; hot (running), warm (turned on, but not ready) and cold (turned off). Maintaining the PMs in three pools (in general, multiple tiered pools) helps to minimize power and cooling costs without incurring high startup delays for all VMs. A pre-instantiated VM can be readily provisioned and brought to ready state on a running PM (hot PM) with minimum provisioning delay. Instantiating a VM from an image and deploying it on a warm PM needs additional provisioning time. PMs in the cold pool are turned-off when not in use and deploying a VM on such a PM adds to the startup delays. A performability model of this system was presented in (Ghosh, Trivedi, Naik, & Kim, 2010) where it has been shown that the “bottleneck” model is the availability model. Hence, the objective of this chapter is to show a scalable availability model of Cloud service with the following assumptions:

(1) Variety of failures/repairs can occur in a Cloud environment such as failure/repair of hardware, software, hypervisor, VM, OS and applications. Here, we consider only the net effect of different failures and repairs of PMs in the hot, warm and cold pools. We do not consider software and OS failures in a VM. Typically, these failures are handled by restarting the VM. Although the cause of a PM failure can be because of variety of reasons, in our analysis we consider the net combined effect on the PM failure rate. However, it is possible to extend such availability model to capture detailed PM failure modes and recovery steps as in (Trivedi, 2001; Trivedi, Wang, Hunt, Rindos, Smith, & Vashaw, 2008).

(2) We assume that all times to failure are exponentially distributed. Equivalent mean time to failure (MTTF) of each hot PM is $1/\lambda_h$ and that of each warm PM is $1/\lambda_w$. Typically $1/\lambda_w$ is higher than $1/\lambda_h$ by a factor of 2 to 4. Cold machines can fail with a very low failure rate $1/\lambda_c$ with $\lambda_h \gg \lambda_c$ and $\lambda_w \gg \lambda_c$. It is possible to remove the assumption of exponential distribution as done in (Wang, Fricks, & Trivedi, 2003).

(3) All PMs in a pool are identical. Failure of a PM in one pool triggers migration of a PM (if available) from other pools to replace the failed one. When a hot PM fails, the failed PM needs to be repaired and at the same time the system tries to replace it by a warm PM, if available (i.e., in “UP” state). If no warm PM is available, replacement is attempted by migrating an available cold PM to the hot pool. When a warm PM fails, the failed PM undergoes repair and at the same time it is replaced by a PM from the cold pool (if there is at least one PM available in cold pool). We assume that the migration process is instantaneous.

(4) Each pool has its own repair facilities. Within a pool, maximum number of PMs that can be repaired in parallel is assumed to be n_r. Value of n_r is assumed to be greater or equal to 1 but less than the maximum number of PMs in the pool. When the number of PM failures are higher than n_r, failed PMs are put in a queue for repair. Across different pools, repairs can be done in parallel. We assume that time to repair is exponentially distributed with mean $1/\mu$. Once a failed PM is repaired, it is returned to the original pool where it belonged before failure. If a PM was borrowed from other pool to replace the failed PM; such borrowed PM is also returned to its original pool instantaneously.

Problem Formulation

Assume that $\boldsymbol{n_h}$, $\boldsymbol{n_w}$, and $\boldsymbol{n_c}$ PMs are initially available in the hot, warm, and cold pools, respectively. A possible definition of availability in such a context is that at least $\boldsymbol{k}$ PMs (with $\boldsymbol{1 \leq k \leq n_h + n_w + n_c}$) should be available across all the pools combined in order for the system to be up. Under the failure, repair and migration of the PMs across different pools, we show how to compute the average number of PMs in each pool at steady state and the effects of downtime on the Cloud service.

We start by developing a monolithic model using the high level formalism of SRN for the automated generation and solution of the underlying Markov chain. The monolithic model is not scalable to the large size Clouds that we wish to analyze and that can be present in a federated environment. Hence, we show an interacting SRN sub-models approach that is scalable. As an important side-benefit, the decomposition also enables us to obtain closed form solutions of sub-models. Three key comparisons are made between these two approaches: (i) errors introduced by interacting sub-models, (ii) maximum number of PMs that each approach can handle and (iii) solution time required for both the approaches. Through systematic analysis, we show that interacting SRN sub-models approach is highly scalable compared to single monolithic modeling approach. Closed form solutions of the sub-models are especially useful in providing a highly scalable and fast method for the availability analysis of large sized IaaS Cloud.

MONOLITHIC AVAILABILITY MODEL

Monolithic SRN model for the availability analysis of IaaS Cloud is shown in Figure 1.

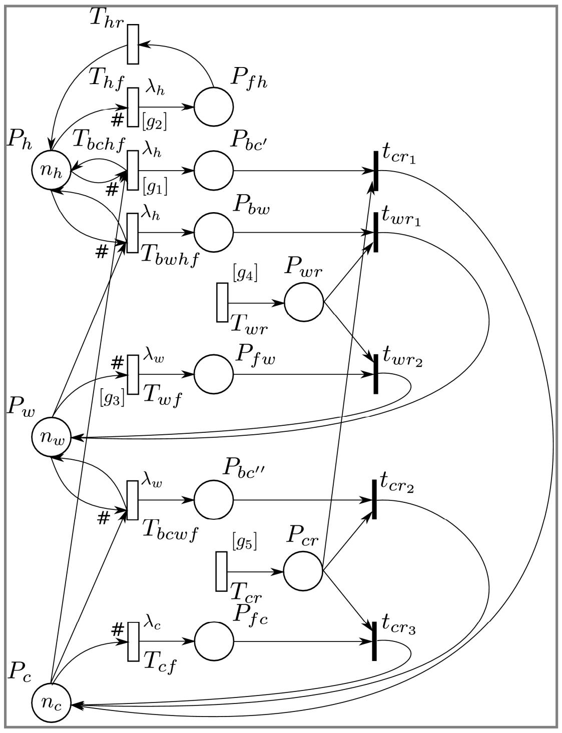


Figure 1. Monolithic SRN model for availability analysis of IaaS Cloud.

Input parameters of monolithic model are: (1) initial number of PMs in each pool (n_h, n_w, and n_c), (2) MTTFs of hot, warm, and cold PMs ($1/\lambda_h$, $1/\lambda_w$, and $1/\lambda_c$, respectively), (3) number of repair facilities for each pool (n_r), (4) MTTR of a PM ($1/\mu$). Among the input parameters, n_h, n_w, n_c and n_r are design parameters, MTTF and MTTR values are measured. Five guard functions are defined on the model and they are described in Table 1.

Guard functions	Values
g_1	1 if $\#P_w = 0$ 0 otherwise
g_2	1 if $\#P_w = 0$ and $\#P_c = 0$ 0 otherwise
g_3	1 if $\#P_c = 0$ 0 otherwise
g_4	1 if $\#P_{fw} + \#P_{bw} > 0$ 0 otherwise
g_5	1 if $\#P_{fc} + \#P_{bc'} + \#P_{bc''} > 0$ 0 otherwise

Table 1. Guard functions defined on monolithic SRN model and interacting SRN sub-models.

Places P_h, P_w, and P_c represent the hot, warm, and cold pool, respectively. Number of tokens in these places indicates the number of "UP" PMs in the corresponding pool. Transitions T_{bwhf}, T_{bchf}, and T_{hf} represent the failure event of a hot PM. Since migration of a PM is attempted upon failure of a hot PM, three cases are possible: (1) T_{bwhf} fires if a warm PM is available for migration to the hot pool, (2) T_{bchf} fires if the warm pool is empty but a cold PM is available to be borrowed, and (3) T_{hf} fires if both the warm and the cold pool are empty so that no PM is available to substitute the failed hot PM. The guard functions $[g_1]$ and $[g_2]$ model the three mutually exclusive cases. Moreover, rates of these transitions are considered to be dependent on the number of tokens in place P_h so that the overall hot PM failure rate is equal to λ_h multiplied by the number of available hot PMs. These marking dependent firing rates are represented by the # symbol near the input arcs which connect the transitions T_{bwhf}, T_{bchf}, and T_{hf} to the place P_h.

