Performance and Power Modeling and Evaluation of Virtualized Servers in IaaS Clouds

Reza Entezari-Maleki[a,*], Leonel Sousa[b], Ali Movaghar[c]

[a]*School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran*
[b]*INESC-ID, Instituto Superior Tecnico, Universidade de Lisboa, Lisbon, Portugal*
[c]*Department of Computer Engineering, Sharif University of Technology, Tehran, Iran*

Abstract

In this paper, Stochastic Activity Networks (SANs) are exploited to model and evaluate the power consumption and performance of virtualized servers in cloud computing. The proposed SAN models the physical servers in three different power consumption and provisioning delay modes, switching the status of the servers according to the workload of the corresponding cluster if required. The Dynamic Voltage and Frequency Scaling (DVFS) technique is considered in the proposed model for dynamically controlling the supply voltage and clock frequency of CPUs. Thus, Virtual Machines (VMs) on top a physical server can be divided into several power consumption and processing speed groups. According to the workload of the system and the number of waiting requests, the proposed SAN decides to scale up or down the VMs, so it helps the overall system to save power when it still preserves satisfiable performance. After modeling the servers and VMs using SAN formalism, some performance related measures together with the power consumption metric are defined on the proposed model. The results obtained by solving the proposed SAN model configured with real data show the prominence of the proposed model in comparison with some baselines and previously proposed models.

Keywords: Cloud computing, virtualization, power consumption, performance modeling, stochastic activity network.

Acronyms

CPN	Colored Petri Net
CTMDP	Continuous-Time Markov Decision Process
DVFS	Dynamic Voltage and Frequency Scaling
GSPN	Generalized Stochastic Petri Net
IaaS	Infrastructure-as-a-Service

*School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran.
Email addresses: entezari@ipm.ir (Reza Entezari-Maleki), Leonel.Sousa@inesc-id.pt (Leonel Sousa), movaghar@sharif.edu (Ali Movaghar)

MC	Markov Chain
MDP	Markov Decision Process
MDPN	Markov Decision Petri Net
MDWN	Markov Decision Well-formed Net
MMPP	Markov Modulated Poisson Process
MRM	Markov Reward Model
PaaS	Platform-as-a-Service
PN	Petri Net
QoS	Quality of Service
SaaS	Software-as-a-Service
SAN	Stochastic Activity Network
SLA	Service Level Agreement
SPN	Stochastic Petri Net
SRN	Stochastic Reward Net
SSDN	Stochastic Service Decision Net
VM	Virtual Machines
VMM	Virtual Machine Monitor

1. Introduction

Cloud Computing has attracted increasing attention from both researchers and practitioners as a new paradigm of information technology, which principles have numerous applications [12, 36]. According to the definition by National Institute of Standards and Technology (NIST), "cloud computing is a model for enabling convenient, and on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [4]. Cloud computing includes three major delivery models: Software-as-a-Service (SaaS) in which the consumer is able to use an application to meet specific needs, Platform-as-a-Service (PaaS) which provides the consumer with a hosting environment for application development, and Infrastructure-as-a-Service (IaaS) in which the consumer has a greater access to computing resources including processing power, storage, networking components and middleware [4, 36]. In this paper, we focus on IaaS clouds. It is essential in all kinds of delivery models that clients have guarantees from providers on service delivery. Typically, these guarantees are provided by Service Level Agreements (SLAs) between cloud service providers and clients [5, 12, 34]. The SLAs contain a number of good measurable objectives such as power consumption, delivered performance, storage space, availability, security, and the penalties for the supplier for every unit of order that he is unable to satisfy.

Recently, virtualization has enabled the abstraction of computing resources such that a physical server is able to function as a set of multiple logical Virtual Machines (VMs) [7, 12, 30]. Most of the modern cloud data centers are equipped with virtualized clusters, running hypervisors on virtualization-supported hardware. Virtualization in cloud data centers allows performance isolation, optimization of power consumption, fault tolerance

and improved system management with seamless VM [7]. Alongside with virtualization, power consumption is utmost important in modern data center and enterprise environments since it directly impacts both the deployment (peak power delivery capacity) and operational costs (power supply and cooling) [13, 16]. In 2013, US data centers consumed an estimated 91 billion kilowatt-hours of electricity, equivalent to the annual output of 34 large (500 megawatt) coal-fired power plants [3]. Data center electricity consumption is projected to increase to roughly 140 billion kilowatt-hours annually by 2020, the equivalent annual output of 50 power plants, costing American businesses $13 billion annually in electricity bills and emitting nearly 100 million metric tons of carbon pollution per year [3]. Therefore, among the main challenges currently faced by most of data centers is the optimization of power consumption of the virtualized data centers. Mechanisms such as grouping servers of a data center into several power consumption pools, and moving servers among these pools if it is required, may be applied [9, 10, 20, 21]. Another managing mechanism to reduce the power consumption of a virtualized server is to adapt the server speed to the workload. By applying Dynamic Voltage and Frequency Scaling (DVFS), unlike shutting down or sleeping idle servers, a server runs at different factors of the peak service rate by scaling up or down the processing speed of its CPUs [14, 33, 34]. DVFS is a commonly used technique to save power on a wide range of computing systems from embedded, laptop and desktop systems to high performance server-class systems [32].

In IaaS clouds, which provide users with virtualized computing resources over the network, the need for methods to accurately evaluate the power consumption of the system when different energy saving methods are applied is of utmost importance. In addition to assess the power consumption of the system, a good modeling technique should be able to evaluate the performance delivered to end-users to decide whether the SLA conditions of hosted user are met or not. There are various ways to achieve this evaluation, namely measurement-based evaluation, modeling with simulation, and analytical modeling [9, 10, 19]. Measurement-based evaluation on cloud systems, and generally, on any complex and highly distributed system, needs an extensive experimentation with different workloads and system configurations, which may not be feasible due to the large network size, and time and budget limitations. Modeling with simulation could be useful for such systems, but it may take time to get dependable results because the model needs to be run several times to get an average result. Moreover, to consider the impact of any modification in each input parameter, separate runs of the simulation model are required which makes the running times more severe. Using analytical modeling in this context not only can help providers to assess power consumption and performance related measures in different situations, but it can also be useful in terms of budget and time constraints. To fulfill this requirement, an analytical model based on Stochastic Activity Networks (SANs) [26, 28] is proposed in this paper to evaluate the power consumption and performance related measures of virtualized servers of a cluster in cloud computing environments.

