Scalable Analytics for IaaS Cloud Availability

Article *in* IEEE Transactions on Cloud Computing · January 2014

DOI: 10.1109/TCC.2014.2310737

5 authors, including:

Rahul Ghosh
Xerox Corporation
24 PUBLICATIONS **278** CITATIONS
SEE PROFILE

Francesco Longo
Università degli Studi di Messina
57 PUBLICATIONS **343** CITATIONS
SEE PROFILE

Flavio Frattini
University of Naples Federico II
16 PUBLICATIONS **30** CITATIONS
SEE PROFILE

Kishor S Trivedi
Duke University
745 PUBLICATIONS **24,319** CITATIONS
SEE PROFILE

Scalable Analytics for IaaS Cloud Availability

Rahul Ghosh, Francesco Longo, Flavio Frattini, Stefano Russo, and Kishor S. Trivedi

Abstract—In a large Infrastructure-as-a-Service (IaaS) cloud, component failures are quite common. Such failures may lead to occasional system downtime and eventual violation of Service Level Agreements (SLAs) on the cloud service availability. The availability analysis of the underlying infrastructure is useful to the service provider to design a system capable of providing a defined SLA, as well as to evaluate the capabilities of an existing one. This paper presents a scalable, stochastic model-driven approach to quantify the availability of a large-scale IaaS cloud, where failures are typically dealt with through migration of physical machines among three pools: hot (running), warm (turned on, but not ready), and cold (turned off). Since monolithic models do not scale for large systems, we use an interacting Markov chain based approach to demonstrate the reduction in the complexity of analysis and the solution time. The three pools are modeled by interacting sub-models. Dependencies among them are resolved using fixed-point iteration, for which existence of a solution is proved. The analytic-numeric solutions obtained from the proposed approach and from the monolithic model are compared. We show that the errors introduced by interacting sub-models are insignificant and that our approach can handle very large size IaaS clouds. The simulative solution is also considered for the proposed model, and solution time of the methods are compared.

Index Terms—Analytic-numeric solution, availability, downtime, cloud computing, simulation, stochastic reward nets

1 Introduction

Component failures in a large distributed environment are quite common phenomena. Nevertheless, large service providers' data centers should be designed to guarantee a certain level of availability to the consumer. Infrastructure-as-a-Service (IaaS) cloud provides computational resources (e.g., CPU and memory), storage resources, and networking capacity that promise high availability in face of such failures. Service availability (i.e., the probability of receiving the proper service at any given time) is usually specified in Service Level Agreements (SLAs) as downtime in minutes per year or as the percentage of time the service will be up through out the year. Hence, cloud service providers need to perform an availability analysis to quantify the expected downtime that the service may experience over a period of time.

Cloud availability analysis can be performed through proper modeling techniques. Among these, state-space models are popular as they capture complicated interactions among system components and different failure/repair behaviors. However, for a large IaaS cloud, the model state space tends to be too large. Monolithic or one-level Markov chains are a typical modeling formalism, representative of the state-of-the-art in cloud availability modeling. The growth of the state space as the model takes into account more details of the system is known as the largeness problem of Markov models [1]. Stochastic Petri Nets (SPNs) can be used to tolerate the largeness problem, as they allow the automated generation of the (underlying) Markov model. Still, the solution of large models is an issue.

In this paper, we consider an IaaS cloud where physical machines (PMs) are grouped into three pools based on power consumption and provisioning delay characteristics. PMs can be migrated from one pool to another due to failure/repair events. We evaluated the availability of a similar system in [2], where some simplistic assumptions (see Section 2) restricted the applicability to limited scenarios. After relaxing those simplistic model assumptions, we first follow the common approach of developing a *single monolithic model* using a variant of SPNs called Stochastic Reward Nets (SRNs) [3]. However, such a monolithic model is not scalable for large clouds due to the large state-space of the underlying Markov chain. To resolve this issue, we present a scalable stochastic modeling approach based on *interacting sub-models*.

Note that, many of the existing published models [4], [5] are hierarchical in nature. In our case, complexity and characteristics of large IaaS clouds (e.g., migration of PMs from one pool to another) lead to cyclic dependency among the sub-models, needing fixed-point iteration. Another solution to the largeness problem is shown where underlying large Markov model generation is avoided by directly solving the stochastic net by means of *simulation*. We show the effectiveness of this approach for solving very large state-space models.

After carefully reviewing the related research (Section 2) we identify that the previous research did not address the issue of scalability in terms of accuracy and solution time in solving large IaaS cloud availability models. Thus, the contributions of this paper are the following:

- *R. Ghosh is with IBM, Durham, North Carolina. E-mail: rghosh@us.ibm.com.*
- *F. Longo is with the Dipartimento di Matematica, Università degli Studi di Messina, Contrada di Dio - S.Agata, Messina 98164, Italy. E-mail: flongo@unime.it.*
- *F. Frattini and S. Russo are with the Department of Electrical Engineering and Information Technology, Università degli Studi di Napoli Federico II, Via Claudio 21, 80125, Naples, Italy. E-mail: {flavio.frattini, stefano.russo}@unina.it.*
- *K.S. Trivedi is with the Department of Electrical and Computer Engineering, Duke University, 206 Hudson Hall, Durham, NC 27708-0291. E-mail: kst@ee.duke.edu.*

Manuscript received 17 Sept. 2013; revised 19 Dec. 2013; accepted 20 Feb. 2014; date of publication xx xx xxxx; date of current version xx xx xxxx. Recommended for acceptance by X. Zhou.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TCC.2014.2310737

1. We state the availability assessment problem for an IaaS cloud (Section 3) and propose a realistic monolithic model representative of the state-of-the-art (Section 4).
2. We propose an interacting sub-models approach to solve the largeness problem of the monolithic availability model (Section 5). The overall model solution is obtained by fixed-point iteration over individual sub-model solutions (Section 6).
3. For selected software, hardware, and parameters (Section 7), the interacting sub-models approach is compared w.r.t. the common modeling approach of adopting a single monolithic model (Section 8). The simulation of the SRN monolithic model is also considered to avoid the generation of the large state-space of the Markov chain (Section 9). Results show that the monolithic model becomes intractable and fails to produce results as the size of the cloud system increases and demonstrate the scalability of the interacting stochastic sub-models approach and of the simulation. Moreover, we show that for very large systems, simulation results are obtained faster than the numerical solution of the interacting sub-models. The accuracy of decomposition and simulation is verified by comparing their solutions against the analytic-numeric solution of the monolithic model.
4. An additional advantage of the interacting sub-models approach is that sub-models often become simple enough to afford a closed-form solution (Section 10). Such closed-form expressions can complement the use of stochastic modeling software packages such as SHARPE [6] and SPNP [7], when dealing with large number of model states.

2 Related Research

The largeness and scalability problems of complex models are described in [8]. To solve such problems, the hierarchical composition is introduced in [9] (and many other papers and books), where a two-level hierarchical model is proposed. Each subsystem is modeled by a Markov chain and the system reliability is modeled by a series system of independent Markov components. In [10], simulation is shown as an alternative to analytic-numeric methods when high computational requirements forbid their use. The paper illustrates advantages and drawbacks of each method and provides some hints about which is to be chosen.

In the specific case of cloud computing, some modeling approaches focusing on dependability aspects have been proposed in recent years. Nevertheless, these works do not take the scalability and largeness issues into account and hence they are complementary to this paper.

We proposed an availability model similar to the one presented in this paper in [2]. However, the model was simplified assuming the migration process to be instantaneous, hence migration delay was considered equal to zero. In the current paper, we introduce the migration delay in the monolithic model and in the interacting sub-models making the model significantly different and more accurate and realistic. Moreover, the introduction of the migration delay makes the availability model more complex to be analyzed. As a matter of fact, time for analytic-numeric solution largely increases when considering systems with 100 or more machines per pool, even with the interacting sub-models approach. Hence, besides comparing the analytic-numeric results between the monolithic and interacting sub-models, in this paper also the simulative solution is considered. The three approaches are compared in terms of number of states and of non-zero entries of the underlying Markov chain, values of output measures (availability and downtime), and solution time. Thus, the paper offers a comprehensive comparison among multiple modeling/solution techniques for stochastic models showing advantages and disadvantages of all of them.

In [11], Yang et al. investigate the impact of fault recovery on cloud service performance. They consider response time as the performance metric and model the service on it. Although joint analysis of availability and performance is important, the authors do not address the scalability issue which is the main focus of our work.

A discrete time semi-Markov process is used in [12] to describe the stochastic behavior of a scalable intrusion tolerant system that aims at increasing the steady-state availability. In contrast, for cloud availability analysis, we start with Petri net 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 [13], authors address the problem of VMs scheduling using a multivariate probabilistic model. While other techniques only consider static allocation, the paper faces reallocation optimization considering several factors (such as CPU, I/O, memory). Also this work is complementary to our research since it may provide useful insights in the identification of PMs that are to be migrated from a pool to another to guarantee good resource allocation.