Upon firing of transition T_{bwhf}, a token is removed from place P_w and the number of tokens in place P_h remains unchanged. At the same time, a token is deposited in place P_{bw}. This place keeps track of number of failed PMs that need to be repaired and given back to the warm pool at the end of the repair process. Similarly, upon firing of transition T_{bchf}, a token is removed from place P_c and the number of tokens in place P_h remains unchanged. Simultaneously, a token is deposited in place P_{bc}, to take into account that a PM has to be repaired and given back to the cold pool. Upon firing of transition T_{hf}, following token exchanges happen: (i) removal of a token from place P_h to model the reduction in number of available PMs in the hot pool by one and (ii) deposition of a token in place P_{fh} to model that the failed PM has to be repaired and given back to the hot pool.

Failure-repair behavior of warm pool is modeled similarly. Transitions T_{bcwf}, and T_{wf} model the failure event of a warm PM. Two cases are possible: (1) T_{bcwf} fires if a cold PM is available for migration to warm pool, and (2) T_{wf} fires if the cold pool is empty and no PM is available to substitute the failed warm PM. The guard function $[g_3]$ models the fact that the two cases are mutually exclusive. Rates of transitions T_{bcwf} and T_{wf} are considered to be dependent on the number of tokens in place P_w so that the overall warm PM failure rate is equal to λ_w multiplied by the number of available warm PMs. Firing of T_{bcwf} removes a token from place P_c and deposits a token to place P_{bc}, representing the failed PM that needs to be repaired and given back to the cold pool. Upon firing of T_{wf}, a token is removed from place P_w and a token is deposited to place P_{fw} representing the failed PM that needs to be repaired and given back to the warm pool.

Transition T_{cf} fires when a cold PM fails. Rate of such transition is considered to be dependent on the number of tokens in place P_c so that the overall cold PM failure rate is equal to λ_c multiplied by the number of available cold PMs. Upon firing of T_{cf}, a token is removed from place P_c and deposited to place P_{fc}.

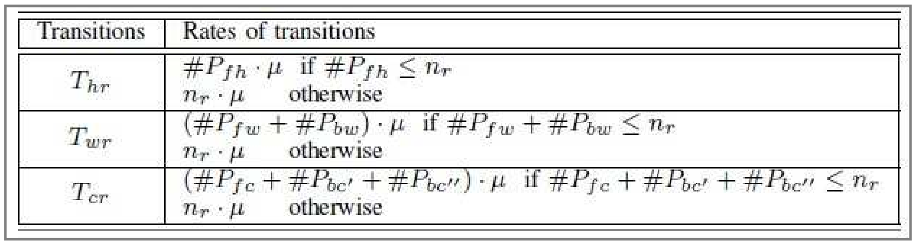
Transitions T_{hr}, T_{wr}, and T_{cr} model the repair of the failed PMs. Rates of these transitions are marking dependent to take into account the presence of n_r repair facilities for each pool. In particular, the rates of the above mentioned transitions are reported in Table 2. Guard functions $[g_4]$ and $[g_5]$ allow transitions T_{wr}, and T_{cr} to be enabled only when at least one PM needs to be repaired. Immediate transitions t_{wr_1}, t_{wr_2}, t_{cr_1}, t_{cr_2}, and t_{cr_3} model the instantaneous migrations of repaired PMs to the original pool.

Model outputs

Outputs of the model are obtained using the Markov reward approach by assigning an appropriate reward rate to each marking of the SRN and then computing the expected reward rate both in transient and steady state as the desired measures (Trivedi, 2001). Let r_i be the reward rate assigned to marking *i* of the SRN in Figure 1. If $\pi_i(t)$ denotes the probability for the SRN to be in marking *i* at time *t* then the expected reward rate at time *t* is given by $\sum_i \pi_i(t)\, r_i$. The expected steady state reward rate can be computed by taking into consideration the steady state probabilities π_i of the SRN as $\sum_i \pi_i\, r_i$. Our measures of interest are following.

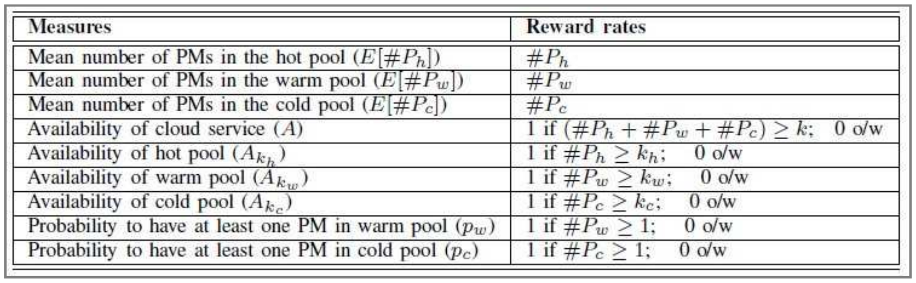
(i) Mean number of PMs in each pool - The mean number of PMs in the hot pool is given by the mean number of tokens in the corresponding place P_h ($E[P_h]$). Similarly, for the warm and the cold pool we consider the mean number of tokens in places P_w and P_c, respectively ($E[P_w]$ and $E[P_c]$). Reward assignment for this measures is shown in Table 3.

(ii) Availability of Cloud service (A) - As mentioned above, we consider the Cloud service to be available if the total number of PMs across all hot, warm, and cold pool is greater than or equal to $\boldsymbol{k}$ (with $\boldsymbol{1 \leq k \leq n_h + n_w + n_c}$). As a consequence, the reward assignment for this measure is the one shown in Table 3.



Transitions	Rates of transitions
T_{hr}	$\#P_{fh} \cdot \mu$ if $\#P_{fh} \leq n_r$ $n_r \cdot \mu$ otherwise
T_{wr}	$(\#P_{fw} + \#P_{bw}) \cdot \mu$ if $\#P_{fw} + \#P_{bw} \leq n_r$ $n_r \cdot \mu$ otherwise
T_{cr}	$(\#P_{fc} + \#P_{bc'} + \#P_{bc''}) \cdot \mu$ if $\#P_{fc} + \#P_{bc'} + \#P_{bc''} \leq n_r$ $n_r \cdot \mu$ otherwise

Table 2. Rates of transitions modeling the repair of failed PMs in monolithic SRN model and interacting SRN sub-models



Measures	Reward rates
Mean number of PMs in the hot pool ($E[\#P_h]$)	$\#P_h$
Mean number of PMs in the warm pool ($E[\#P_w]$)	$\#P_w$
Mean number of PMs in the cold pool ($E[\#P_c]$)	$\#P_c$
Availability of cloud service (A)	1 if $(\#P_h + \#P_w + \#P_c) \geq k$; 0 o/w
Availability of hot pool (A_{k_h})	1 if $\#P_h \geq k_h$; 0 o/w
Availability of warm pool (A_{k_w})	1 if $\#P_w \geq k_w$; 0 o/w
Availability of cold pool (A_{k_c})	1 if $\#P_c \geq k_c$; 0 o/w
Probability to have at least one PM in warm pool (p_w)	1 if $\#P_w \geq 1$; 0 o/w
Probability to have at least one PM in cold pool (p_c)	1 if $\#P_c \geq 1$; 0 o/w

Table 3. Reward rates to compute different output measures from monolithic SRN model and interacting SRN sub-models

INTERACTING SRN SUB-MODELS

We decompose the monolithic model into three sub-models, each of which captures the failure and repair behavior of a single pool. Here, we describe how these three sub-models interact with each other and can be used to compute the same quantities that are computed from the monolithic model. The SRN sub-models for the hot, warm and cold pool are shown in Figures 2, 3 and 4, respectively. Observe that some of the transitions of the monolithic model are present in more than one sub-model. Hence, to obtain overall model solution, sub-models exchange some of the input parameters and output measures. Guard functions $[g_1]$, $[g_2]$, and $[g_3]$ $a$re not present in the interacting sub-models approach while the rates of transitions T_{hr}, T_{wr}, and T_{cr} are still marking dependent according to the functions described in Table 2.