The proposed SAN simultaneously models two techniques in different steps, namely *grouping servers in power consumption pools* and *DVFS technique*, to save energy within cloud data centers. The main contribution of the proposed model, which makes it different from other related work presented in this context, can be summarized in three major cate-

gories. (1) A more flexible analytical model based on SAN formalism is proposed to evaluate both power consumption and performance measures of virtualized servers. Compared to the previously presented models, the proposed stochastic model is scalable enough to model large scale systems, and simultaneously, compute performance and power consumption estimates. (2) The DVFS mechanism is modeled using SANs, which makes it possible to scale up or down the speed of CPUs as needed. Therefore, not only the horizontal scaling of a cloud system with increasing and decreasing the working VMs can be modeled using the proposed SAN, but also the vertical scaling of VMs can be appropriately addressed in the proposed model. (3) SANs are exploited in the body of an optimization problem to appropriately change the power/speed mode of VMs according to the overall workload of the corresponding cluster. The model presented in this paper uses the specific characteristics of SANs and its capability to code inside input/output gates to scale up and down the speed of CPUs according to the workload of the cluster. It allows to maintain the power consumption of physical servers in an acceptable level while performance considerations are taken into account.

The remaining of this paper is organized as follows. Section 2 presents the related work done on analytical modeling of cloud and grid computing environments. In Section 3, the formal definition of SANs and the background of structure and behavior of SAN models are given. Section 4 presents the architecture of the system under study, and explains the concept of DVFS. In Section 5, the proposed SAN model is described with details and some interesting measures are introduced which can be assessed with the proposed model. Section 6 presents numerical results obtained from solving the proposed SAN model and compares the model with some baselines and another model previously proposed in this area. Moreover, the sensitivity of output parameters of the proposed model to the variation of some input parameters is analyzed in Section 6. Finally, Section 7 concludes the paper, and presents some guidelines for future work that can be done in this research field.

2. Related Work

There have been proposed many analytical models to evaluate the power consumption, performance, and dependability of distributed computing systems. These models usually use various extensions of Markov Chains (MCs) and Petri Nets (PNs) for modeling and evaluation of grid and cloud computing systems.

Ghosh et al. have presented a scalable stochastic analytical model for performance quantification of an IaaS cloud [21]. They divided the physical machines into three different pools according to their power consumption and provisioning delays. Since the performance of IaaS clouds can be affected by a large set of parameters (e.g. workload, system charactristics and management policies), a multi-level interacting sub-model solution was proposed in [21] to overcome the intractability of traditional analytical models [21]. It has been shown that the proposed model based on the interaction of sub-models is much more scalable than the monolithic model. Ghosh et al. [22] have also proposed an analytical model based on MCs to an end-to-end performability analysis of a cloud service where two Quality of Service (QoS) metrics, service availability and provisioning response delays, are taken into account. The

most important novelty of that approach was to reduce the complexity of analysis by dividing the overall model into sub-models, and then, obtaining the overall solution by iterating over individual sub-model solutions.

Bruneo et al. have proposed the Stochastic Reward Net (SRN)-based model for comparing two different energy policies in green clouds [9, 10]. The SRN models presented in [9] and [10] consider physical machines that are moved among three pools named *sleep*, *idle* and *running*, and allocate VMs on top of those physical machines to the user requests. Although these models can appropriately estimate the power consumption and performance related measures of green clouds, they cannot handle the DVFS technique inside the SRN, so they use some parameters obtained by analyzing the proposed model to compute the power. The main drawback of those models is that no optimization method in assigning user requests to the running VMs is supported which causes to run all VMs at their highest speed. Both allocation mechanisms presented in [9] and [10] only assign requests to the VMs without considering the number of running VMs and waiting requests. Longo et al. have proposed an SRN model to evaluate the availability of large scale IaaS clouds in which component failures are quite common [25]. The physical machine failures are considered to happen when they migrate among three pools: *cold*, *warm*, and *hot*. These failures may lead to occasional system downtime and eventual violation of SLAs on the cloud service availability. To reduce the complexity of analysis and the solution time, interacting SRN models are used and the interacting sub-models were solved with the fixed-point iteration method.

Wang et al. have investigated the DVFS technique and proposed a Continuous-Time Markov Decision Process (CTMDP) model to maximize the total profit of a cluster in a cloud environment [34]. The total profit is defined as the total price earned from serving the clients subtracted by the operation cost of the cluster. The total price depends on the average request response time for each client, while the operation cost is related to the total energy consumption. The CTMDP model only takes into account the price without paying any attention to the performance requirements. Tian et al. have proposed a Stochastic Service Decision Net (SSDN) to investigate energy-efficient speed scaling for web servers in cloud computing [33]. The SSDN considers two different speeds for a web server, selected according to the number of waiting tasks when a user submits a task to the system. The process of accessing the cache, memory and disk was also modeled using the SSDN model in [33]. Entezari-Maleki et al. have proposed three SRN models to jointly evaluate performance and availability of a single grid resource, and their use to model an entire grid environment [19]. Since the exact monolithic model of an entire grid shows state space explosion, two approximate models were proposed to estimate the performability of a grid [19]. All SRN models for a single grid resource and the models presented for an entire gird environment are aimed to evaluate the performability without paying any attention to the power consumption. Entezari-Maleki et al. have also proposed a Markov Reward Model (MRM) to model and evaluate the performability of a single grid resource [17]. Although MRM presented in [17] is a mathematical model for a grid resource, it ignores details of the resource such as various numbers of processors inside a resource, failing the processors servicing grid and local tasks, and energy consumption of processors which can be seen in real systems.

Roohitavaf et al. have proposed a SAN model to evaluate the availability of virtual data centers in cloud computing [29]. The proposed SAN was used to investigate the impact of different managing mechanisms in the service availability of virtual data centers in IaaS clouds. Entezari-Maleki et al. have proposed a SAN model for assessing the availability of a grid environment composed of grid manager and many grid resources [18]. The SAN model presented in [18] considers the grid manager to be composed of several servers, and grid resources to be geographically distributed within the grid environment, trying to dispatch the tasks submitted to the manager to the servers and resources. The impact of applying different task scheduling policies to dispatch both grid and local tasks among grid resources has been appropriately studied by the SAN model proposed in [18].