In [14], Vishwanath and Nagappan investigate failure characteristics of servers in large cloud data centers. Authors empirically found failure/repair rates for the hardware of the PMs in cloud systems. The scalable stochastic approach that we describe can be complementary to this work as the measured failure/repair rates of hardware components can be used to parameterize the model we propose.

In [15], Bonvin et al. design a reliable and cost-effective storage system (Skute) based on data replicas, which maintains high availability guarantees despite failures of servers. The authors also address interesting cost-optimization questions. Similar optimization problems can be formulated using the models and approach described in this paper.

In [16], SPNs and reliability block diagrams are used for the quantification of sustainability impact, cost and dependability of data center cooling infrastructures. However, the paper only focuses on the cooling system and does not face scalability issues for availability evaluation.

In general, there are limited research efforts which investigated availability in large scale infrastructure.

In [17], an anomaly prediction system (ALERT) for achieving robust hosting infrastructures is proposed. The ALERT system is complementary to our modeling approach in the identification of anomalies that require machines migrations.

A large distributed system is analyzed in [18] aiming at finding better scheduling algorithms. Authors propose a

methodology to find host subsets with similar stochastic behavior and to model the system considering such different subsets. Although such a methodology may be useful to model sets of PMs or VMs, we assumed identical machines, hence one failure distribution is considered. Furthermore, the scheduling problem is not the focus of our paper.

In [19], Chen et al. use a deterministic and stochastic PN method to illustrate the performance of producer/consumer based application models in the cloud context. Unlike the approach proposed here, they focus only on cloud performance.

3 Problem Statement

A cloud service provisions VMs with specific characteristics in terms of number and frequency of CPU cores, memory, and storage. Such VMs are deployed on the PMs of a data center. For backup and recovery, as well as for reducing power consumption, PMs in data centers are organized in pools. We assume the existence of three pools, namely *hot* (running PMs), *warm* (turned on, but not ready PMs), and *cold* (turned off PMs). When using default VM images, the deployment on hot PMs can be performed with minimum provisioning delay. The deployment on a warm PM requires additional provisioning time (to make the PM ready). Further delay is added when using PMs in the cold pool (which need to be turned on before being used).

Focus of this paper is to develop a scalable availability model for an IaaS cloud with three pools. Note that the architecture that we take into consideration is not tied to a specific cloud implementation; rather it is a generic one on top of which several real cloud infrastructures can be mapped. Assume that n_h, n_w, and n_c are the initial number of non-failed PMs in the hot, warm, and cold pools, respectively. We consider the system being available if at least k PMs are properly working in the hot pool ($1 \leq k \leq n_h$). As additional measures of the availability of the system, we consider (i) the steady state average number of PMs in each pool, and (ii) the steady state system downtime when varying the number of PMs in the pools.

Initially, we develop a monolithic model of the system. To automate the construction of the underlying Markov chain, we use SRN [3]. Since obtaining solution from the monolithic model is difficult for large cloud systems, our eventual goal is to develop a scalable model for IaaS cloud. To overcome this difficulty, fixed-point iterative models [9], as well as simulation are used.

In our modeling, the following assumptions are made for failures, repairs, and migrations.

1. Several types of failures can take place in a cloud system such as software failures, hardware failures, or network failures [20]. Our model considers the overall effect of these possible failures in PMs with an aggregated mean time to failure (MTTF) [1], [21].
2. For the hot, warm, and cold pools, TTF distributions of PMs are exponential with rates λ_h, λ_w, and λ_c, respectively. It is reasonable to consider $\lambda_h > \lambda_w$ and $\lambda_w \gg \lambda_c$.
3. Hot, warm, and cold pools contain identical PMs. Upon failure of a PM in the hot pool, the failed PM is removed from the hot pool for repairing and replaced by one in the warm pool, if available. In case the warm pool is empty, a migration from the cold pool to the hot pool is done, if a PM in the cold pool is available. If a warm PM fails, the PM is removed from the warm pool for repairing. If at least one PM is present in the cold pool, it replaces the failed one in the warm pool. Once repaired, a PM is moved to its original pool and PMs migrated from other pool are returned to their original pool, for charging/accounting purposes. Migration of PMs from one pool to another introduces the interdependencies among the pools.
4. Time to migrate a PM from one pool to another is exponentially distributed. Mean times to migrate (MTTM) from warm and cold pools to hot pool are $1/\gamma_{wh}$ and $1/\gamma_{ch}$, respectively. MTTMs from hot and cold pools to warm pool are $1/\gamma_{hw}$ and $1/\gamma_{cw}$, respectively. MTTMs from hot and warm pools to cold pool are $1/\gamma_{hc}$ and $1/\gamma_{wc}$, respectively.
5. Repairs can be performed in parallel in individual pools. Maximum number of PMs that can be repaired in parallel is $n_r \geq 1$, with $n_r \leq n_h$, $n_r \leq n_w$, and $n_r \leq n_c$. If the number of PMs to be repaired is higher than n_r, failed PMs are put in a queue for repair.
6. PM repair time is exponentially distributed with mean time to repair (MTTR) $1/\mu$.

All sub-models are homogeneous continuous time Markov chains (CTMC) and inter-event times are exponentially distributed. This assumption can be relaxed as described in [1], [22].

The three approaches, namely monolithic model, interacting sub-models, and simulation, are compared considering:

1. errors introduced by the analytic-numeric solution of the interacting sub-models and the simulation of the monolithic model with respect to the analytic-numeric solution of the monolithic model;
2. scalability of each approach in terms of number of PMs in each pool; and
3. time required for the model solution.

Interacting sub-models approach also allows to compute closed form solution, when migration process is instantaneous. This is discussed in Section 10.

4 Monolithic Availability Model

Fig. 1 shows a monolithic SRN model for the availability analysis of IaaS cloud. Input parameters for such a 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 ($1/\mu$), and 5) MTTMs of PMs among pools ($1/\gamma_{wh}, 1/\gamma_{ch}, 1/\gamma_{hw}, 1/\gamma_{cw}, 1/\gamma_{hc}, 1/\gamma_{wc}$). While n_h, n_w, n_c, and n_r are design parameters, MTTF, MTTR, and MTTM values can be experimentally measured. Guard functions of the model are in Table 1.

Hot, warm, and cold pool are modeled by places P_h, P_w, and P_c, respectively and the number of tokens in such places represent the number of non-failed PMs in the

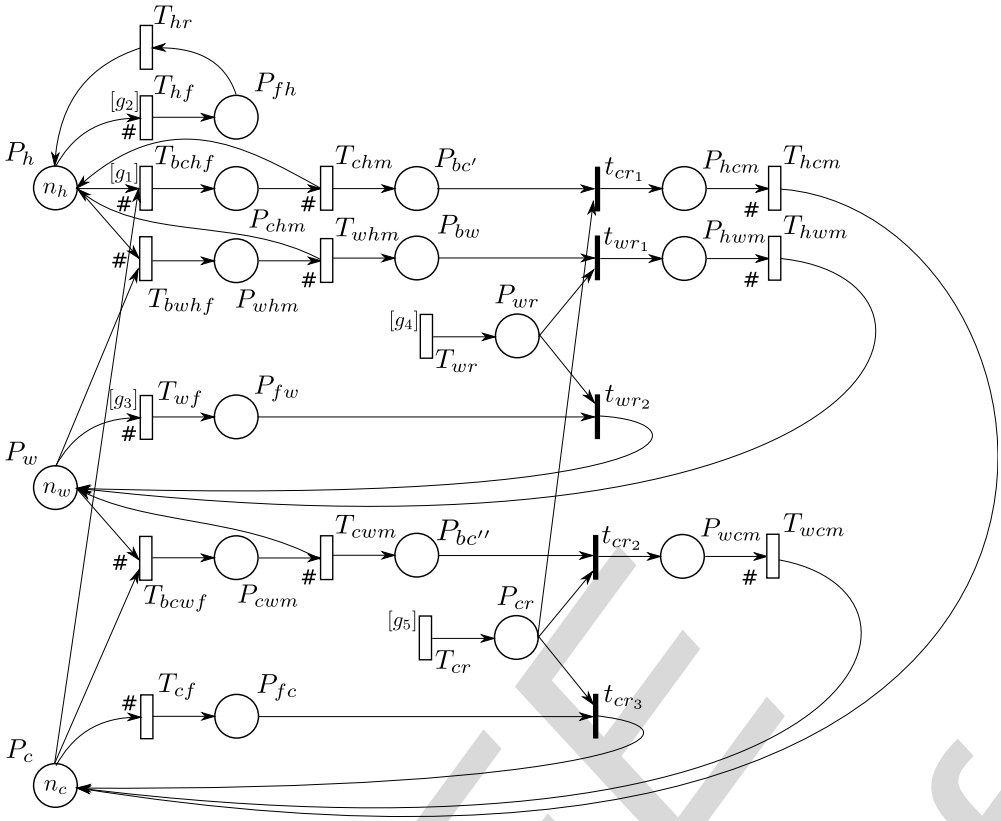

Fig. 1. Monolithic SRN model for availability analysis of IaaS cloud.

associated pool. The failure event of a hot PM is represented by the firing of transitions T_{bwhf}, T_{bchf}, and T_{hf}. When a hot PM fails, migration of a PM from a different pool is attempted and three cases can arise: 1) a warm PM is available for migration to the hot pool (T_{bwhf} fires), 2) the warm pool is empty but a cold PM can be migrated (T_{bchf} fires), and 3) both the warm and the cold pool are empty and the failed hot PM cannot be substituted by any other PM (T_{hf} fires). The three mutually exclusive cases are modeled by guard functions $[g_1]$ and $[g_2]$. Since the overall hot PM failure rate should be equal to λ_h multiplied by the number of available hot PMs, rates of the above mentioned transitions are considered to be dependent on the number of tokens in place P_h as reported in Table 2. The # symbol near the input arcs is used to represent such marking dependent firing rates.