The structure of the hot pool sub-model in Figure 2 is obtained from the structure of the monolithic model by keeping the transitions that directly interact with place P_h and disregarding the others and the related places. Input parameters to this sub-model are: (i) initial number of PMs in hot pool (n_h), (ii) hot PMs failure rate (λ_h), (iii) hot PMs repair rate (μ), (iv) number of repair facilities in the hot pool (n_r). Among these input parameters, n_h and n_r are design parameters, λ_h and μ are measured. Assume p_w and p_c are the probabilities to have at least one PM available in warm and cold pool, respectively as computed from the warm and cold pool sub-models discussed later. In the hot pool sub-model, the rate of transition T_{bwhf} is $\lambda_h p_w$. This is because, in the monolithic model, an arc is present from place P_w to such a transition. In the hot pool sub-model, place P_w is not present but still the impact of the behavior of the warm pool sub-model on the throughput of transition T_{bwhf} needs to be taken into account by scaling its rate with the quantity p_w. Similarly, the rate of transition T_{bchf} is $\lambda_h(1-p_w)p_c$ because it is necessary to take into account the presence of the arc from place P_c (by multiplying with p_c) and the guard function $[g_1]$ (by multiplying with $1-p_w$) as used in the monolithic model. Finally, the rate of transition T_{hf} is $\lambda_h(1-p_w)(1-p_c)$ because we need to take into account the presence of the guard function $[g_2]$. From the hot pool sub-model we compute $E[\#P_h]$ that represents the mean number of tokens in place P_h, i.e., mean number of available PMs in the hot pool. It will be used in the warm and cold pool SRN sub-models to approximate the rate of transitions T_{bwhf} and T_{bchf}. Moreover, from the hot pool sub-model we compute the probability (A_{k_h}) for the hot pool to be available, i.e., the probability for the number of tokens in place P_h to be greater or equal to k_h with $0 < k_h \leq n_h$. It will be used to compute the overall Cloud service availability from the interacting sub-models. These output measures can be computed by assigning the reward rates reported in Table 3.

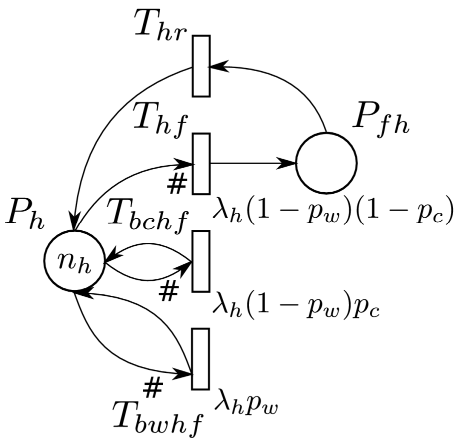


Figure 2. SRN sub-model for the availability analysis of the hot pool

Similar to the hot pool sub-model, the structure of the warm pool sub-model in Figure 3 is obtained from the structure of the monolithic model by keeping the transitions that directly interact with place P_w and disregarding the others and the related places. Input parameters to this sub-model are: (i) initial number of PMs in warm pool (n_w), (ii) warm PMs failure rate (λ_w), (iii) warm PMs repair rate (μ) and (iv) number of repair facilities for the warm pool (n_r). Among these input parameters, n_w and n_r are design parameters, λ_w and μ are measured. Computation of rate of transitions T_{bcwf} and T_{wf} is similar to the computation of rate of transitions T_{bwhf}, T_{bchf}, and T_{hf} as described for hot pool sub-model. Probability p_c is obtained from cold pool sub-model. However, the rate of transition T_{bwhf} needs to be set so that the throughput of this transition and the throughput of the transition with the same name in the hot pool sub-model are equal. In fact, the two transitions are same in the monolithic model. By equaling the throughput of the such transitions, we obtain:

$$rate_w(T_{bwhf}) = \lambda_h E[\#P_h] \quad (1)$$

where $E[\#P_h]$ is obtained from hot pool sub-model.

Outputs of warm pool sub-model are: (i) probability (p_w) to have at least one token in place P_w, i.e., at least one PM is available in the warm pool, (ii) mean number of tokens ($E[\#P_w]$) in place P_w, i.e., mean number of available PMs in the warm pool, and probability (A_{k_w}) for the number of tokens in place P_w to be greater or equal to k_w (with $0 \leq k_w \leq n_w$), i.e., availability of the warm pool. Among these output measures p_w will be used as an input parameter to the hot pool SRN sub-model to approximate the rates of transitions T_{bwhf}, T_{bchf}, and T_{hf}, $E[\#P_w]$ will be used as an input parameter to the cold pool SRN sub-model to approximate the rate of transition T_{bcwf}, and A_{k_w} will be used to compute the overall Cloud service availability from the interacting sub-models. The reward rates assignment for such output measures are shown in Table 3.

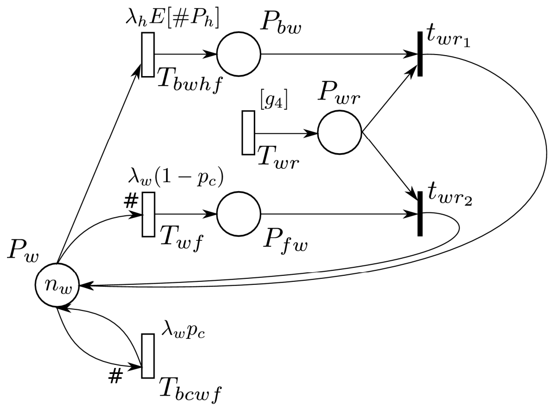


Figure 3. SRN sub-model for the availability analysis of the warm pool

Also in the case of cold pool, the structure of the sub-model in Figure 4 can be obtained from the structure of the monolithic model by keeping the transitions that directly interact with place P_c and disregarding the others and the related places. Input parameters to the cold pool sub-model are: (i) initial number of PMs in cold pool (n_c), (ii) cold PMs failure rate (λ_c), (iii) cold PMs repair rate (μ) and (iv) number of repair facilities for the cold pool (n_r). Among these input parameters, n_c and n_r are design parameters, λ_c and μ are measured. Following similar arguments as in the case of warm pool, we can compute the rate of transitions T_{bchf} and T_{bcwf} in the cold pool sub-model:

$$rate_c(T_{bchf}) = \lambda_h(1 - p_w)E[\#P_h] \quad (2)$$

and

$$rate_c(T_{bcwf}) = \lambda_w E[\#P_w] \quad (3)$$

where p_w and $E[\#P_w]$ are obtained from warm pool sub-model, and $E[\#P_h]$ is obtained from hot pool sub-model. Output measures of cold pool sub-model are: (i) probability (p_c) to have at least one token in place P_c, i.e., at least one PM is available in the cold pool, (ii) mean number of tokens ($E[\#P_c]$) in place P_c, i.e., mean number of available PMs in the cold pool, and (iii) the probability (A_{k_c}) for the number of tokens in place P_c to be greater or equal to k_c (with $0 \leq k_c \leq n_c$), i.e., the cold pool availability. p_c will be used as an input parameter to the hot and warm SRN sub-models to approximate the rate of transitions T_{bchf}, T_{hf}, T_{bcwf} and T_{wf}. A_{k_c} will be used to compute the overall Cloud service availability. Reward assignments for such output measures are reported in Table 3.