In addition to the papers referred above, which apply analytical models to evaluate wide range of parameters in cloud and grid systems, there have been proposed some measurement-based approaches to compute the power consumption and performance related measures on distributed systems. As examples of these efforts in cloud computing, several work can be mentioned [6, 7, 13, 16, 23]. Each of the methods has advantages and disadvantages. One drawback of those approaches is that only a few of them evaluate both power consumption and performance related measures. The research work that consider both, only do experiment-based analysis and do not propose any analytical framework to be applied to a wide range of similar systems. Moreover, the problem of switching servers among different power consumption modes, and scaling up and down the speed of CPUs considering the workload of the related cluster have not been tackled properly. Hence, we try to address the aforementioned problems in this paper by proposing a SAN to model and compute the power consumption and performance related measures of virtualized servers in an IaaS cloud.

3. Overview of SANs

Stochastic Activity Networks (SANs) are the stochastic generalization of Petri Nets (PNs) generally defined for the modeling and analysis of distributed real-time systems [26, 28]. SANs are more powerful and flexible than other stochastic extensions of PNs such as Stochastic Petri Nets (SPNs) and Generalized Stochastic Petri Nets (GSPNs) [27]. If $\mathcal{N}$ denotes the set of natural numbers, an activity network can be formally defined as a 7-tuple $(P, IA, TA, IG, OG, IR, OR)$ [27], where:

- P is a finite set of *places*,
- IA is a finite set of *instantaneous activities*,
- TA is a finite set of *timed activities*,
- IG is a finite set of *input gates*; each input gate has a finite number of *inputs*; to each $G \in IG$ with m inputs, it is associated a function $f_G : \mathcal{N}^m \to \mathcal{N}^m$, called the $function$ of G, and a predicate $g_G : \mathcal{N}^m \to \{true, false\}$, called the *enabling predicate* of G,
- OG is a finite set of *output gates*; each output gate has a finite number of *outputs*; to each $G \in OG$ with m outputs, it is associated a function $f_G : \mathcal{N}^m \to \mathcal{N}^m$, called the $function$ of G,

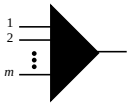
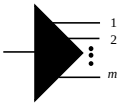
Element	*Place*	*Timed activity*	*Instantaneous activity*	*Input gate*	*Output gate*
Graphical notation					

Figure 1: Graphical representation of SAN elements

- $IR \subseteq P \times \{1, ..., |P|\} \times IG \times (IA \cup TA)$ is the *input relation*; IR satisfies the following conditions:
  - for any $(P_1, i, G, a) \in IR$ such that G has m inputs, $i \leq m$,
  - for any $G \in IG$ with m inputs and $i \in \mathcal{N}, i \leq m$, there exist $a \in (IA \cup TA)$ and $P_1 \in P$ such that $(P_1, i, G, a) \in IR$,
  - for any $(P_1, i, G_1, a), (P_1, j, G_2, a) \in IR$, $i = j$ and $G_1 = G_2$,
- $OR \subseteq (IA \cup TA) \times OG \times \{1, ..., |P|\} \times P$ is the *output relation*; OR satisfies the following conditions:
  - for any $(a, G, i, P_1) \in OR$ such that G has m outputs, $i \leq m$,
  - for any $G \in OG$ with m outputs and $i \in \mathcal{N}, i \leq m$, there exist $a \in (IA \cup TA)$ and $P_1 \in P$ such that $(a, G, i, P_1) \in OR$,
  - for any $(a, G_1, i, P_1), (a, G_2, j, P_1) \in OR$, $i = j$ and $G_1 = G_2$.

In General, SANs are probabilistic extensions of activity networks that are equipped by a set of activity time distribution functions, reactivation predicates and enabling rate functions. The nature of the extension is similar to the one that constructs SPNs from classical PNs. SANs were defined with the express purpose of facilitating unified performance/dependability (performability) evaluation, as well as more traditional performance and dependability evaluation [31]. More detailed information about SANs can be found in [26, 27, 28, 31].

A SAN model can be graphically represented by simple elements such as circles, bars (or rectangles) and triangles. Fig. 1 shows the graphical notations of SAN primitives. Modeling and analysis with SANs need a software tool to help to construct and evaluate the model. The original definition of SANs has been used as a modeling formalism in some modeling tools, such as METASAN, UltraSAN and Möbius [31]. All of these tools are intended for the evaluation of operational aspects (such as performance, dependability or performability) of systems. In this paper, the Möbius tool [15] is used to construct and analyze the proposed SAN model.

4. System Description

Virtualization is an important enabling technology for many large data centers and cloud computing environments. A cloud service provisions VMs with specific characteristics in terms of number and frequency of CPU cores, memory, and storage according to the user request [20]. VMs are deployed on servers each of which may be shared by multiple VMs [21]. System virtualization is commonly supported on the hypervisor technology. Hypervisor/Virtual Machine Monitor (VMM) is a software that allows the virtualization of resources [24, 37]. There are various types of Hypervisors/VMMs in virtualized systems, but we skip describing them because it is out of the scope of our paper. A general scheme of a virtualized cloud data center is shown in Fig. 2, which contains N physical servers and M VMs on top of each server.

Although virtualization is a useful tool for unifying the access and to reduce the power consumption of a server in a data center, deploying more VMs into a single server results in an increase in the execution time due to the bottleneck caused by sharing resources like CPU, memory, and storage [38]. In addition to the increment in execution time of requests in virtualized systems imposed by resource sharing, the VM deployment and provisioning time should also be taken into account in modeling and assessing the performance of a virtualized system. The performance degradation due to resource sharing can be mitigated by carefully selecting the number of VMs on top of a single server, and the deployment and provisioning times can be optimized by grouping servers into multiple pools characterized by different degrees of provisioning delays and power consumptions. In this paper, we consider the servers to be grouped into three pools named *cold* (turned off), *warm* (turned on, but not ready) and *hot* (running). Power consumption of servers in the cold pool can be neglected since they are turned off. When a cold server is selected to be moved into the warm pool, a predefined time named *server wake-up time* is taken to the server be turned on. Although the warm servers are turned on and consume power, they are not ready to service user requests, so they should be moved into running state (hot pool) before being able to host VMs. The power consumption of hot servers is much more than the power usage of warm servers. Deciding about appropriately moving servers among cold, warm, and hot pools not only influences the overall power consumption of the cluster, but also it affects the response time to user requests. As an example, Fig. 2 shows *Server* 1 and *Server* i in cold and warm pools, respectively. Moreover, since both *Server* 2 and *Server* N in Fig. 2 are considered to be in hot pool, they can host VMs and be allocated to user requests.