Tokens in places P_{whm}, P_{chm}, and P_{cwm} represent PMs waiting until the migration to the hot pool is completed. In particular, when transition T_{bwhf} fires, one token is taken from places P_w and P_h each and one token is put in place P_{whm}, modeling a warm PM borrowed for migration to the hot pool. Subsequently, upon firing of transition T_{whm} (migration completed), one token is removed from place P_{whm} and one token is deposited to places P_h and P_{bw} each. Rate of transition T_{whm} is dependent on the number of tokens in place P_{whm} modeling the migration process being performed in parallel for all migrating PMs. Place P_{bw} 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}, one token is removed from places P_c and P_h each and one token is deposited to place P_{chm}. Upon firing of transition T_{chm}, one

TABLE 1
Guard Functions for Monolithic and Interacting SRN Sub-Models

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 2
Rates of Transitions Modeling the Failure of PMs in Monolithic SRN Model

Transitions	Rates of transitions
T_{hf}	$\#P_h \cdot \lambda_h$
T_{bchf}	$\#P_h \cdot \lambda_h$
T_{bwhf}	$\#P_h \cdot \lambda_h$
T_{wf}	$\#P_w \cdot \lambda_w$
T_{bcwf}	$\#P_w \cdot \lambda_w$
T_{cf}	$\#P_c \cdot \lambda_c$

TABLE 3
Rates of Transitions Modeling the Repair of Failed PMs in Monolithic SRN Model and Interacting SRN Sub-Models

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

token is removed from place P_{chm} and one token is deposited to places P_h and $P_{bc'}$ each. Rate of transition T_{chm} is dependent on the number of tokens in place P_{chm}. Place $P_{bc'}$ keeps track of number of failed PMs that need to be repaired and given back to the cold pool at the end of the repair process. When transition T_{hf} fires, a token is removed from place P_h modeling the reduction in number of available PMs in the hot pool by one and a token is deposited in place P_{fh} representing the failed PM that has to be repaired and given back to the hot pool.

Failure, repair, and migration of PMs in the warm pool are represented in a similar way. The failure event of a warm PM is modeled by transitions T_{bcwf} and T_{wf}. If a cold PM is available for migration to warm pool, transition T_{bcwf} fires, while, if the cold pool is empty, T_{wf} fires. Mutual exclusion between the two cases is assured by guard function $[g_3]$. Given that the overall warm PM failure rate should be equal to λ_w multiplied by the number of available warm PMs, rates of transitions T_{bcwf} and T_{wf} are considered to be dependent on the number of tokens in place P_w as reported in Table 2. When transition T_{bcwf} fires, a token is taken from place P_c and it is deposited to place P_{cwm}. In place P_{cwm}, cold PMs are kept in hold until the migration to the warm pool is completed. Upon firing of transition T_{cwm}, one token is removed from place P_{cwm} and one token is deposited to places P_w and $P_{bc''}$. The latter models the failed PM that have to be repaired and migrated back to the cold pool. Upon firing of T_{wf}, a token moves from place P_w to place P_{fw}. This models the failed PM that have to be repaired and migrated back to the warm pool.

Firing of transition T_{cf} models the failure of a cold PM being its rate dependent on the number of tokens in place P_c (see Table 2). In this way, the overall cold PM failure rate is equal to λ_c multiplied by the number of available cold PMs. When transition T_{cf} fires, a token moves from place P_c to place P_{fc}.

Repair of failed PMs is represented by transitions T_{hr}, T_{wr}, and T_{cr} whose rates are marking dependent in order to model the presence of n_r repair facilities for each pool (see Table 3). Transitions T_{wr} and T_{cr} have to be enabled only when at least one PM needs to be repaired. This is assured by guard functions $[g_4]$ and $[g_5]$. Immediate transitions t_{wr_1}, t_{wr_2}, t_{cr_1}, t_{cr_2}, and t_{cr_3} model the fact that the migrations of repaired PMs to the original pool start immediately after the repair process is completed. Places P_{hcm}, P_{hwm}, and P_{wcm} model the holding of repaired PMs for migrations. Migration is modeled by transitions T_{hcm}, T_{hwm}, and T_{wcm} that have place dependent firing rates. In Table 4, we summarize the rates of all transitions modeling the migration of PMs.

Model outputs. A Markov reward approach is used to compute model outputs: a reward rate function is assigned at the SRN level and the expected reward rate at steady state is computed as the desired measures [1]. Our measures of interest are the following:

(i) Mean number of PMs in each pool. The mean number of non-failed PMs in the hot, warm, and cold pool is given by the mean number of tokens in the corresponding place P_h, P_w, and P_c (indicated as $E[\#P_h]$, $E[\#P_w]$, and $E[\#P_c]$ in the following). Reward assignments for these measures are shown in Table 5.

(ii) Availability of cloud service (A). We consider the cloud service to be available if the total number of PMs in hot pool is greater than or equal to k (with $1 \leq k \leq n_h$). The reward assignment for this measure is the one shown in Table 5.

5 Interacting SRN Sub-Models

To improve scalability, monolithic model is decomposed into three sub-models to describe the behavior of three pools. This section presents the interactions among the sub-models that facilitate the computation of same output measures as obtained from the monolithic model.

Figs. 2, 3, and 4, show the SRN sub-models for hot, warm, and cold pool respectively. Specific places and transitions of monolithic model appear in more than one sub-model. As a result, (i) guard functions $[g_1]$, $[g_2]$, and $[g_3]$ are absent in the interacting sub-models, (ii) transition rates are conveniently adapted for each sub-model, and (iii) sub-models exchange some of the input parameters and output measures to obtain the overall model solution.

Table 6 shows the rates of transitions that model the failure of PMs in interacting SRN sub-models. All other transition rates are same as in the monolithic model.

TABLE 4
Rates of Transitions Modeling the Migration of PMs in Monolithic SRN Model and Interacting SRN Sub-Models

Transitions	Rates of transitions
T_{whm}	$\#P_{whm} \cdot \gamma_{wh}$
T_{chm}	$\#P_{chm} \cdot \gamma_{ch}$
T_{cwm}	$\#P_{cwm} \cdot \gamma_{cw}$
T_{hcm}	$\#P_{hcm} \cdot \gamma_{hc}$
T_{hwm}	$\#P_{hwm} \cdot \gamma_{hw}$
T_{wcm}	$\#P_{wcm} \cdot \gamma_{wc}$

TABLE 5
Reward Rates to Compute Different Output Measures from 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 $\#P_h \geq k$; 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

SRN sub-model for hot pool. Fig. 2 shows the hot pool sub-model, which is derived from the structure of the monolithic model by keeping the transitions that directly interact with place P_h and disregarding the others. Sub-model input parameters are: (i) initial number of hot PMs (n_h), (ii) PM failure rate in hot pool (λ_h), (iii) repair rate (μ), (iv) number of repair facilities (n_r), (v) migration rates from warm and cold pool to hot pool (γ_{wh} and γ_{ch} respectively), and (vi) steady state probabilities that warm and cold pool have at least one PM available (p_h and p_w respectively). Parameters $\lambda_h, \mu, \gamma_{wh}$, and γ_{ch} are measured, n_h and n_r are design parameters, while p_w and p_c are obtained from the warm and cold pool sub-models respectively.

Next, we describe the intuition behind the modified transition rates shown in the hot PM sub-model with an example. Observe, the rate of transition T_{bwhf} is $\#P_h \cdot \lambda_h \cdot p_w$. While in the monolithic model an arc is present from place P_w to such a transition, hot sub-model does not have the place P_w. Thus, to capture the impact of the behavior of the warm pool sub-model on the throughput of transition T_{bwhf}, its original transition rate is scaled with p_w.

Similar approach is used to capture the impact of warm and cold pool on transitions T_{bchf} and T_{hf}. Transition rates of T_{bchf} and T_{hf} are $\#P_h \cdot \lambda_h \cdot (1-p_w) \cdot p_c$ and $\#P_h \cdot \lambda_h \cdot (1-p_w) \cdot (1-p_c)$ respectively.