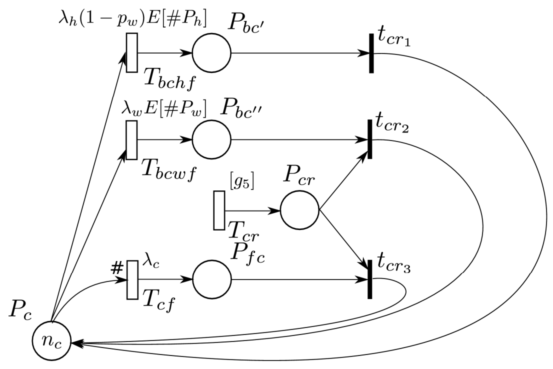


Figure 4. SRN sub-model for the availability analysis of the cold pool

All these sub-models and the interactions among them are shown as an import graph in Figure 5. We briefly describe the interactions among these models here. The hot pool sub-model computes the mean number of PMs in the hot pool ($E[\#P_c]$) that is needed as an input parameter to both the warm and cold pool sub-models. The warm pool sub-model compute the probability for the warm pool to have at least one available PM (p_w) and the mean number of PMs in the warm pool ($E[\#P_w]$). The former quantity is used both in the hot and cold pool sub-models while the latter is used in the cold pool sub-model. Finally, the output measure of old pool sub-model (p_c, i.e., the probability for the cold pool to have at least one available PM) is used both in the hot and warm pool sub-models. Observe, the import graph shows cyclic dependencies among the sub-models. Such dependencies are resolved using fixed point iteration (Mainkar & Trivedi, 1996; Tomek, & Trivedi, 1991).

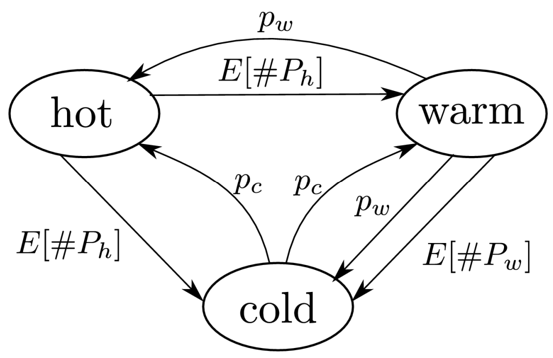


Figure 5. Interactions among the sub-models as an import graph.

Model outputs

Once the interacting sub-models have been solved, the same output measures of interest for the availability analysis of the Cloud service that can be computed from the monolithic model can be obtained. In particular, the mean number of PMs in each pool ($E[\#P_h]$, $E[\#P_w]$, and $E[\#P_c]$) are immediately available from the hot, warm, and cold pool sub-models, respectively. The availability of Cloud service for a given k can be computed by combining the availability of hot, warm, and cold pool as computed from hot, warm, and cold pool sub-models such that $k_h + k_w + k_c \geq k$.

CLOSED FORM SOLUTION OF THE SUB-MODELS

In this section, we show the closed form solution for the interacting SRN sub-models approach by using equivalent Markov chain models. The Markov chains are reported in the case of $n_r = 1$ to simplify calculations, but the closed form results can also be derived for the cases $n_r > 1$. The SRN sub-model for the hot pool shown in Figure 2 is equivalent to the Markov chain model shown in Figure 6. In this Markov chain, state *i* represents the configuration of the hot pool in which *i* PMs are available. While solving this Markov chain for steady state probability of each state, we can ignore the self-loops (Trivedi, 2001). Hence, the Markov chain depicted in Figure 6 is a simple birth-death process where birth rate for state i is $\lambda_h(1-p_w)(1-p_c)i$ and death rate for all states is μ. Let

$$\lambda'_w = \lambda_h\ (1-p_w)(1-p_c)\ \ (4)$$

Let p_{h_i} be the steady state probability to be in state *i* for the Markov chain of Figure 6, i.e., the probability to have *i* PMs in the hot pool. Under such assumptions, p_{h_i} is given by:

$$p_{h_i} = \frac{\lambda'_h{}^{(n_h-i)}}{\mu^{(n_h-i)}}\frac{(n_h)!}{i!}p_{h_{n_h}} \text{ with } (0 \leq i \leq n_h - 1)\ (5)$$

and $p_{h_{n_h}}$is given by:

$$p_{h_{n_h}} = \frac{1}{\frac{\sum_i \lambda'_h{}^{(n_h-i)}(n_h)!}{\mu^{(n_h-i)}\quad i!}}\ \ (6)$$

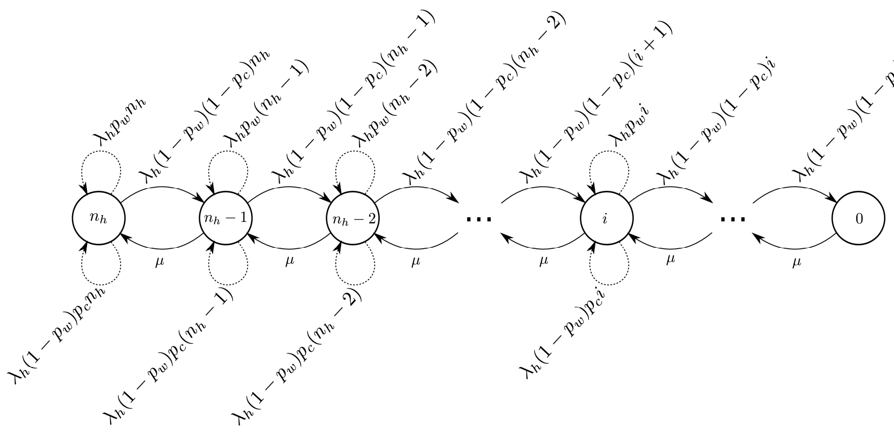
From the steady-state state probabilities, we can compute the mean number of PMs in the hot pool ($E[\#P_h]$) that needs to be exchanged with the other sub-models:

$$E[\#P_h] = \sum_i i\, p_{h_i}\ \ (7)$$

and the availability of the hot pool:

$$A_{k_h} = \sum_i p_{h_i}\ \ (8)$$

Similar solutions can be found for the warm and cold pools.



$\lambda_h p_w n_h$
$\lambda_h(1-p_w)(1-p_c)n_h$
$\lambda_h p_w(n_h-1)$
$\lambda_h(1-p_w)(1-p_c)(n_h-1)$
$\lambda_h p_w(n_h-2)$
$\lambda_h(1-p_w)(1-p_c)(n_h-2)$
$\lambda_h(1-p_w)(1-p_c)(i+1)$
$\lambda_h p_w i$
$\lambda_h(1-p_w)(1-p_c)i$
$\lambda_h(1-p_w)(1-p_c)$
n_h
n_h-1
n_h-2
$\cdots$
i
$\cdots$
0
μ
μ
μ
μ
μ
μ
$\lambda_h(1-p_w)p_c n_h$
$\lambda_h(1-p_w)p_c(n_h-1)$
$\lambda_h(1-p_w)p_c(n_h-2)$
$\lambda_h(1-p_w)p_c i$

Figure 6. Markov chain equivalent to the hot pool SRN sub-models

FIXED POINT ITERATION

To resolve the cyclic dependency among the interacting sub-models, we need to use a fixed point iteration approach. Fixed point iteration variables are reported in the import graph depicted in Figure 5. Our fixed point equation is given by:

$$\vec{x} = \vec{G}(\vec{x}) \text{ (9)}$$

where $\vec{x} = (p_w,\ p_c,\ E[\#P_h],\ E[\#P_w])$. It can be shown that all variables can be expressed as functions of p_w and p_c. Hence, the fixed point equation (9) can be rewritten as:

$$\vec{y} = \vec{F}(\vec{y}) \text{ (10)}$$

where $\vec{y} = (p_w, p_c)$.