In order to reduce the power consumption of a cluster while maintaining in an acceptable level of performance, we consider that the DVFS technique is applied in each server. DVFS has proven to be an effective method of achieving low power consumption for the CPU while meeting the performance requirements. The key idea behind the DVFS technique is to dynamically scale the supply voltage level of the CPU, so as to provide enough speed for processing the system workload meeting the computation time and/or throughput constraints [14, 34]. According to the CPU allocated to a VM to run the specific user requests, we can scale up or down the processing speed of the VM (its relevant CPU) to process the requests faster if it is required or save power if it is the case. Therefore, considering

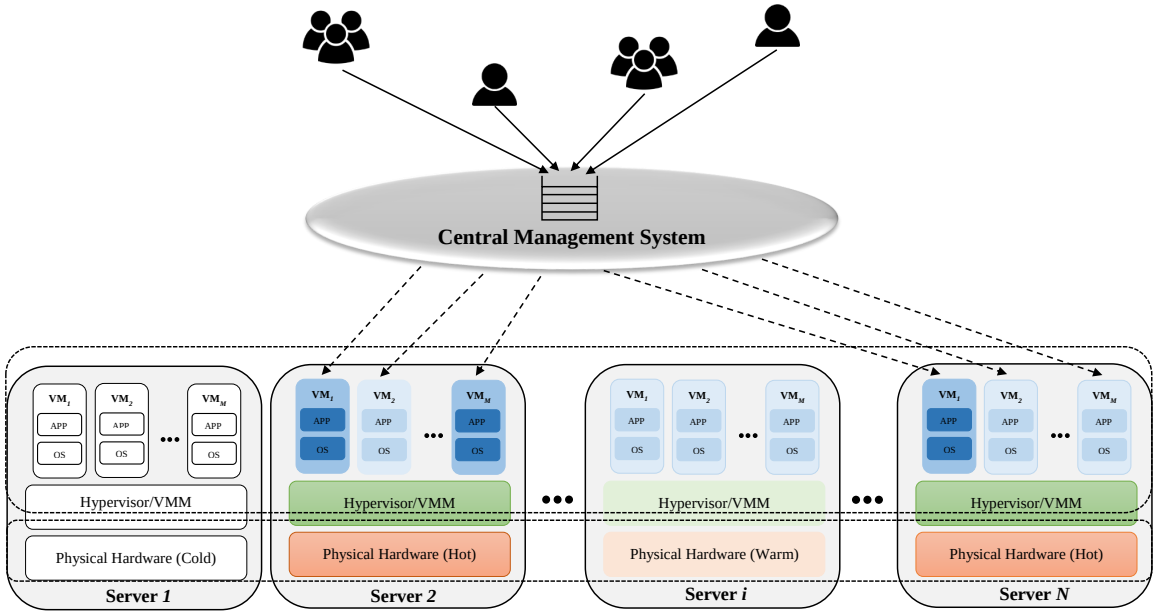


Figure 2: The architecture of the virtualized system considered in this paper

the workload of the system, we can derive a mechanism to switch the VMs among various power consumption/processing speed modes to reach a reasonable trade-off between power consumption and system performance. According to the type of the CPU in each server and its capability for supporting DVFS technique, we can consider several modes for a VM on top of a server. For example in Fig. 2, VMs 1 and M in *Server* 2, and VM 1 in *Server* N run in the same DVFS mode and VM 2 in *Server* 2 and VMs 2 and M in *Server* N run in another DVFS mode.

5. The Proposed Model

This section presents the proposed SAN model in detail. Firstly, each component of the proposed SAN and its relation with the others are explained in Subsection 5.A, and then power consumption and performance related measures, which can be assessed by steady-state analysis of the proposed model, are introduced in Subsection 5.B.

A. The SAN Model

Fig. 3 shows the SAN model proposed in this paper to evaluate the performance and power consumption of virtualized servers of a cluster in an IaaS cloud with the architecture presented in Fig. 2. Although the reference architecture shown in Fig. 2 is a simple and abstract architecture of an IaaS cloud data center, mathematically modeling this simple system structure, and analytically evaluate performance and power consumption measures are of utmost importance. Moreover, in previous research papers presented in this context