Hot sub-model outputs. For place P_h, we obtain the average number of tokens $E[\#P_h]$, that represents mean number of available PMs in the hot pool. $E[\#P_h]$ is used to approximate the rate of transitions T_{bwhf} and T_{bchf} in the warm and cold pool SRN sub-models.

Probability (A) that at least k non-failed hot PMs are available, (i.e., $\#P_h \geq k$, with $1 \leq k \leq n_h$), is also obtained from hot pool sub-model. Overall cloud service availability is determined by the value of A.

Table 5 shows the reward rates to obtain these output measures.

SRN sub-model for warm pool. Fig. 3 shows the warm pool sub-model. Similar to the hot sub-model, warm pool sub-model structure is also derived from the monolithic model.

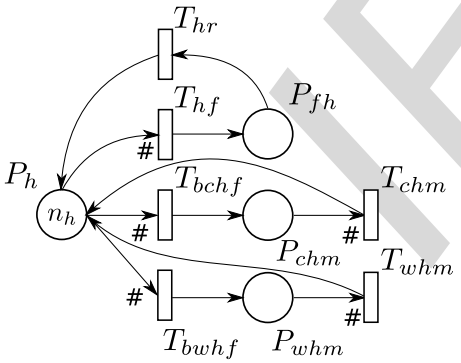

Fig. 2. Hot pool SRN sub-model.

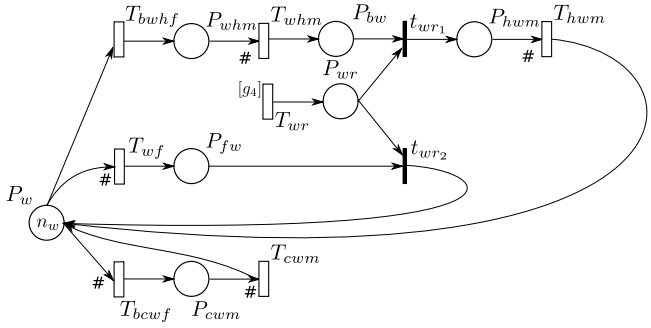

Fig. 3. Warm pool SRN sub-model.

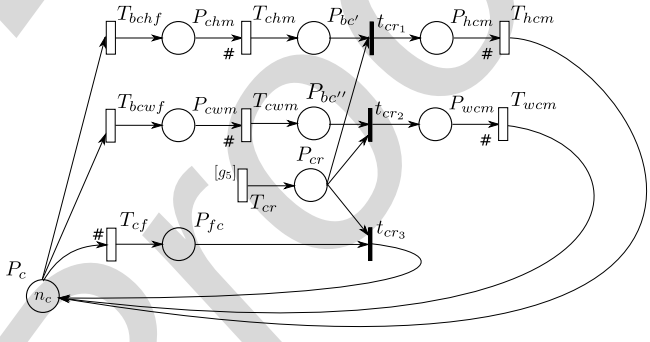

Fig. 4. Cold pool SRN sub-model.

TABLE 6
Rates of Transitions that Model the Failure of PMs in Interacting SRN Sub-Models

Transitions	Rates of transitions
T_{hf}	$\#P_h \cdot \lambda_h \cdot (1-p_w) \cdot (1-p_c)$
T_{bchf} (hot s-m)	$\#P_h \cdot \lambda_h \cdot (1-p_w) \cdot p_c$
T_{bchf} (cold s-m)	$\lambda_h \cdot (1-p_w) \cdot E[\#P_h]$
T_{bwhf} (hot s-m)	$\#P_h \cdot \lambda_h \cdot p_w$
T_{bwhf} (warm s-m)	$\lambda_h \cdot E[\#P_h]$
T_{wf}	$\#P_w \cdot \lambda_w \cdot (1-p_c)$
T_{bcwf} (warm s-m)	$\#P_w \cdot \lambda_w \cdot p_c$
T_{bcwf} (cold s-m)	$\lambda_w \cdot E[\#P_w]$
T_{cf}	$\#P_c \cdot \lambda_c$

Warm sub-model input parameters are: (i) initial number of warm PMs (n_w), (ii) PM failure rate in warm pool (λ_w), (iii) repair rate (μ), (iv) number of repair facilities (n_r), (v) migration rates from warm to hot pool (γ_{wh}), from hot to warm pool (γ_{hw}), and from cold to warm pool (γ_{cw}), (vi) mean number of available hot PMs ($E[\#P_h]$), and (vii) steady state probability that cold pool has at least one PM available (p_c). Parameters $\lambda_w, \mu, \gamma_{wh}, \gamma_{hw}$ and γ_{cw} are measured, while n_w and n_r are design parameters. $E[\#P_h]$ is obtained from the hot pool sub-model and p_c is obtained from the cold pool sub-model.

Rate of transitions T_{bcwf} and T_{wf} can be computed following the same approach used to the compute the rate of transitions $T_{bwhf}, T_{bchf}, T_{hf}$ in hot pool sub-model. Thus, rate of transition T_{bcwf} is $\#P_w \cdot \lambda_w \cdot p_c$ and rate of transition T_{wf} is $\#P_w \cdot \lambda_w \cdot (1 - p_c)$. Rate of transition T_{bwhf} (in warm pool) needs to be computed in a manner so that the throughput of T_{bwhf} is the same in warm and hot pool sub-model. Expected throughput of transition T_{bwhf} in the hot pool is given by:

$$\begin{aligned} Th_h(T_{bwhf}) &= \sum_{i=0}^{n_h} i \cdot \lambda_h \cdot p_w \cdot p(\#P_h = i) \\ &= \lambda_h \cdot p_w \cdot E[\#P_h], \end{aligned} \tag{1}$$

where $p(\#P_h = i)$ is the probability such that $\#P_h = i$. Assume, $rate_w(T_{bwhf})$ denotes the rate of transition T_{bwhf} in warm pool sub-model. Expected throughput of T_{bwhf} is given by:

$$\begin{aligned} Th_w(T_{bwhf}) &= \sum_{i=0}^{n_w} rate_w(T_{bwhf}) \cdot p(\#P_w = i) \\ &= rate_w(T_{bwhf}) \cdot p(\#P_w > 0) \\ &= rate_w(T_{bwhf}) \cdot p_w. \end{aligned} \tag{2}$$

Since $Th_w(T_{bwhf}) = Th_h(T_{bwhf})$, we can compute $rate_w(T_{bwhf})$ as

$$rate_w(T_{bwhf}) = \lambda_h \cdot E[\#P_h]. \tag{3}$$

Warm sub-model outputs. From warm pool sub-model, we compute: (i) probability (p_w) that at least one non-failed PM is available in the warm pool ($\#P_w \geq 1$), and (ii) mean number of available PMs in the warm pool ($E[\#P_w]$). As shown later, $E[\#P_w]$ will be used as an input parameter to the cold pool SRN sub-model, and p_w will be used as an input parameter to the hot and cold pool SRN sub-model.

Table 5 shows the reward rates assignment for warm pool output measures.

SRN sub-model for cold pool. Fig. 4 shows the structure of cold pool sub-model, which is derived from the monolithic model in a similar manner.

Cold sub-model input parameters are: (i) initial number of cold PMs (n_c), (ii) PM failure rate in cold pool (λ_c), (iii) repair rate (μ), (iv) number of repair facilities (n_r), (v) migration rate from cold to hot pool (γ_{ch}), from hot to cold pool (γ_{hc}), and from cold to warm pool (γ_{cw}), from warm to cold pool (γ_{wc}), (vi) mean number of hot PMs available in the hot pool ($E[\#P_h]$) and warm pool ($E[\#P_w]$), and (vii) steady state probability that warm pool has at least one PM available (p_w). Parameters $\lambda_c, \mu, \gamma_{ch}, \gamma_{hc}, \gamma_{cw}$ and γ_{cw} are measured, n_c and n_r are design parameters, $E[\#P_h]$ is obtained from the hot pool sub-model, $E[\#P_w]$ and p_w are obtained from the warm pool sub-model.

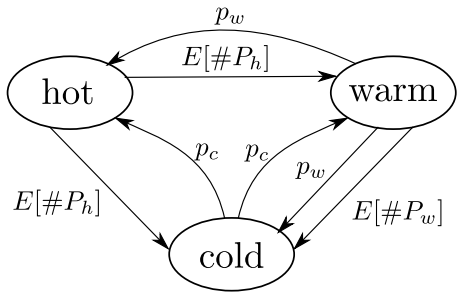

Fig. 5. Import graph describing sub-model interactions.

Rate of transitions T_{bchf} and T_{bcwf} in the cold pool sub-model are computed following the same approach as in case of warm pool:

$$rate_c(T_{bchf}) = \lambda_h \cdot (1 - p_w) \cdot E[\#P_h] \tag{4}$$

and

$$rate_c(T_{bcwf}) = \lambda_w \cdot E[\#P_w]. \tag{5}$$

Cold sub-model outputs. From cold pool sub-model, we compute: (i) probability (p_c) that at least one non-failed PM is available in the cold pool ($\#P_c \geq 1$), and (ii) mean number of available PMs in the cold pool ($E[\#P_c]$). In the hot and warm SRN sub-models, we use p_c as an input parameter to approximate the rate of transitions T_{bchf}, T_{hf}, T_{bcwf}, and T_{wf}.