Proof of existence of a solution to equation (10) implies the existence of a solution to equation (9). We use the Brouwer's fixed point theorem (Ortega & Rheinboldt, 1970): "*Let $\vec{F}: C \subset \mathbb{R}^2 \rightarrow \mathbb{R}^2$ be continuous on the compact, convex set C, and suppose that $\vec{F}(C) \subseteq C$. Then, $\vec{F}$ has a fixed point in C.*"

In our case, given that p_w and p_c are probabilities, we can define $C = \{\vec{y} = (p_w, p_c)$: $p_w \in [0,1]$, $p_c \in [0,1]\}$. By mean of Heine-Borel theorem, it is straighforward to demonstrate that set C is compact and convex. Moreover, given that the component functions of $\vec{F}$ are continuous in C, then also $\vec{F}$ is continuous in C. This proves the existence of a solution for the fixed point equation $\vec{y} = \vec{F}(\vec{y})$.

NUMERICAL RESULTS

SPNP (Hirel, Tuffin, & Trivedi, 2000) can be used to solve the SRN models. Results have been obtained by solving the models for a broad range of parameter space so that they can represent large variety of Clouds. However, for space limitation, here we report only interesting results. We assume MTTF of hot PMs to be in the range of 1−6 months, MTTF of warm PMs to be in the range of 3.5−12 months and MTTF of cold PMs to be in the range of 7 months - 2 years. MTTR of a PM can vary depending on type of repair process: (i) software based completely automated repair (1−30 minutes), (ii) completely manual repair (1−5 days) and (iii) combination of manual and automated repair (1−12 hours).

In Table IV, we report the state space and storage requirements for both the monolithic model and interacting sub-models. Monolithic model runs into a memory overflow problem when the number of PMs in each pool increases beyond 19. We observe that the state space size of the monolithic model increases quickly and becomes too large to construct the reachability graph even for small number of PMs. However, with interacting sub-models approach, the state space increases at a slower rate as the number of PMs in the system is increased. Table IV also shows a comparison of non-zero entries. These entries are number of non-zero elements in the infinitesimal generator matrix of the underlying continuous time Markov chain. For the same number of PMs, number of non-zero entries in interacting sub-models is 3 − 4 orders of magnitude smaller compared to the monolithic model. Observe that, for interacting sub-models, in both cases (i.e., number of states and number of non-zero entries), we report only the maximum value among the three sub-models. Since three sub-models are solved separately, we assumed that for a given execution only states and non-zero entries of only one sub-model are require to be stored in the memory. Reduction in state space and non-zero entries for interacting sub-models also leads to concomitant reduction in solution time needed. A comparison of solution times is shown in Table V. Solution time for monolithic model increases almost exponentially with the increase in model size. Solution time for interacting sub-models remains almost constant with the increase in model size.

#PMs in each pool in the beginning	#States in monolithic model	Maximum #states in interacting sub-models	#Non-zero entries in monolithic model	Maximum #non-zero entries in interacting sub-models
5	7056	56	44520	210
10	207636	286	1535490	1320
15	1775616	136	13948160	480
17	3508920	171	27976968	612
19	6468000	210	52189200	760
20	Memory overflow	231	Memory overflow	840
50	-	1326	-	5100
100	-	5151	-	20200
150	-	11476	-	45300
200	-	20301	-	80400

Table 4. Comparison of number of states and number of non-zero entries.

#PMs in each pool in the beginning	Monolithic model	Interacting sub-models
5	0.627	0.406
10	18.670	0.517
15	373.822	0.278
17	1004.494	0.279
19	2459.553	0.280
20	Memory overflow	0.281
50	-	0.296
100	-	0.377
150	-	0.564
200	-	0.948

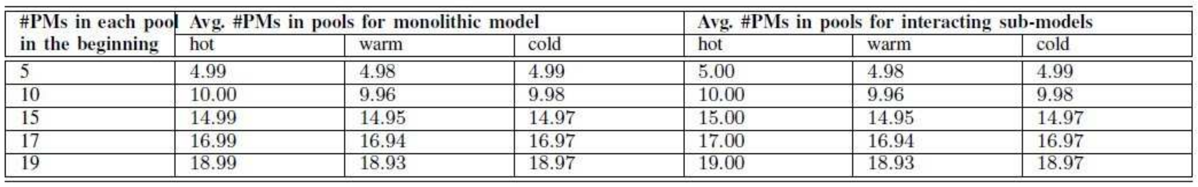
Table 5. Comparison of solution times in seconds.

In Table VI, we compare the downtime values as obtained from the monolithic model and interacting sub-models. We assume that Cloud is available if there are at least k "UP" PMs across all pools. For the example scenario investigated, we vary the value of k, with 10 PMs in each pool and 30 PMs in total. When k is 30, any failure of PM results in unavailability of Cloud service. For each value k, we also change the value of n_r which denotes maximum number of PMs that can be repaired in parallel. If n_r is 1, failed PMs are repaired serially, i.e., one after another. MTTFs of hot, warm and cold PMs were assumed to be 1000 hrs, 3500 hrs and 5000 hrs respectively. MTTR was assumed to be 3 hrs. Table VI shows that results obtained from the interacting sub-models are accurate. As expected, downtime values are higher with increasing in values of k. For each k, downtime reduces if we increase value of n_r. This gives rise to interesting optimization problems as discussed in Section XI.

Value of k	Value of n_r	Downtime (minutes per year)	
		Monolithic model	Interacting sub-models
30	1	23185.793	23178.956
	2	22904.919	22898.454
	3	22903.681	22897.219
29	1	792.475	798.651
	2	499.081	505.258
	3	497.787	503.964
28	1	24.722	25.336
	2	8.412	8.691
	3	7.118	7.396
27	1	0.740	0.778
	2	0.129	0.138
	3	0.081	0.087
26	1	0.022	0.024
	2	0.002	0.002
	3	0.0008	0.0009

Table 6. Comparison of downtime values in minutes per year.

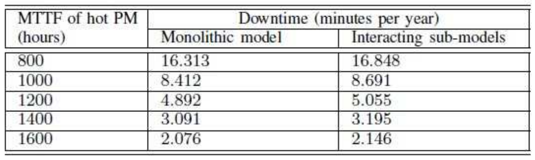
In Table VII, we show the mean number of PMs in each pool. MTTF and MTTR values for this case were assumed to same as in the Table VI. Value of n_r was assumed to be 1 for this example scenario. Results obtained from interacting sub-models are in good agreement with the results obtained from monolithic model. In Table VIII, we show the effect of changing MTTF of PMs on downtime. We assume 10 PMs in each pool (i.e., 30 PMs in total) and maximum number of parallel repairs in each pool is 2. In this example scenario, we further assume that Cloud service is available when at least 28 PMs across all pools are "UP". For each value MTTF of hot PM, MTTFs of warm and cold PM were assumed to be 3.5 times and 5 times greater than that of hot PM's MTTF.