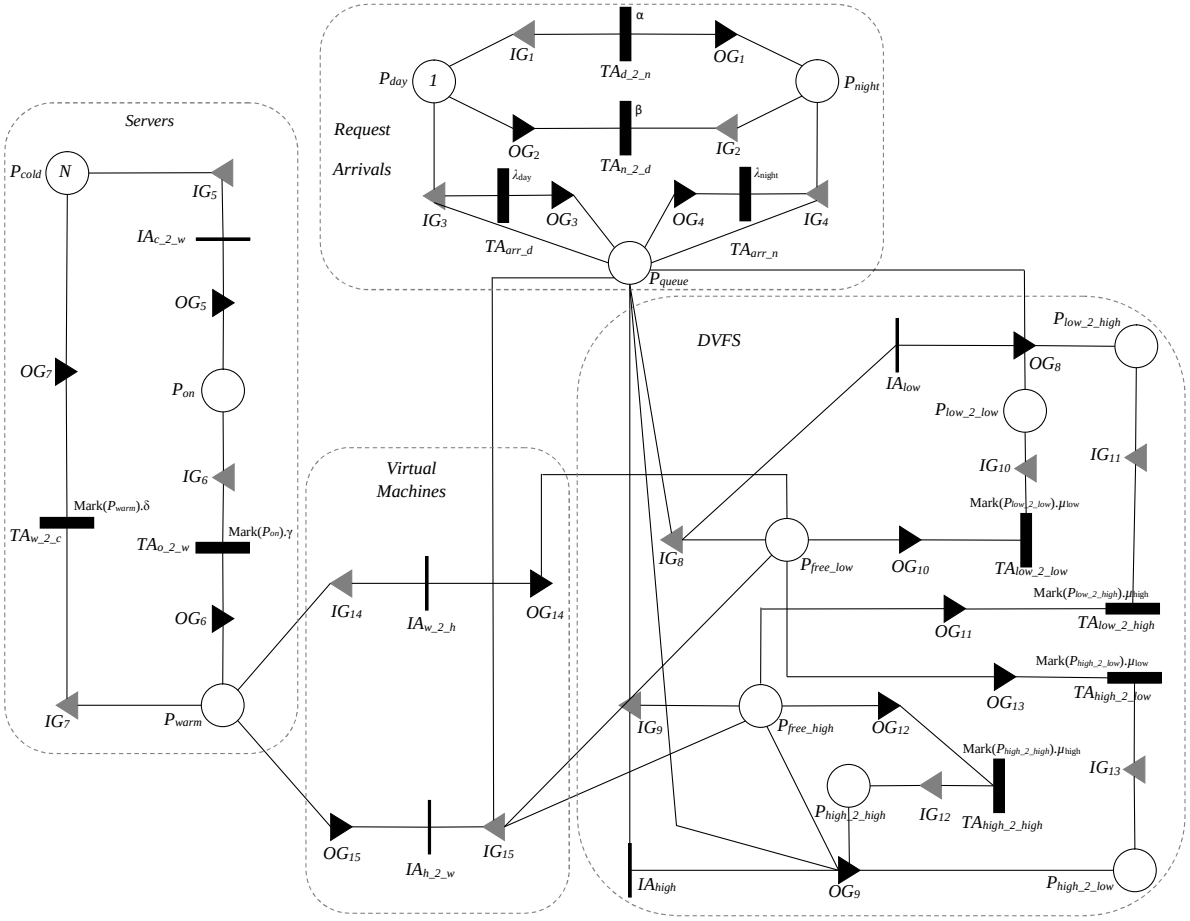


Figure 3: The proposed SAN model

that model real complex distributed systems using analytical methods considering an abstract architecture of a system is a conventional way to relax the problem to be solved by mathematics [7, 9, 10, 11, 14, 17, 18, 19, 20, 21, 22, 25, 29, 33, 34].

Since the proposed SAN model is complex and contains many components, we conceptually divide the entire model into *four* different parts named *Request Arrivals*, *Servers*, *Virtual Machines* and *DVFS*, and present them in this subsection with details.

- *Request Arrivals*. The request arrivals to the system is modeled by the part of Fig. 3 surrounded with a box tagged with *Request Arrivals*. We use Markov Modulated Poisson Process (MMPP) to model different arrival rates of requests during day and night hours. MMPPs, which are a subclass of the doubly stochastic Poisson processes, can be used to model time-varying arrival rates and important correlations between interarrival times [8]. The input parameters associated with this part of the SAN are: (1) the rate of transitioning between *day* and *night* hours (α and β), (2) the request arrival rate at day and night hours (λ_{day} and λ_{night}), and (3) the buffer size

Table 1: Gate predicates/functions of *Request Arrivals* part of the SAN model shown in Fig. 3

Gate	**Predicate**	**Function**
IG_1	`Pday->Mark() > 0`	`Pday->Mark()--;`
OG_1		`Pnight->Mark()++;`
IG_2	`Pnight->Mark() > 0`	`Pnight->Mark()--;`
OG_2		`Pday->Mark()++;`
IG_3	`(Pday->Mark() > 0) && (Pqueue->Mark() < S)`	`;`
OG_3		`Pqueue->Mark()++;`
IG_4	`(Pnight->Mark() > 0) && (Pqueue->Mark() < S)`	`;`
OG_4		`Pqueue->Mark()++;`

of the cluster queue (S). The times associated with timed activities $TA_{d\_2\_n}$, $TA_{n\_2\_d}$, TA_{arr_d}, and TA_{arr_n} follow exponential distributions with rates α, β, λ_{day}, and λ_{night}, receptively.

The existence of a token in place P_{day} (P_{night}) shows that it is day (night) and we should use the request arrival rate for day (night) hours to model request arrivals to the system. Referring to Fig. 3, if there is a token in place P_{day}, the timed activity $TA_{d\_2\_n}$ is activated and it can complete. With completion of this activity, a token is removed from place P_{day} by input gate IG_1 and a token is deposited into place P_{night} by output gate OG_1. The inverse mechanism is done to show moving from night to day by timed activity $TA_{n\_2\_d}$, and gates IG_2 and OG_2. The input gate IG_3 (IG_4) checks the existence of a token in place P_{day} (P_{night}), and if there is a token in that place, it activates the timed activity TA_{arr_d} (TA_{arr_n}). When the activity TA_{arr_d} (TA_{arr_n}) completes, a token is put in place P_{queue} by output gate OG_3 (OG_4) to show that a new request just arrived to the queue, and it should be serviced by the system. We consider a single queue for the cluster with limited capacity equal to S. If the number of tokens inside place P_{queue} reaches S, the input gate IG_3 (IG_4) prevents the activity TA_{arr_d} (TA_{arr_n}) to complete, so the arriving requests are rejected from the system. Table 1 shows the input predicates/functions of input gates IG_1 to IG_4 and the output functions of output gates OG_1 to OG_4. In this table, the notation $Pi- > Mark()$ represents the number of tokens inside place P_i.

- *Servers.* As described in Section 4, and it can be seen in Fig. 2, we consider the physical servers to transit among three pools to save power: *cold*, *warm*, and *hot*. Transitions among the pools are modeled in the part *Servers* shown in the most left side of the SAN model shown in Fig. 3. The input parameters of this part of the proposed SAN are: (1) the number of servers (N), (2) the wake-up rate of a server which is the rate of moving a server from the cold pool to the warm pool (γ), (3) the rate of moving a server from the warm pool to the cold pool (δ), which is actually a timer that tells a

warm server to be switched off when it is idle for a certain amount of time.

Place P_{cold} represents the cold servers of a cluster. We assume that there are N cold servers in the beginning, and if there is no request waiting to receive service in the cluster, all N servers are in their cold states. If there is at least one waiting request in place P_{queue}, and there is no VM to be assigned to the waiting request, one of the servers in the cold pool is selected to be moved to the warm pool. This movement is done by instantaneous activity $IA_{c\_2\_w}$, but before moving a token from place P_{cold} to place P_{on}, we should make sure that the number of already existing tokens inside place P_{on} is not enough to service all waiting requests. This check is required to save power and prevent the cold servers to be turned on when they are not actually needed. As shown in Table 2, the check is done by predicate of input gate IG_5, where $(P_{free\_low}- > Mark()) + (P_{free\_high}- > Mark())$ denotes the number of idle VMs existing in the cluster, as described below in the $DVFS$ part of the SAN model.