Table 5 shows the reward assignments for cold sub-model output measures.

The above described sub-models and the interactions among them are shown as an import graph in Fig. 5. In the following, we briefly describe such interactions in terms of exchanged parameters.

Mean number of PMs in the hot pool ($E[\#P_h]$), as computed from hot pool sub-model, is needed as an input parameter to both the warm and cold pool sub-models.

Probability that the warm pool has at least one available PM (p_w) and the mean number of PMs in the warm pool ($E[\#P_w]$), are obtained as output measures from warm pool sub-model. While p_w is used both in the hot and cold pool sub-models, $E[\#P_w]$ is used in the cold pool sub-model.

Probability that the cold pool has at least one available PM (p_c), as computed from the cold pool sub-model, is used in the hot and warm pool sub-models.

Notice that there are cyclic dependencies among the sub-models as shown in Fig. 5. Fixed point iteration [23], [24], [25] is used to resolve such dependencies.

Overall model outputs. From the solution of interacting sub-models, we can obtain the same output measures of interest for the availability analysis of the cloud service that can be computed from the monolithic model. Specifically, 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. For a given k ($0 \leq k \leq n_h$), availability of cloud service can be computed from hot pool sub-model.

TABLE 7
Comparison of Number of States and Number of Non-Zero Entries between Monolithic and Interacting Sub-Models

n	Monolithic model		Interacting sub-models	
	#states	#non-zero entries	#states	#non-zero entries
3	10,272	59,560	196	588
4	67,075	453,970	491	1,768
5	334,948	2,526,920	1,100	4,518
6	1,371,436	11,220,964	2,262	10,272
7	4,816,252	41,980,324	3,770	18,434
8	Memory overflow	Memory overflow	6,939	36,316
10	-	-	20,460	118,710
20	-	-	21,273	106,300
40	-	-	271,543	1,481,000
60	-	-	1,270,813	7,148,100
80	-	-	3,859,083	22,051,600
100	-	-	9,196,353	53,055,500

n is the initial #PM in each pool.

6 Fixed Point Iteration

When using an interacting sub-models approach, fixed point iteration is used to solve the dependencies among the sub-models [23], [24].

We consider the fixed point equation

$$\mathbf{x} = \mathbf{G}(\mathbf{x}), \tag{6}$$

where $\mathbf{x} = (p_w, p_c, E[\#P_h], E[\#P_w])$, and variables p_w, p_c, $E[\#P_h]$, $E[\#P_w]$ are the ones illustrated in Fig. 5. Since it is possible to find functions f_1, f_2, and f_3 such that

(i) $E[\#P_h] = f_1(p_w, p_c)$, and

(ii) $E[\#P_w] = f_2(E[\#P_h], p_c) = f_3(p_w, p_c)$,

all the variables of the model can be expressed as functions of p_w and p_c. Hence, we change Equation (6) into:

$$\mathbf{y} = \mathbf{F}(\mathbf{y}), \tag{7}$$

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

To demonstrate the existence of a solution to Equation (7)—which implies the existence of a solution to Equation (6)—we use the Brouwer's theorem [26]:

Let $\mathbf{F} : \mathcal{C} \subset \mathbb{R}^2 \to \mathbb{R}^2$ be continuous on the compact, convex set $\mathcal{C}$, and suppose that $\mathbf{F}(\mathcal{C}) \subseteq \mathcal{C}$. Then, $\mathbf{F}$ has a fixed point in $\mathcal{C}$.

Let us recall that: (i) if a subset of the euclidean space $\mathbb{R}^n$ is closed and bounded then it is also compact (Heine-Borel theorem); (ii) a set $\mathcal{C}$ is convex if, for all $x \in \mathcal{C}$, $y \in \mathcal{C}$, and $t \in [0,1]$, $tx + (1-t)y \in \mathcal{C}$; (iii) a vector function $\mathbf{F}$ is continuous over the set $\mathcal{C}$ if its component functions $F_1(\mathbf{y})$ and $F_2(\mathbf{y})$ are continuous over $\mathcal{C}$, for each $\mathbf{y} \in \mathcal{C}$; (iv) the functions F_1 and F_2 are continuous over $\mathcal{C}$ if, for each $\mathbf{y} \in \mathcal{C}$ and for each $\hat{\mathbf{y}} \in \mathcal{C}$, $lim_{\mathbf{y}\to\hat{\mathbf{y}}} F_i(\mathbf{y}) = F_i(\hat{\mathbf{y}})$, $i = 1, 2$.

Since p_w and p_c belong to the closed and bounded interval $[0,1]$ (they are probabilities), define $\mathcal{C} = \{\mathbf{y} = (p_w, p_c) : p_w \in [0,1],\ p_c \in [0,1]\}$, such that $\mathcal{C}$ is compact and convex. Moreover, notice that, $\mathcal{C}$ is a singleton set of the only element $\mathbf{y} = <p_w, p_c>$ for which $lim_{\mathbf{y}\to\hat{\mathbf{y}}} F_i(\mathbf{y}) = F_i(\hat{\mathbf{y}})$, $i = 1, 2$. Hence, $\mathbf{F}$ is continuous over $\mathcal{C}$.

This demonstrates the existence of a solution.

We also note that the solution is unique even if the initial guess of the solution is changed and that the maximum number of required iterations is 3 for all the investigated scenarios. Hence, the fixed-point iteration converges reasonably fast.

7 Solving the Models

We use the SPNP software package [7] to solve the SRN models. The interacting SRN sub-models are solved by implementing a fixed point iteration approach using Python scripts. Using SPNP, simulative solution of the monolithic SRN model is also obtained. In fact, SPNP provides a nice paradigm to obtain analytic-numeric as well as simulative solution from the same SRN model.

Our models were solved for a broad range of parameter values so that they can represent a large variety of clouds. However, we report some interesting results in this paper. We assume MTTF of hot PMs to be in the range of 100-500 hrs, MTTF of warm PMs to be in the range of 300-1,750 hrs and MTTF of cold PMs to be in the range of 500-2,500 hrs. 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). MTTMs of a PM are assumed to vary between 10-60 minutes.

All models were solved using a desktop PC with 3.0 GHz CPU and 4 GB memory. Hence, reported figures are relative to this machine, but we expect similar trends also using other computers.

8 Comparison between Monolithic Model and Interacting Sub-Models

In Table 7, we report the state space size and storage requirements for both the monolithic model and the interacting sub-models. Monolithic model runs into a memory overflow problem when the number of PMs in each pool increases beyond 7. We observe that the number of states and the number of non-zero entries in the generator matrix

TABLE 8
Comparison of Analytic-Numeric Solution Times (In Seconds) between Monolithic and Interacting Sub-Models

n	Monolithic model	Interacting sub-models
3	0.416	0.545
4	2.339	0.513
5	13.741	0.568
6	71.638	0.614
7	301.700	0.584
8	Mem. overflow	1.515
10	-	0.873
20	-	0.783
40	-	5.041
60	-	25.660
80	-	100.034
100	-	364.025

n is the initial #PM in each pool.

of the underlying CTMC of the monolithic model increase quickly and become too large to construct the reachability graph even for small number of PMs. For the same number of PMs, with interacting sub-models approach, the number of states and the non-zero entries are several orders of magnitude smaller compared to the monolithic model. Such reduction in the state space and the number of non-zero entries also leads to concomitant reduction in solution time needed. The memory overflow problem for the interacting sub-models approach, instead, appears when increasing the number of machines per pool beyond 100.

A comparison of solution times (in seconds) is shown in Table 8. Solution time for monolithic model increases nearly exponentially with the increase in model size. Solution time for interacting sub-models increases very slowly with the increase in model size.

TABLE 9
Comparison of Downtime Values (In Minutes per Year) between Monolithic and Interacting Sub-Models

n	k	Monolithic m.	Interacting s-m.
7	7	3,664.527	3,664.527
	6	10.978	10.978
	5	0.018	0.018
6	6	3,142.591	3,142.591
	5	7.847	7.847
	4	0.010	0.010
5	5	2,620.134	2,620.134
	4	5.235	5.235
	3	0.005	0.005
4	4	2,097.154	2,097.154
	3	3.143	3.143
	2	0.002	0.002
3	3	1,573.651	1,573.651
	2	1.572	1.572
	1	0.0005	0.0005

n is the initial #PM in each pool. k is the #PM in hot pool to have the cloud available.