#PMs in each pool in the beginning	Avg. #PMs in pools for monolithic model			Avg. #PMs in pools for interacting sub-models		
	hot	warm	cold	hot	warm	cold
5	4.99	4.98	4.99	5.00	4.98	4.99
10	10.00	9.96	9.98	10.00	9.96	9.98
15	14.99	14.95	14.97	15.00	14.95	14.97
17	16.99	16.94	16.97	17.00	16.94	16.97
19	18.99	18.93	18.97	19.00	18.93	18.97

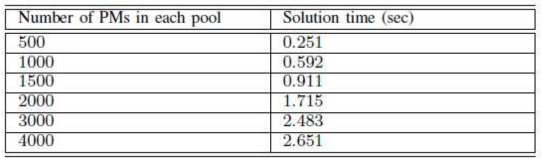
Table 7. Comparison of average number of PMs in each pool.

Results described so far were obtained by solving SRN models using SPNP. Next, using the closed form solutions for the interacting sub-models, we solve large scale models (order of thousands PMs in each pool). Table IX shows that solution time needed for solving large models increases very slowly with the model input size. Clearly, interacting sub-models approach facilitates availability analysis of large sized Clouds with a reasonably small solution time.



MTTF of hot PM (hours)	Downtime (minutes per year)	
	Monolithic model	Interacting sub-models
800	16.313	16.848
1000	8.412	8.691
1200	4.892	5.055
1400	3.091	3.195
1600	2.076	2.146

Table 8. Effect of varying MTTF of PMs with 10 PMs in each pool



Number of PMs in each pool	Solution time (sec)
500	0.251
1000	0.592
1500	0.911
2000	1.715
3000	2.483
4000	2.651

Table 9. Solution time required for availability analysis of large scale Cloud using closed-form.

DECOMPOSED MODELS IN CLOUD FEDERATIONS

Cloud federations are characterized by the presence of different small and medium Private Clouds, belonging to the same or to different organizations, that join each other to achieve a common goal, usually represented by the optimization of resources utilization. Public Clouds are usually used as backup when it is necessary to deal with load burst that cannot be managed by the Private federation. In (Bruneo, Longo, Puliafito, 2011) we showed a methodology based on the use of SRN monolithic models to investigate the more convenient strategies to manage a federation of two or more Private or Public Clouds. The final goal was to optimize energy consumption and performance in an energy-aware Green computing context. In this chapter, we presented a monolithic SRN that can be used to model the availability of a single Cloud infrastructure composed of a certain number of pools of PMs. Both the models are characterized by scalability problems. In fact, a greater number of Cloud infrastructures and/or PMs in the scenarios that it is necessary to model

corresponds to the raising of model states and to the impossibility to store and analyze the model in a common computer memory.
The solution to the state space explosion problem that we presented in the present chapter, i.e. to decompose the monolithic model in a set of interactive sub-models that can be analyzed by mean of fixed point iteration finds a perfect application in a federated environment. In fact, the presence of more than one Cloud infrastructure, each of which is characterized by two or more PM pools, would lead to an unmanageable monolithic model while is still tractable in a decomposed fashion. The advantages of the interactive sub-models approach are evident not only from an analysis point of view but also for what concerns clearness and conciseness of the representation power. In fact, while the use of SRN monolithic models to represent a Cloud federation scenario would lead to a complex and error-prone model with a huge number of places and transitions, the use of decomposed models limits the errors and the complexity of the representation. This is even more clear if we note that the main functional blocks of the system to be represented (in our case a Cloud infrastructure) are usually the same among different administrative domain. For example, warm and cold pools sub-models presented in Section VI are very similar and this characteristic is still valid if we consider different Clouds. This allows us to represent and solve a single sub-model for each of the functional block with different parameters and still be able to analyze the variety of behaviors that are present in a federated environment.

DISCUSSIONS AND FUTURE RESEARCH

In the previous section, we compared the analytic results obtained from monolithic PN model, interacting PN sub-models and closed form solutions of Markov chains. The models developed so far can be used by Cloud service provider during design, development, testing and operational phases of IaaS Cloud. During the design and development stage, the providers can use these models to determine the pool size required to offer a specific availability SLA. During the testing and operational stages, the providers can dynamically learn, how the repair strategy should be designed (i.e., number of parallel repairs, automated vs. manual repairs etc.) to maintain the promised availability SLA.
While, this chapter outlines the existing research on Cloud availability analysis and describes a novel approach for large scale analysis, many open research questions are yet to be solved. With the fast scalable approach for modeling availability, one can extend the performability analysis described in (Ghosh, Trivedi, Naik, & Kim, 2010) to large size IaaS Clouds. Characterizing the system behavior from such a coupled pure performance and availability models with thousands of PMs taking into account workload arrival, admission control, queueing, resource provisioning decisions, VM provisioning, and run-time execution in addition to the failures is challenging. Combining the performance model with availability model, the economics of failure-repair for a given utilization rate of the PMs needs to be determined. For different utilization and failure-repair rates, there are different break-even points between loss of revenue and repair costs, which can be determined and analyzed. Optimization problems on Cloud availability can also be developed. Table 6 shows that downtime can be reduced by increasing the maximum number repairs that can be done in parallel. Since, there is a cost associated with each repair, an optimal number of repair facilities required to minimize the repair cost for acceptable value of downtime can be determined. Our availability models also allow us to perform trade-off analysis of longer MTTF vs faster MTTR on system availability, the effect of having multiple concurrent repair facilities (i.e., higher labor costs) vs. higher availability but expensive components, repairing failed components vs replacing components for the a given service availability and so on. Another interesting type of analysis possible with this work is analyzing tradeoff between cost of availability SLAs vs operational costs including repair, replacement, and energy costs. These are important questions Cloud architects and designers often face. It is possible to answer these type of questions using tools based on the modeling and analysis techniques we describe in this chapter.

CONCLUSIONS

In this chapter, we showed how analytic models can be used to analyze Cloud based systems in the context of federation. After a deep analysis of the state of the art in such field, we showed how it is possible to overcame the state space explosion problem, deriving from the largeness of scale in such scenarios, by mean of an interactive sub-models approach. The reference context is the availability analysis in the presence of multiple class of server pools. The possibility to compute closed-form solutions for the sub-models and the managing of dependencies among sub-models through the use of fixed-point iterations make the approach able to analyze IaaS Cloud systems with thousands of PMs in order of seconds. This is indeed promising in a federated environment where a great number of Clouds, each of which is composed by a certain number of pools of PMs, interact among each other in order to fulfill their business goals. After discussing the possible applications and the usefulness of the approach for Cloud providers during design, development, testing and operational phases, open research questions that are yet to be solved have been summarized and future work have been indicated.

REFERENCES

Amazon EC2 (2011). http://aws.amazon.com/ec2.

Bonvin, N., Papaioannou, T. G., & Aberer, K. (2009). Dynamic cost efficient replications in data Clouds. In *1th ACM Workshop on Automated Control for Datacenters and Clouds* (ACDC) (pp. 49-56), Barcelona, Spain: ACM Press.

Bruneo, D., Longo, F., Puliafito, A. (2011). Evaluating energy consumption in a Cloud infrastructure. In *Proceedings of the 2011 IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks* (WoWMoM) (pp. 1-6), Lucca, Italy: IEEE Computer Society.

Chen, D., Dharmaraja, S., Chen, D., Li, L., Trivedi, K.S., Some, R.R., & Nikora, A.P. (2002). Reliability and availability analysis for the JPL remote exploration and experimentation system. In *Proceedings of the International Conference on Dependable Systems and Networks* (DSN) (pp. 337-342), Bethesda, MD, USA: IEEE Computer Society.

Chen, H., Zhou, C., & Xiong, N. (2010). Petri net modeling of the reconfigurable protocol stack for Cloud computing based control systems. In *Proceedings of the 1st International Conference on Cloud Computing* (CLOUDCOM) (pp. 393-400), Indianapolis, IN, USA: IEEE Computer Society.