The existence of a token in place P_{on} shows that a server has already been selected to be moved into the warm pool, causing the timed activity $TA_{o\_2\_w}$ to be activated. The time required for a cold server to become a warm server follows an exponentially distributed function with mean $1/\gamma$. The rate assigned to timed activity $TA_{o\_2\_w}$ is $Mark(P_{on}).\gamma$, which shows that the completion rate of this activity is marking dependent. So, the actual completion rate of activity $TA_{o\_2\_w}$ is computed as $k.\gamma$, where k is the number of tokens in place P_{on} and $1/\gamma$ is the mean time required to wake-up a server (moving a server from the cold pool to the warm pool). With completion of timed activity $TA_{o\_2\_w}$, a token is removed from place P_{on} and deposited into place P_{warm} through gates IG_6 and OG_6. Tokens in place P_{warm} represent warm servers waiting to go to the hot pool. If a server waits for more than a predefined time in the warm pool, we switch off this server to save power. This action is done by completion of timed activity $TA_{w\_2\_c}$, which moves a token from place P_{warm} to place P_{cold}. The rate of this activity also follows an exponentially distributed function with rate δ for each server. Since the completion rate is place dependent, we show it as $Mark(P_{warm}).\delta$ in Fig. 3. The predicates and functions corresponding to all input and output gates of the $Servers$ part of the proposed SAN are given in Table 2.

- $DVFS$**.** As mentioned in Section 1 and Section 4, we model the application of the DVFS technique in the proposed SAN to reduce power consumption. In addition to the DVFS mechanism, other techniques such as resource allocation methods are also very important in this area, and they have direct impact on both performance and power consumption measures, but herein we are just focused on modeling DVFS technique using SANs. It is worthwhile to mention that applying resource allocation techniques in SAN models has been done before in [18], in the context of grid computing environments. Although the model presented in [18] just considers the performance and dependability measures, paying no attention to power consumption aspects, using the methods mentioned in that paper, one can model resource allocation mechanisms in other distributed computing systems such as IaaS clouds. Applying DVFS mecha-

Table 2: Gate predicates/functions of *Servers* part of the SAN model shown in Fig. 3

Gate	**Predicate**	**Function**
IG_5	`(Pcold->Mark() > 0) && (Pwarm->Mark() == 0) && (Pqueue->Mark() > M*(Pon->Mark())) && (Pfree_low->Mark() + Pfree_high->Mark() == 0)`	`Pcold->Mark()--;`
OG_5		`Pon->Mark()++;`
IG_6	`Pon->Mark() > 0`	`Pon->Mark()--;`
OG_6		`Pwarm->Mark()++;`
IG_7	`Pwarm->Mark() > 0`	`Pwarm->Mark()--;`
OG_7		`Pcold->Mark()++;`

nism, different power consumption modes and their related processing speeds can be considered for each VM. Without loss of generality, assume we have only two states of power consumption and processing speeds, designated *low* and *high*. One can easily extend the number of power consumption/processing speed states and use the method mentioned below to handle them. The $DVFS$ part of the proposed model can be seen in the right most dashed box of the SAN shown in Fig. 3.

Places P_{free_low} and P_{free_high} in this figure show the idle VMs in the system in low and high states, respectively. If there is a token in place P_{free_low} (P_{free_high}), it means that there is an idle VM ready to be allocated to a request submitted to the queue. The mechanisms related to putting tokens in place P_{free_low} or removing tokens from both places P_{free_low} and P_{free_high} are described in the next bullet, where the $Virtual\ Machines$ part of the proposed SAN is explained. Input gates IG_8 and IG_9 check the existence of tokens in places P_{queue}, P_{free_low}, and P_{free_high}. In order to save power and scale down idle high speed VMs when they are not needed, we assign higher priority to instantaneous activity IA_{high} against IA_{low}. For this reason, the input gate IG_9 first checks the existence of at least one token in both places P_{queue} and P_{free_high}, and if the condition is evaluated to true, the instantaneous activity IA_{high} is activated and it completes. The predicate of input gate IG_8 checks the condition in which there is no token inside place P_{free_high} and at least one token in both places P_{queue} and P_{free_low}. According to this mechanism, if there is at least one idle high speed VM, we allocate this VM to the waiting request, and an idle low speed VM is allocated to a waiting request only when there is no idle high speed VM. We can easily extend this mechanism to a system which contains more than two power consumption/processing speed states and check VMs with higher power/speed to be allocated to a waiting request each time an allocation wants to be happened. The predicates and input functions of input gates IG_8 and IG_9 are given in Table 3.

With completion of instantaneous activity IA_{high}, a token from place P_{queue} together with another token from place P_{free_high} is removed and a token is placed in either

place $P_{high\_2\_high}$ or place $P_{high\_2\_low}$. The selection between two places $P_{high\_2\_high}$ and $P_{high\_2\_low}$ is done inside output gate OG_9 in which the numbers of tokens in places P_{free_high} and P_{queue} are compared to each other. If the number of tokens inside place P_{free_high} is higher than the number of tokens in place P_{queue}, it shows that the number of idle high speed VMs is more than it is required, so it is better to scale down those VMs, so the output gate OG_9 puts a token in place $P_{high\_2\_low}$. The existence of a token in place $P_{high\_2\_low}$ shows that a VM has been allocated to a request, but it has been scaled down, so the VM runs in its low power consumption/processing speed mode. If the condition in output gate OG_9 is evaluated to false, which means that the number of tokens in place P_{free_high} is equal to or less than the number of tokens in place P_{queue}, a token is deposited into place $P_{high\_2\_high}$. The existence of a token in place $P_{high\_2\_high}$ shows that an idle high speed VM has been allocated to a request, and it still runs in its high speed mode. Similarly, output gate OG_8 selects one of places $P_{low\_2\_high}$ or $P_{low\_2\_low}$ to put token in. According to the predicate of input gate IG_8, the instantaneous activity IA_{low} completes only when there is no token in place P_{free_high}. Therefore, in output gate OG_8, we are sure that there is no high speed VM, so we can decide to scale up a low speed VM inside this gate if required. To do this, we compare the number of tokens in place P_{queue} with the queue size. If the number of tokens in place P_{queue} is more than the half of the queue size, it means that we need to service the requests faster because the queue is going to be full if we do not appropriately service the waiting requests. In this case, we scale up a low speed VM when allocating it to a waiting request, so a token is deposited into place $P_{low\_2\_high}$. Otherwise (the number of tokens inside place P_{queue} is equal to or less than the half of the queue size), a token is deposited into place $P_{low\_2\_low}$ to allocate a low speed VM to a waiting request. The output functions of output gates OG_8 and OG_9 modeling the aforementioned mechanism are given in Table 3. It should be mentioned that other mechanisms can also be modeled inside those gates to scale up/down VMs according to the user requests and provider policies.