TABLE 10
Comparison of Mean #PMs in Each Pool between Monolithic and Interacting Sub-Models

n	Monolithic m.			Interacting s-m.		
	hot	warm	cold	hot	warm	cold
3	2.99	2.97	2.98	2.99	2.97	2.98
4	3.99	3.96	3.98	3.99	3.96	3.98
5	4.99	4.95	4.98	4.99	4.95	4.98
6	5.99	5.94	5.97	5.99	5.94	5.97
7	6.99	6.94	6.97	6.99	6.94	6.97

n is the initial #PMs in each pool.

In Table 9, we compare the downtime values (in minutes per year) as obtained from the monolithic model and interacting sub-models. We assume that cloud is available if there are at least k non-failed PMs in hot pool. For different initial number of PMs per pool (n), we vary the value of k and report the results. MTTFs of hot, warm and cold PMs were assumed to be 500, 1,750 and 2,500 hrs respectively. MTTR was assumed to be 3 hours. MTTM values across the pools were assumed to be 30 minutes. Table 9 shows that the results obtained from the interacting sub-models are accurate. The results differ only after the 8th significant figure, and hence they are not reported in Table 9. As expected, downtime values are higher with increasing values of k.

In Table 10, we show the mean number of non-failed PMs in each pool. We used the same set of MTTF, MTTR and MTTM values that were used to generate the results in Table 9. Results obtained from interacting sub-models are in good agreement with the results obtained from monolithic model and the difference appears only after the 8th significant figure.

9 Comparison between Simulation and Analytic-Numeric Solution

Even though the interacting sub-models approach slows down the growth of the number of states in the underlying Markov model (discussed in [2]), in a more realistic model when the number of PMs per pool is larger than 100, model solution time increases and the memory overflow issue may arise in not well equipped machines. Hence, we also consider the solution by means of simulation of the monolithic model, which avoids the generation of the underlying Markov model by directly solving the stochastic net. We use the method of *batch means*, where a single long simulation run is generated till we are sure that steady-state is reached. This single run is decomposed into several batches, which can be considered nearly independent, so that statistical analysis can be performed on them [1]. Compared to many independent simulation runs (method of *independent replications*), key advantage of the method of batch means is the reduction of unproductive portion of simulation time to just one initial stabilization period.

We compare the values of cloud service downtime as obtained from the analytic-numeric solution of the monolithic model, analytic-numeric solution of the interacting sub-models, and simulation of the monolithic model as well as the solution times when using the three approaches.

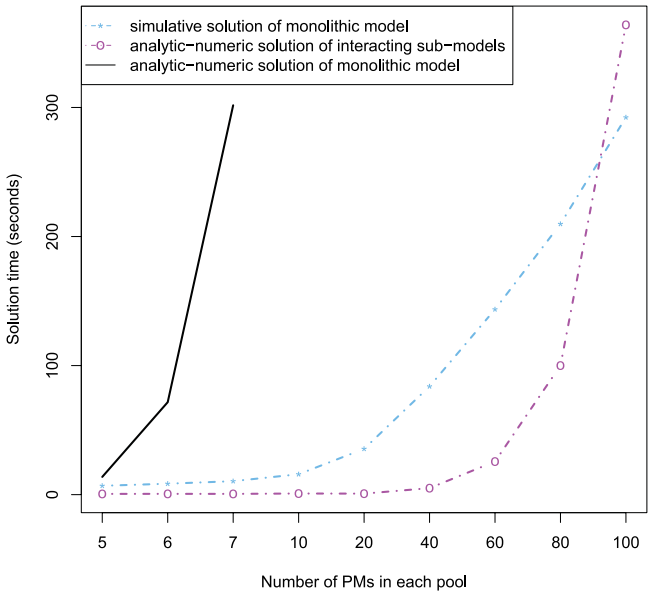

Fig. 6. Comparison of solution times among monolithic model simulative solution, monolithic model analytic-numeric solution, and analytic-numeric solution of the interacting sub-models.

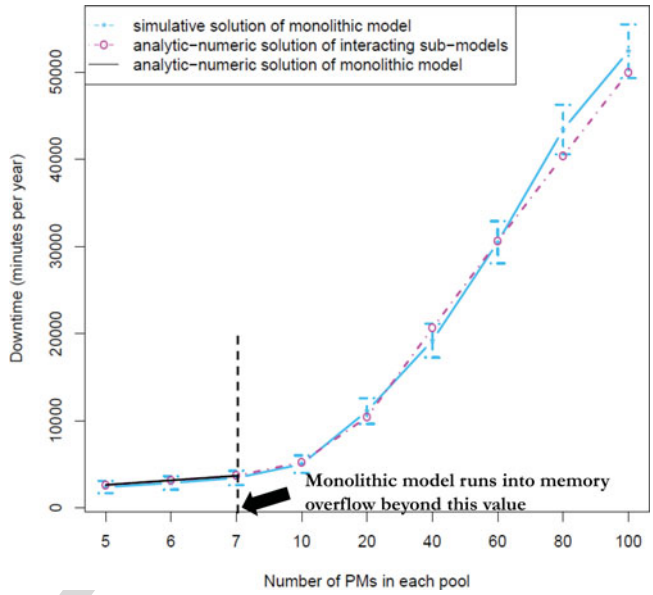

Fig. 7. Comparison of cloud service downtime among monolithic model simulative solution, monolithic model analytic-numeric solution, and analytic-numeric solution of the interacting sub-models.

Simulation results are reported with 95 percent confidence interval and the size of each batch is assumed to be 10,000 hours. Half-width relative error is 10 percent when the number of PMs per pool is more than 40. When the number of PMs per pool varies between 5 to 40, half-width relative error varies between 10-30 percent. To achieve lower half-width relative error with fewer PMs per pool, time required for simulative solution of monolithic model is significantly (order of hours) higher even compared to the time required for analytic-numeric solution of the monolithic model.

Solution times of the three approaches are compared in Fig. 6. Up to 7 PMs per pool, time for analytic-numeric solution of the monolithic model is the largest among the three approaches. For the example scenarios investigated, when the number of PMs per pool is less than 100, analytic-numeric solution times for interacting sub-models are less compared to simulation solution time. However, for larger systems, simulation results are obtained faster compared to the interacting sub-models. This is because, generation of reachability graph is not needed in simulation.

Fig. 7 shows the comparison of downtime values achieved using the three approaches, when the number of PMs in each pool are varied. We observe that up to 7 PMs per pool the simulation results match closely with both analytic-numeric approaches (monolithic and interacting sub-models). Analytic-numeric solution of the monolithic model runs into a memory overflow problem beyond 7 PMs per pool and hence, can not produce any further result. Simulative solution of the monolithic model is also in good agreement with the result obtained from the interacting sub-models beyond 7 PMs per pool.

In Table 11, we show the mean number of PMs in each pool for analytic-numeric solution of the interacting sub-models and simulative solution of the monolithic model. Simulation results are reported with 95 percent confidence interval and 10 percent half-width relative error. We can observe that computed numbers of both approaches are very close. Almost all results achieved by means of analytic-numeric solution of the interacting sub-models are in the corresponding confidence intervals achieved when simulating the monolithic model.

TABLE 11
Comparison of Mean Number of Non-Failed PMs in Each Pool among Analytic-Numeric Solution of the Interacting Sub-Models and Simulative Solution of the Monolithic Model

n	Interacting sub-models			Simulation		
	hot	warm	cold	hot	warm	cold
5	4.9950	4.9576	4.9824	4.9955, [4.9942, 4.9968]	4.9650, [4.9613, 4.9687]	4.9868, [4.9845, 4.9891]
10	9.9900	9.9134	9.9645	9.9905, [9.9886, 9.9924]	9.9269, [9.9215, 9.9323]	9.9687, [9.9652, 9.9722]
20	19.980	19.818	19.927	19.979, [19.976, 19.982]	19.823, [19.815, 19.832]	19.933, [19.928, 19.938]
40	39.960	39.593	39.846	39.963, [39.959, 39.966]	39.607, [39.593, 39.621]	39.849, [39.841, 39.857]
60	59.940	59.301	59.756	59.939, [59.935, 59.944]	59.298, [59.278, 59.317]	59.758, [59.748, 59.769]
80	79.920	78.896	79.655	79.910, [79.904, 79.916]	78.913, [79.886, 78.940]	79.656, [79.643, 79.669]
100	99.900	98.275	99.540	99.891, [99.884, 99.897]	98.297, [98.258, 99.335]	99.534, [99.519, 99.550]

n is the initial #PMs in each pool.

TABLE 12
Summary of Comparison among the Three Approaches: (i) Interacting Sub-Models, (ii) Simulation, and (iii) Commonly Used Single Monolithic Model

Comparison index	Metric	Configuration	Monolithic model	Interacting sub-models	Simulation
Accuracy	Downtime (sec)	7 PMs per pool	3664.527	3664.527	3416.4
		100 PMs per pool	-	49993	52402
Scalability	Max #PMs per pool	-	≤ 7	> 100	> 200
	#states	7 PMs per pool	$4,816,252$	$3,770$	-
	#states	100 PMs per pool	-	$9,196,353$	-
Efficiency	Solution time (sec)	7 PMs per pool	301.700	0.584	10.434
		100 PMs per pool	-	364.025	294.346

As a result, for small cloud instances, analytic-numeric solutions can give accurate results within reasonable time. On the other hand, simulation should be preferred for larger systems, where storage requirements as well as solution times are much less compared to analytic-numeric solution of the models.