Ciardo, G., Blakemore, A., Chimento, P.F., Muppala, J.K., & Trivedi, K.S. (1993). Automated generation and analysis of Markov reward models using stochastic reward nets. In *Linear Algebra, Markov Chains and Queuing Models* (pp. 145-191), New York, USA: Springer.

Ghosh, R., Longo, F., Naik, V.K., & Trivedi, K.S. (2010). Quantifying resiliency of IaaS Cloud. In *Proceedings of the IEEE Symposium on Reliable Distributed Systems* (SRDS) (pp. 343-347), Los Alamitos, CA, USA: IEEE Computer Society.

Ghosh, R., Trivedi, K.S., Naik, V.K., & Kim, D. S. (2010). End-to-End Performability analysis for Infrastructure-as-a-Service Cloud: An interacting stochastic models approach. In *Proceedings of the Pacific Rim International Symposium on Dependable Computing* (PRDC) (pp. 125-132), Tokyo, Japan: IEEE Computer Society.

Haring, G., Marie, R., Puigjaner, R., & Trivedi, K.S. (2001). Loss Formulae and Their Application to Optimization for Cellular Networks. In *IEEE Transaction on Vehicular Technology* (pp. 664-673), Washington, DC, USA: IEEE Computer Society.

Hirel, C., Tuffin, B., & Trivedi, K. S. (2000). SPNP: Stochastic Petri Nets. Version 6. In *Proceedings of the 11th International Conference on Technology of Object-Oriented Languages and Systems* (TOOLS) (p.354), Malaga, Spain: Springer.

IBM SmatCloud Enterprise (2001). www.ibm.com/services/us/en/cloud-enterprise/

Javadi, B., Kondo, D., Vincent, J., & Anderson, D. (2010). Discovering statistical models of availability in large distributed systems: An empirical study of seti@home. In *IEEE Transaction on Parallel and Distributed Systems* (pp. 1896-1903), Washington, DC, USA: IEEE Computer Society.

Joshi, K.R., Bunker, G., Jahanian, F., Moorsel, A.P.A.V., & Weinman, J. (2009). Dependability in the Cloud: Challenges and opportunities. In *Proceedings of the 39th annual IEEE/IFIP International Conference on Dependable Systems and Networks* (DSN) (pp. 103-104), Lisbon, Portugal: IEEE Computer Society.

Kim, D.S., Machida, F., & Trivedi, K.S. (2009). Availability Modeling and Analysis of a Virtualized System. In *Proceedings of the Pacific Rim International Symposium on Dependable Computing* (PRDC) (pp. 365-371), Seoul, Korea: IEEE Computer Society.

Kulkarni, V.G. (2010). Modeling and Analysis of Stochastic Systems, Boca Raton, FL, USA: CRC Press - Taylor & Francis Group.

Longo, F., Ghosh, R., Naik, V.K., & Trivedi, K.S. (2011). A Scalable Availability Model for Infrastructure-as-a-Service Cloud. In *Proceedings of the 41st annual IEEE/IFIP International Conference on Dependable Systems and Networks* (DSN) (pp. 335-346), Hong Kong, China: IEEE Computer Society.

Mainkar, V., & Trivedi, K.S. (1996). Sufficient conditions for existence of a fixed point in stochastic reward net-based iterative models. In *IEEE Transactions on Software Engineering* (pp. 640–653), Washington, DC, USA: IEEE Computer Society.

Marsan, M.A., Balbo, G., & Conte, G. (1984). A class of generalized stochastic petri nets for the performance evaluation of the multiprocessor systems. In *ACM Transactions on Computer Systems* (pp. 93-122), New York, USA: ACM Press.

Nicol, D.M., Sanders, W.H., & Trivedi, K.S. (2004). Model-based evaluation: From dependability to security. In *IEEE Transactions on Dependable and Secure Computing* (pp. 48-65), Washington, DC, USA, IEEE Computer Society.

Ortega, J.M., & Rheinboldt, W.C. (1970). Iterative Solution of Nonlinear Equations in Several Variables. New York, USA: Academic Press.

Smith, W.E., Trivedi, K.S., Tomek, L.A., & Ackaret, J. (2008). Availability analysis of blade server systems. In *IBM Systems Journal* (pp. 621-640), Indianapolis, IN, USA: IBM Press.

Tomek, L., Muppala, J., & Trivedi, K.S. (1993). Modeling Correlation in Software Recovery Blocks. In *IEEE Transactions on Software Engineering* (pp. 1071-1086), Washington, DC, USA: IEEE Computer Society.

Tan, Y., Gu, X., & Wang, H. (2010). Adaptive system anomaly prediction for large-scale hosting infrastructures. In *Proceedings of the 29th ACM Symposium on Principles of distributed computing* (PODC) (pp. 173-182), Zurich, Switzerland: ACM Press.

Tomek, L., & Trivedi, K.S. (1991). Fixed-Point Iteration in Availability Modeling. In *Informatik-Fachberichte, Vol. 283* (pp. 229-240), Berlin, Germany: Springer-Verlag.

Trivedi, K.S. (2001). Probability and Statistics with Reliability, Queuing, and Computer Science Applications. New York, USA: John Wiley and Sons.

Trivedi, K.S., Kim, D.S., Roy, A., & Medhi, D. (2009). Dependability and security models. In *Proceedings of the 7th International Workshop on the Design of Reliable Communication Networks* (DRCN) (pp. 11-20), Washington, DC, USA: IEEE Computer Society.

Trivedi, K.S., & Sahner, R. (2009). SHARPE at the age of twenty two. In *Sigmetrics Performance Evaluation Review* (pp. 52-57), New York, USA: ACM Press.

Trivedi, K.S., Vasireddy, R., Trindade, D., Nathan, S., & Castro, R. (2006). Modeling High Availability Systems. In *Proceedings of the Pacific Rim International Symposium on Dependable Computing* (PRDC) (pp. 154-164), Riverside, CA, USA: IEEE Computer Society.

Trivedi, K.S., Wang, D., Hunt, D.J., Rindos, A., Smith, W.E., & Vashaw, B. (2008). Availability Modeling of SIP Protocol on IBM WebSphere. In *Proceedings of the Pacific Rim International Symposium on Dependable Computing* (PRDC) (pp. 323-330), Taipei, Taiwan: IEEE Computer Society.

Uemura, T., Dohi, T., & Kaio, N. (2009). Availability analysis of a scalable intrusion tolerant architecture with two detection modes. In *Proceedings of the 1st International Conference on Cloud Computing* (CLOUDCOM) (pp. 178-189), Beijing, China: Springer.

Vishwanath, K.V., & Nagappan, N. (2010). Characterizing Cloud computing hardware reliability. In *Proceedings of the ACM Symposium on Cloud Computing* (SOCC) (pp. 193-204), Indianapolis, IN, USA: ACM Press.

Wang, D., Fricks, R.M., & Trivedi, K.S. (2003). Dealing with non-exponential distributions in dependability models. In *Performance Evaluation - Stories and Perspectives* (pp. 273-302), Vienna, Austria: Oesterreichchische Computer Gessellschaft.

Yang, B., Tan, F., Dai, Y.S., & Guo, S. (2009). Performance evaluation of Cloud service considering fault recovery. In *Proceedings of the 1st International Conference on Cloud Computing* (CLOUDCOM) (pp. 571-576), Beijing, China: Springer.

Yin, L., Fricks, R.M., & Trivedi, K.S. (2002). Application of semi-Markov process and CTMC to evaluation of UPS system availability. In *Proceedings of the Annual Reliability and Maintainability Symposium* (pp. 584-591), Seattle, WA, USA: IEEE Computer Society.