According to the predicate of input gate IG_{12} (IG_{13}), if there is a token in place $P_{high\_2\_high}$ ($P_{high\_2\_low}$), the timed activity $TA_{high\_2\_high}$ ($TA_{high\_2\_low}$) is activated and it can complete. With completion of activity $TA_{high\_2\_high}$ ($TA_{high\_2\_low}$), a token is deposited into place P_{free_high} (P_{free_low}) by output gate OG_{12} (OG_{13}) to show that a high (low) speed VM already finished servicing a request, and it is available to be allocated to another request. It is worthwhile to mention that the completion rate of timed activity $TA_{high\_2\_high}$ ($TA_{high\_2\_low}$) is marking dependent and it is equal to $Mark(P_{high\_2\_high}).\mu_{high}$ $\big(Mark(P_{high\_2\_low}).\mu_{low}\big)$, where μ_{high} (μ_{low}) is the service rate of a single VM when it is in high (low) power consumption/processing speed mode. Similarly, the functionality of timed activities $TA_{low\_2\_low}$ and $TA_{low\_2\_high}$ can be easily inferred according to the explanation of timed activities $TA_{high\_2\_high}$ and $TA_{high\_2\_low}$ given above. The predicate and input/output functions of input/output gates related to this part of the proposed SAN are presented in Table 3.

Table 3: Gate predicates/functions of $DVFS$ part of the SAN model shown in Fig. 3

Gate	**Predicate**	**Function**

IG_8	`(Pfree_low->Mark() > 0) &&`
	`(Pqueue->Mark() > 0) &&`
	`(Pfree_high->Mark() == 0)`
		`Pqueue->Mark()--;`
		`Pfree_low->Mark()--;`

OG_8		`if (Pqueue->Mark() > S/2)`
		`  Plow_2_high->Mark()++;`
		`else`
		`  Plow_2_low->Mark()++;`

IG_9	`(Pfree_high->Mark() > 0) &&` `(Pqueue->Mark() > 0)`	`Pqueue->Mark()--;` `Pfree_high->Mark()--;`

OG_9		`if (Pfree_high->Mark() > Pqueue->Mark())`
		`  Phigh_2_low->Mark()++;`
		`else`
		`  Phigh_2_high->Mark()++;`

IG_{10}	`Plow_2_low->Mark() > 0`	`Plow_2_low->Mark()--;`
OG_{10}		`Pfree_low->Mark()++;`
IG_{11}	`Plow_2_high->Mark() > 0`	`Plow_2_high->Mark()--;`
OG_{11}		`Pfree_high->Mark()++;`
IG_{12}	`Phigh_2_high->Mark() > 0`	`Phigh_2_high->Mark()--;`
OG_{12}		`Pfree_high->Mark()++;`
IG_{13}	`Phigh_2_low->Mark() > 0`	`Phigh_2_low->Mark()--;`
OG_{13}		`Pfree_low->Mark()++;`

Table 4: Gate predicates/functions of *Virtual Machines* part of the SAN model shown in Fig. 3

Gate	Predicate	Function
IG_{14}	`(Pfree_low->Mark() + Pfree_high->Mark() == 0) && (Pqueue->Mark() > 0) && (Pwarm->Mark() > 0)`	`Pwarm->Mark()--;`
OG_{14}		`Pfree_low->Mark() = Pfree_low->Mark() + M;`
IG_{15}	`(Pfree_low->Mark() + Pfree_high->Mark() - Pqueue->Mark()) >= M`	`Pfree_low->Mark() = Pfree_low->Mark() - (M - Pfree_high->Mark()); Pfree_high->Mark() = 0;`
OG_{15}		`Pwarm->Mark()++;`

- *Virtual Machines*. In this part of the proposed SAN which is surrounded by a dashed box tagged with *Virtual Machines* in Fig. 3, switching a server from the warm state to the hot state and vice versa is modeled. The input gate IG_{14} checks the condition required to change the status of a server from warm to hot. If there is a waiting request in place P_{queue} and no idle VM in places P_{free_low} and P_{free_high} to be allocated to the waiting request, the input gate IG_{14} checks the existence of a token in place P_{warm}, and if there is any, it activates instantaneous activity $IA_{w\_2\_h}$. With completion of activity $IA_{w\_2\_h}$, a token is removed from place P_{warm} and deposited M tokens into place P_{free_low}, where M is the number of VMs on top of each server, to show that the status of a server has been changed from warm to hot. It should be mentioned that we consider all VMs of a server to be in low state when the server just transits to the hot state. This assumption is made to save power. If there are more waiting requests in place P_{queue}, our model automatically changes the status of low power/speed VMs to high power/speed state to service the requests faster by executing output function of output gate OG_8.

  This part of the proposed SAN is also responsible for switching the status of hot servers to the warm if required. To do this, input gate IG_{15} checks the number of idle VMs inside both places P_{free_high} and P_{free_low}. If the number of idle VMs is equal to or greater than M, it collects M tokens from both places P_{free_high} and P_{free_low} and activates instantaneous activity $IA_{h\_2\_w}$. After completion of activity $IA_{h\_2\_w}$, a token is put in place P_{warm} to show that a server has already been switched from the hot state to the warm state. In order to save power, the tokens in pale P_{free_high} have higher priority to be removed, so we first decrease the number of tokens in place P_{free_low} by $M - Mark(P_{free\_high})$, and then, empty all tokens of place P_{free_high}. The predicates and input functions of input gates IG_{14} and IG_{15} together with output functions of output gates OG_{14} and OG_{15} are presented in Table 4.

B. Performance Measures

In order to evaluate the performance and power consumption of the servers modeled by the proposed SAN, we need to define some reward functions on the SAN. This is done by assigning appropriate reward rate to each feasible marking of the SAN model, and then, computing the expected reward rates in the steady-state. Let ρ_i denote the reward rate assigned to marking i of the SAN model shown in Fig. 3. If $\pi_i(t)$ denotes the probability for the SAN model to be in marking i at time t, then the expected reward at time t can be computed as $\sum_i \pi_i(t)\rho_i$. The expected steady-state reward can be computed using the same formula by replacing $\pi_i(t)$ by π_i, representing the steady-state probability for the SAN model to be in marking i. The interesting measures in the proposed SAN model are as follows.