Finally, Table 12 summarizes the comparison among the three approaches: (i) interacting sub-models, (ii) simulation, and (iii) commonly used single monolithic model, in terms of accuracy, scalability, and efficiency. While the interacting sub-models approach is accurate w.r.t. the monolithic model, the errors introduced by the simulation is 4.8 percent when considering 100 PMs per pool. However, when the system has more than 100 PMs, simulation also appears to be more scalable than the decomposition.In fact, unlike the results shown in [2], the number of states of the underlying Markov chain in the interacting sub-models easily reaches 9 million, making this approach infeasible for very large cloud systems. By contrast, simulation does not require to build the underlying state-space model and allows us to solve very large cloud models with 19.2 percent reduction in solution time.

10 Sub-model Closed Form Solution

Closed-form solution of the sub-models can be derived when the migration process is instantaneous (i.e., migration delay is zero). This can be obtained by substituting transitions T_{whm}, T_{chm}, T_{hcm}, T_{hwm}, T_{cwm}, and T_{wcm} with immediate transitions. Underlying Markov chains of such modified sub-models are used to obtain the closed form solutions. We show results in the case of $n_r = 1$, but closed form results can also be derived for the cases where $n_r > 1$ [1].

Closed form solution for the hot pool. The Markov chain underlying the hot pool SRN sub-model with no migration delay and $n_r = 1$ is shown in Fig. 8 (rates associated to self-loops are not reported). State i models the configuration in which i PMs are available in the hot pool. Self-loops can be ignored during solution of such a continuous time Markov chain [1] and, as a consequence, the Markov chain represents a simple birth-death process. Birth rate for state i is $\lambda_h \cdot (1 - p_w) \cdot (1 - p_c) \cdot i$ while death rate for all states is μ. Let

$$\lambda'_h = \lambda_h \cdot (1 - p_w) \cdot (1 - p_c) \tag{8}$$

and p_{h_i} be the steady state probability to have i PMs in the hot pool, i.e., the probability to be in state i of the Markov chain. p_{h_i} is given by

$$p_{h_i} = \frac{\lambda_h'^{(n_h - i)}}{\mu^{(n_h - i)}} \cdot \frac{(n_h)!}{i!} \cdot p_{h_{n_h}} \text{ with } (0 \leq i \leq n_h - 1) \tag{9}$$

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

$$p_{h_{n_h}} = \frac{1}{\sum_{i=0}^{n_h} \frac{\lambda_h'^{(n_h - i)}}{\mu^{(n_h - i)}} \cdot \frac{(n_h)!}{i!}}. \tag{10}$$

Starting from such steady-state probabilities, the mean number of PMs in the hot pool ($E[\#P_h]$) that needs to be exchanged with the other sub-models can be computed as follows:

$$\begin{aligned} E[\#P_h] &= \sum_{i=0}^{n_h} i \cdot p_{h_i} \\ &= \sum_{i=0}^{n_h} i \cdot \frac{\lambda_h'^{(n_h - i)}}{\mu^{(n_h - i)}} \cdot \frac{(n_h)!}{i!} \cdot \frac{1}{\sum_{j=0}^{n_h} \frac{\lambda_h'^{(n_h - j)}}{\mu^{(n_h - j)}} \cdot \frac{(n_h)!}{j!}}. \end{aligned} \tag{11}$$

The availability of the cloud is given by:

$$A = \sum_{i=k}^{n_h} p_{h_i}. \tag{12}$$

Closed form solution for the warm pool. The Markov chain underlying the warm pool SRN sub-model with no

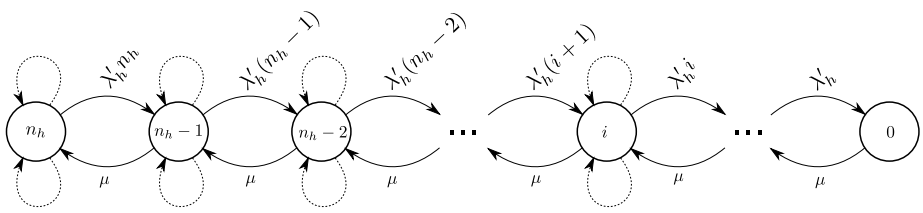

Fig. 8. Markov chain for the hot pool when migration delay is zero.

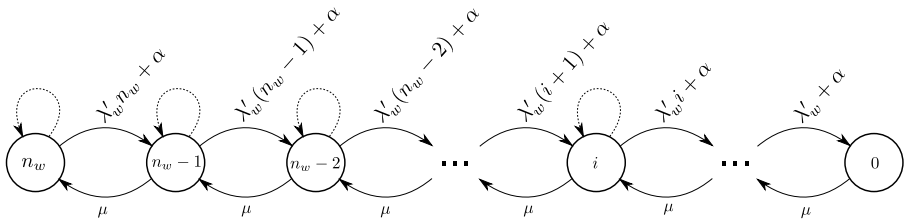

Fig. 9. Markov chain for the warm pool when migration delay is zero.

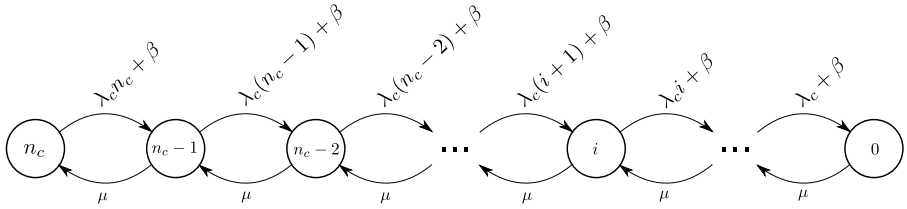

Fig. 10. Markov chain for the cold pool when migration delay is zero.

migration delay and $n_r = 1$ is shown in Fig. 9 (rates associated to self-loops are not reported). Also in this case, the Markov chain collapses to a simple birth-death process. Birth rate for state i is $\lambda_w \cdot (1-p_c)i + \lambda_h \cdot E[\#P_h]$ while death rate for all states is μ. Let us define:

$$\lambda'_w = \lambda_w \cdot (1-p_c) \tag{13}$$

and

$$\alpha = \lambda_h \cdot E[\#P_h] \tag{14}$$

and let p_{w_i} be the steady state probability to have i PMs in the warm pool, i.e., the probability to be in state i of the Markov chain. p_{w_i} is given by:

$$p_{w_i} = \prod_{j=i}^{n_w-1} \frac{\lambda'_w \cdot (j+1)+\alpha}{\mu} \cdot p_{w_{n_w}} \text{ with } (0 \le i \le n_w - 1) \tag{15}$$

while $p_{w_{n_w}}$ is given by:

$$p_{w_{n_w}} = \frac{1}{1+\sum_{i=0}^{n_w-1}\prod_{j=i}^{n_w-1}\frac{\lambda'_w\cdot(j+1)+\alpha}{\mu}}. \tag{16}$$

Starting from such steady-state state probabilities, the values of p_w and $E[\#P_w]$ can be computed as

$$p_w = 1 - \prod_{j=0}^{n_w-1} \frac{\lambda'_w \cdot (j+1)+\alpha}{\mu} \cdot p_{w_{n_w}} \tag{17}$$

$$E[\#P_w] = \sum_{i=0}^{n_w} i \cdot p_{w_i}. \tag{18}$$

Closed form solution for the cold pool. The Markov chain underlying the cold pool SRN sub-model with no migration delay and $n_r = 1$ is shown in Fig. 10 (rates associated to self-loops are not reported). In this case, birth rate for state i is $\lambda_c i + \lambda_h \cdot (1-p_w) \cdot E[\#P_h] + \lambda_w \cdot E[\#P_w]$ while all death rates are μ. Define β as

$$\beta = \lambda_h \cdot (1-p_w) \cdot E[\#P_h] + \lambda_w \cdot E[\#P_w]. \tag{19}$$

Let p_{c_i} be the steady state the probability to have i PMs in the cold pool, i.e., the probability to be in state i of the Markov chain. p_{c_i} is given by:

$$p_{c_i} = \prod_{j=i}^{n_c-1} \frac{\lambda_c \cdot (j+1)+\beta}{\mu} \cdot p_{c_{n_c}} \text{ with } (0 \le i \le n_c - 1) \tag{20}$$

while $p_{c_{n_c}}$ is given by:

$$p_{c_{n_c}} = \frac{1}{1+\sum_{i=0}^{n_c-1}\prod_{j=i}^{n_c-1}\frac{\lambda_c(j+1)+\beta}{\mu}}. \tag{21}$$

Starting from such steady-state state probabilities, we can compute p_c and $E[\#P_c]$ as

$$p_c = 1 - \prod_{j=0}^{n_c-1} \frac{\lambda_c(j+1)+\beta}{\mu} \cdot p_{c_{n_c}} \tag{22}$$

$$E[\#P_c] = \sum_{i=0}^{n_c} i \cdot p_{c_i}. \tag{23}$$

Table 13 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.