ADDITIONAL READING

Chen, D., Trivedi, K.S. (2001). Analysis of Periodic Preventive Maintenance with General System Failure Distribution. In Proceedings of the Pacific Rim International Symposium on Dependable Computing (PRDC) (pp. 103-107), Seoul, Korea: IEEE Computer Society.

Das, C.R., Mohapatra, P., Tien, L., Bhuyan, L.N. (1993). An availability model for MIN-based multiprocessors. In IEEE Transaction on Parallel and Distributed Systems (pp. 1118-1129), Washington, DC, USA: IEEE Computer Society.

Floyd, F., Hawkins, M. (2001). High Availability: Design, Techniques, and Processes. Upper Saddle River, NJ, USA: Prentice Hall.

Fricks, R., Trivedi, K.S. (1997). Modeling Failure Dependencies in Reliability Analysis Using Stochastic Petri Nets. In Proceedings of the 11th European Simulation Multiconference (ESM), Istanbul, Turkey: ACM Press.

Goldschmidt, T., Dittrich, A., Malek, M. (2009). Quantifying Criticality of Dependability-Related IT Organization Processes in CobiT. In Proceedings of the Pacific Rim International Symposium on Dependable Computing (PRDC) (pp. 336-341), Seoul, Korea: IEEE Computer Society.

Goseva-Popstojanov, K., Trivedi, K.S. (2000). Stochastic Modeling Formalisms for Dependability, Performance and Performability. In Performance Evaluation: Origins and Directions (pp. 403-422), Berlin, Germany: Springer-Verlag.

Grottke, M., Nikora, A.P., Trivedi, K.S. (2010). An Empirical Investigation of Fault Types in Space Mission System Software. In Proceedings of the International Conference on Dependable Systems and Networks (DSN) (pp. 447-456), Fairmont Chicago – Millennium Park, Chicago, IL, USA: IEEE Computer Society.

Haberkorn, M., Trivedi, K.S. (2007). Availability Monitor for a Software Based System. In Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium (HASE) (pp. 321-328), Dallas, TX, USA: IEEE Computer Society.

Kim, D.S., Machida, F., Trivedi, K.S. (2009). Availability Modeling and Analysis of a Virtualized System.
In Proceedings of the Pacific Rim International Symposium on Dependable Computing (PRDC) (pp. 365-371), Seoul, Korea: IEEE Computer Society.

Lanus, M., Yin, L., Trivedi, K.S. (2003). Hierarchical Composition and Aggregation of State-Based Availability and Performability Models. In IEEE Transaction on Reliability (pp. 44-52), Washington, DC, USA: IEEE Computer Society.

Lee, F., Marathe, M. (1999). Beyond Redundancy: A guide to designing high-availability networks. Indianapolis, ID, USA: Cisco Press.

Malek, M. (2008). Online Dependability Assessment through Runtime Monitoring and Prediction, Panel Contribution. In Proceedings of the 7th European Dependable Computing Conference (EDCC) (p.181), Kaunas, Lithuania: IEEE Computer Society.

Malhotra, M., Trivedi, K.S. (1994). Power-Hierarchy of Dependability-Model Types. In IEEE Transactions on Reliability (pp.493-502), Washington, DC, USA: IEEE Computer Society.

Malhotra, M., Trivedi, K.S. (1995). Dependability Modeling Using Petri-Nets. In IEEE Transactions on Reliability (pp.428-440), Washington, DC, USA: IEEE Computer Society.

Mendiratta, V. (1999). Reliability Analysis of Clustered Computing Systems. In Proceedings of the International Symposium on Software Reliability Engineering (ISSRE) (pp.268-272), Boca Reton, FL, USA: IEEE Computer Society.

Mishra, K., Trivedi, K.S. (2006). Model Based Approach for Autonomic Availability Management. In Service Availability (pp.1-16), Berlin, Germany: Springer-Verlag.

Muppala, J., Sathaye, A., Howe, R., Trivedi, K.S. (1992). Dependability modeling of a heterogeneous VAXcluster system using stochastic reward nets. In Hardware and Software Fault Tolerance in Parallel Computing Systems, New York, USA: Ellis Horwood.

Muppala, J., Ciardo, G., Trivedi, K.S. (1994). Stochastic Reward Nets for Reliability Prediction. In Reliability, Maintainability and Serviceability (pp.9-20), Washington, DC, USA: SAE International.

Pecchia, A., Cotroneo, D., Kalbarczyk, Z., Iyer, R.K. (2011). Improving Log-Based Field Failure Data Analysis of Multi-Node Computing Systems. In Proceedings of the International Conference on Dependable Systems and Networks (DSN) (pp.97-108), Hong Kong, China: IEEE Computer Society.

Salfner, F., Lenk, M., Malek, M. (2010). A survey of Online Failure Prediction Methods. In ACM Computing Surveys (pp.1-42), New York, USA: ACM Press.

Sahner, R.A., Trivedi, K.S., Puliafito, A. (1996). Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package, Norwell, MA, USA: Kluwer Academic Publishers.

Smith, R.M., Trivedi, K.S., Ramesh, A.V. (1988). Performability Analysis: Measures, an Algorithm, and a Case Study. In IEEE Transactions on Computers (pp. 406-417), Washington, DC, USA: IEEE Computer Society.

Stewart, W.J. (1994). Introduction to the Numerical Solution of Markov Chains, Princeton, NJ, USA: Princeton University Press.

Trivedi, K.S., Kim, D.S., Roy, A., Medhi, D. (2009). Dependability and security models. In Proceedings of the 7th International Workshop on the Design of Reliable Communication Networks (DRCN) (pp. 11-20), Washington, DC, USA: IEEE Computer Society.

Trivedi, K.S., Kim. D.S., Yin, X. (2012). Multi-State Availability Modeling in Practice. In Recent Advances in System Reliability (pp.165-180), Berlin, Germany: Springer-Verlag.

KEY TERMS & DEFINITIONS

Availability: In IaaS Cloud context, availability is the degree to which the Cloud infrastructure is able to provide VMs to clients in the face of hardware/software failures and repairs. Simply put, availability is the proportion of time the IaaS Cloud is in a functioning condition.

Analytic model: In performance and availability evaluation context, an analytic model is a mathematical tool that allows to do "what-if" analysis and predict the behavior of a real system. In contrast with simulation and measurement based approach, can provide results in smaller time.

Petri nets: Petri nets are high-level, graphical and mathematical formalism that allows a modeler to describe a system without explicitly enumerating its states. Compared to Markov chains and other state space formalisms, Petri nets are easy to understand for non-expert users but maintain an exact mathematical definition of their execution semantics, with a well-developed mathematical theory for process analysis. Petri-nets are used when hand-generation of Markov chain is difficult.

Stochastic reward nets: An extension of generalized stochastic Petri nets in which every state of the underlying Markov chain can be associated with a reward rate thus facilitating the computation of a variety of performance and availability measures.

Monolithic model: A single self-contained analytic model that allows to describe the behavior of a system and predict its performance and/or availability characteristics as a whole. In many cases, accurate analytic analysis with monolithic models usually requires dealing with large number of system states, leading to state-space explosion problem.

Interacting sub-models: A possible solution to the state-space explosion problem posed by monolithic model is to decompose the monolithic model to a set of sub-models each of which describes the behavior of one of the system components. The sub-models interact among themselves by exchanging parameters. Such interactions can be described by an import graph.

Fixed point iterations: A technique that allows to resolve cyclic dependencies among interacting sub-models to obtain model solutions without incurring significant errors. Fixed-point iterations are used to solve the equations where the unknown variables can be expressed only in implicit form. Three mathematical issues important to fixed-point iterations are: (i) existence of a solution, (ii) uniqueness of a solution and (iii) rate of convergence.