TABLE 13
Solution Time (in Seconds) Required for Availability Analysis of Large Scale Cloud Using Closed-Form

n	Solution time
1000	0.106
2000	0.131
3000	0.158
4000	0.230
5000	0.268

n is the initial #PM in each pool.

11 Conclusions

This paper describes a stochastic modeling approach for availability analysis of large IaaS cloud systems. We show how scalability issues for a monolithic model can be resolved by means of interacting sub-models or by means of simulation. The interacting sub-models approach quickly provides model solutions facilitating scalability without significantly compromising the accuracy. Simulation provides results that closely match with monolithic and interacting sub-models approaches and, for large systems, results are obtained faster. In special cases, closed form solutions can also be derived to solve very large cloud models quickly. Cloud service providers can benefit from the proposed modeling approach during design, development, testing and operation of IaaS cloud. During design and development, providers can use these models to determine the pool size required to offer a specific availability SLA. In the testing and operational stages, the providers can tune parameters for the dynamic repair strategies (e.g., number of parallel repairs, automated versus manual repairs) to maintain the promised availability SLA.

Acknowledgments

This work has been partially supported by the TENACE PRIN Project (no. 20103P34XC) funded by the Italian Ministry of Education, University and Research. K.S. Trivedi's research was funded in part by an IBM Faculty Award.

References

[1] K.S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. second ed., John Wiley & Sons, 2001.

[2] F. Longo, R. Ghosh, V.K. Naik, and K.S. Trivedi, "A Scalable Availability model for Infrastructure-as-a-Service Cloud," *Proc. Int'l Conf. Dependable Systems and Networks*, pp. 335-346, 2011.

[3] G. Ciardo et al., "Automated Generation and Analysis of Markov Reward Models Using Stochastic Reward Nets," *Mathematics and Its Applications: Linear Algebra, Markov Chains and Queueing Models*, vol. 48, pp. 145-191, Springer, 1993.

[4] K.S. Trivedi, R. Vasireddy, D. Trindade, S. Nathan, and R. Castro, "Modeling High Availability System," *Proc. 12th Pacific Rim Int'l Symp. Dependable Computing*, 2006.

[5] D. Kim, F. Machida, and K.S. Trivedi, "Availability Modeling and Analysis of a Virtualized System," *Proc. 15th IEEE Pacific Rim Int'l Symp. Dependable Computing*, 2009.

[6] K.S. Trivedi and R. Sahner, "SHARPE at the Age of Twenty Two," *ACM SIGMETRICS Perf. Eval. Rev.*, vol. 36, no. 4, pp. 52-57, 2009.

[7] C. Hirel, B. Tuffin, and K.S. Trivedi, "SPNP: Stochastic Petri Nets. Version 6.0," *Proc. 11th Int'l Conf. Computer Performance Evaluation: Modelling Techniques and Tools*, vol. 1768, 2000.

[8] R. Sahner, K.S. Trivedi, and A. Puliafito, *Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software*. Kluwer Academic Publishers, 1996.

[9] J. Blake and K.S. Trivedi, "Reliability Analysis of Interconnection Networks Using Hierarchical Composition," *IEEE Trans. Reliability*, vol. 38, no. 1, pp. 111-120, Apr. 1989.

[10] B. Tuffin, P.K. Choudhary, C. Hirel, and K.S. Trivedi, "Simulation versus Analytic-Numeric Methods: Illustrative Examples," *Proc. Second Int'l Conf. Performance Evaluation Methodologies and Tools (ValueTools '07)*, article 63, 2007.

[11] B. Yang, F. Tan, and Y.-S. Dai, "Performance Evaluation of Cloud Service Considering Fault Recovery," *The J. Supercomputing*, vol. 65, no. 1, pp. 426-444, 2013.

[12] T. Uemura, T. Dohi, and N. Kaio, "Availability Analysis of a Scalable Intrusion Tolerant Architecture with Two Detection Modes," *Proc. Int'l Conf. Cloud Computing (CloudCom)*, 2009.

[13] S. He, L. Guo, M. Ghanem, and Y. Guo, "Improving Resource Utilisation in the Cloud Environment Using Multivariate Probabilistic Models," *Proc. IEEE Fifth Int'l Conf. Cloud Computing (CLOUD)*, 2012.

[14] K. Vishwanath and N. Nagappan, "Characterizing Cloud Computing Hardware Reliability," *Proc. First ACM Symp. Cloud Computing (SOCC)*, 2010.

[15] N. Bonvin, T.G. Papaioannou, and K. Aberer, "Dynamic Cost Efficient Replications in Data Clouds," *Proc. ACM Workshop Automated Control for Datacenters and Clouds*, pp. 49-56, 2009.

[16] G. Callou et al., "Sustainability and Dependability Evaluation on Data Center Architectures," *Proc. IEEE Int'l Conf. Systems, Man, and Cybernetics*, pp. 398-403, 2011.

[17] Y. Tan, X. Gu, and H. Wang, "Adaptive System Anomaly Prediction for Large-Scale Hosting Infrastructures," *Proc. 29th ACM SIGACT-SIGOPS Symp. Principles of Distributed Computing (PODC)*, 2010.

[18] B. Javadi et al., "Discovering Statistical Models of Availability in Large Distributed Systems: An Empirical Study of Seti@home," *IEEE Trans. Parallel and Distributed Systems*, vol. 22, no. 11, pp. 1896-1903, Nov. 2011.

[19] H. Chen et al., "Petri Net Modeling of the Reconfigurable Protocol Stack for Cloud Computing Based Control Systems," *Proc. IEEE Second Int'l Conf. Cloud Computing Technology and Science (CloudCom)*, 2010.

[20] Y.-S. Dai et al., "Cloud Service Reliability: Modeling and Analysis," *Proc. IEEE Pacific Rim Int'l Symp. Dependable Computing (PRDC)*, 2009.

[21] M. Lanus, L. Yin, and K.S. Trivedi, "Hierarchical Composition and Aggregation of State-Based Availability and Performability Models," *IEEE Trans. Reliability*, vol. 52, no. 1, pp. 44-52, Mar. 2003.

[22] D. Wang, R.M. Fricks, and K.S. Trivedi, "Dealing with Non-Exponential Distributions in Dependability Models," G. Kotsis ed., *Performance Evaluation - Stories and Perspectives*. Oesterreichische Computer Gesellschaft, 2003.

[23] V. Mainkar and K.S. Trivedi, "Sufficient Conditions for Existence of a Fixed Point in Stochastic Reward Net-Based Iterative Models," *IEEE Trans. Software Eng.*, vol. 22, no. 9, pp. 640-653, Sept. 1996.

[24] L. Tomek and K.S. Trivedi, "Fixed-Point Iteration in Availability Modeling," *Proc. Fifth Int'l GI/ITG/GMA Conf. Fault-Tolerant Computing Systems, Tests, Diagnosis, Fault Treatment*, vol. 91, pp. 229-240, 1991.

[25] G. Ciardo and K.S. Trivedi, "A Decomposition Approach for Stochastic Reward Net Models," *Performance Evaluation*, vol. 18, pp. 37-59, 1993.

[26] J.M. Ortega and W.C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, 1970.

Rahul Ghosh received the PhD degree in electrical and computer engineering from Duke University in 2012. During his PhD, he also worked as a research intern at IBM T.J. Watson Research Center. He is an advisory software engineer at IBM. His research interests include stochastic processes, queuing systems, Markov chains, and performance and dependability analysis of large scale computer systems.

Francesco Longo received the PhD degree in advanced technologies for information engineering at the University of Messina, Italy, in 2011. He is currently a postdoc researcher within the CloudWave European project. His main research interests include performance and reliability evaluation of distributed systems (in particular Grid and Cloud) with main attention to non-Markovian aspects.

Flavio Frattini is currently working toward the PhD degree in computer and automation engineering in the Department of Electrical Engineering and Information Technology at the Federico II University of Naples. His interests include distributed systems, performance, dependability and consumption analysis, and modeling. He was Research Fellow at the Italian National Research Council (CNR).

Stefano Russo is a professor of computer engineering at the Federico II University of Naples, teaching software engineering and distributed systems, and leading the Distributed and Mobile Systems Research Group. He coauthored more than 130 papers in the areas of distributed software engineering, middleware technologies, software dependability, and mobile computing. He is an associate editor of the *IEEE Transactions on Services Computing*.

Kishor S. Trivedi holds the Hudson chair in the Department of Electrical and Computer Engineering at Duke University. He is the author of *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, published by John Wiley. He has published more than 500 articles and has supervised 45 PhD dissertations. He received the IEEE Computer Society Technical Achievement Award for his research on Software Aging and Rejuvenation.