Blocking probability of arriving requests ($\mathbf{P}_b$). The expected blocking probability can be defined as the steady-state probability that arriving requests are rejected from the system due to the queue saturation. It can be computed by assigning the reward rate shown in Eq. 1 to the network, which compares the number of tokens inside place P_{queue} with the queue size S. If the number of tokens is greater than or equal to S, the queue is saturated and the arriving requests will be rejected from the system.

$$\rho_i = \begin{cases} 1, & Mark(P_{queue}) \geq S \\ 0, & otherwise \end{cases} \tag{1}$$

Instant service probability ($\mathbf{P}_i$). It is the probability that a request arrives to the system and is instantaneously assigned to an idle VM without experiencing any waiting time. To compute this time, the number of already queued requests and idle VMs should be checked. If the number of waiting requests is zero and there exists at least one idle VM, the request submitted to the queue can be immediately assigned to an idle VM. The reward rate to compute this measure is given in Eq. 2.

$$\rho_i = \begin{cases} 1, & \big(Mark(P_{queue}) = 0\big) \ \ and \ \ \big(Mark(P_{free\_low}) + Mark(P_{free\_high}) > 0\big) \\ 0, & otherwise \end{cases} \tag{2}$$

Mean waiting time (W). The mean waiting time is the expected time spent by requests in the queue to be assigned to a VM. To compute this time, we should first compute the mean number of waiting requests in queue. Thus, a reward function is set up to return the number of tokens in place P_{queue} in the steady-state named $E\big[Mark(P_{queue})\big]$. Afterwards, having the mean queue length, we can apply Little's law [8] to compute the mean waiting time of requests in the queue as Eq. 3.

$$W = \frac{E\big[Mark(P_{queue})\big]}{\lambda_{eff}} \tag{3}$$

where λ_{eff} denotes the effective request arrival rate, computed by Eq. 4.

$$\lambda_{eff} = (1 - P_b).\lambda \tag{4}$$

where λ is the requests arrival rate to the system. Since the arrival rate of the requests in day and night hours are different, we use the rate function given in Eq. 5 to compute the measure λ_{eff}.

$$\rho_i = \begin{cases} \lambda_{day}, & \big(Mark(P_{queue}) < S\big) \ \ and \ \ \big(Mark(P_{day}) = 1\big) \\ \lambda_{night}, & \big(Mark(P_{queue}) < S\big) \ \ and \ \ \big(Mark(P_{night}) = 1\big) \\ 0, & otherwise \end{cases} \tag{5}$$

Throughput. It is the rate of request completion by the system. In order to compute the total throughput of the system, the throughput of all timed activities servicing user requests should be computed, and then combined together. For example, in the SAN model shown in Fig. 3, there are four timed activities servicing requests, namely $TA_{low\_2\_low}$, $TA_{low\_2\_high}$, $TA_{high\_2\_low}$, and $TA_{high\_2\_high}$. To compute the throughput of the entire network, the throughputs of all these activities need to be computed. For example, let $\pi(P_{low\_2\_low} = k)$ denote the steady-state probability of there being k tokens in place $P_{low\_2\_low}$. Hence, the throughput of timed activity $TA_{low\_2\_low}$ can be computed by Eq. 6.

$$Throughput = \sum_k \pi(P_{low\_2\_low} = k).k.\mu_{low} \tag{6}$$

where μ_{low} is the service rate of a VM in low power/speed mode. Similarly, the throughput of high power/speed VMs can be computed by replacing their corresponding place probabilities and service rates.

Mean response time (R). This is the expected time required to respond user requests, which is computed for each request as the summation of the time spent by the request in waiting queue and the time taken by a VM to process the request. Since we have computed both the mean waiting time and the throughput of the system, we can use Eq. 7 to compute the mean response time of the system to user requests.

$$R = W + \frac{1}{Throughput} \tag{7}$$

Power consumption (P). As already mentioned in *Section* 4, we consider two mechanisms in our model to reduce the power consumption. One mechanism is scaling down idle VMs when they are not required, and the other one is powering off servers whose VMs are not being used. Since, the power consumption of VMs in different power/speed modes and the power usage of servers in warm and hot pools are different, we need to compute the number of VMs belonging to each power/speed mode and the number of servers in the warm pool. Afterward, we can multiply the number of VMs and servers to their corresponding power consumption value. Let $E\big[Mark(P_{warm})\big]$ denote the number of servers in warm pool, which is obtained by writing a reward function which returns the number of tokens in place P_{warm} in the steady-state. Moreover, let $E[Low\_VMs]$ denote the expected number of VMs

in low power/speed mode which is computed by summing the mean number of tokens in places P_{free_low}, $P_{low\_2\_low}$, and $P_{high\_2\_low}$ in the steady-state. Similarly, $E[high\_VMs]$ shows the expected number of VMs in high power/speed mode which is the sum of tokens in places P_{free_high}, $P_{high\_2\_high}$, and $P_{low\_2\_high}$ in the steady-state. Now, using Eq. 8, the overall power consumption of the system can be computed.

$$P = E\big[Mark(P_{warm})\big].P_{warm} + E[low\_VMs].P_{low} + E[high\_VMs].P_{high} \tag{8}$$

where P_{warm} denotes the power consumption of a physical server in the warm state, and P_{low} and P_{high} denote the power consumption of a single VM in its low power/speed and high power/speed modes, respectively. It is worthwhile to mention that the power model used in this paper is a simplified version of the model proposed in [7] and [13], only the CPU power consumption is herein considered. In the basic formula given in [7] and [13], beyond the power consumption of CPU, the power consumption of cache, DRAM and disk have also been taken into account.

6. Numerical Results

In this section, numerical results obtained from analytically solving the proposed SAN model with Möbius tool [15] are presented. To assess the impact of powering off/on the servers and scaling down/up the VMs on power consumption and performance measures of virtualized servers in a cloud, different scenarios are considered, and the sensitivity of results to the input parameters are analyzed. In the following, three subsections are devoted to present numerical results and analyze the proposed model.

In Subsection 6.A, we use real data reported in related art to compare the proposed model with two baselines in which all VMs of a server run in a single mode. The aim of this subsection is to show the impact of DVFS technique on power consumption and performance of the systems. Furthermore, the results reported in Subsection 6.A show that the proposed model can be applied to real environments, and it can model and evaluate both performance and power consumption appropriately. In Subsection 6.B, we change the value of input variables to the numbers used in [9] and [10] to be able to fairly compare the proposed model with the other models. Since the models presented in [9] and [10] do not consider the DVFS technique, we do not claim that our model outperforms these models. We select them for comparison because they are the only existing models in literature in the same context of our proposed model, albeit with different formalism and assumptions. So, we only compare the results obtained from our model with the results of those models to show how the DVFS technique can be analytically modeled and evaluated in the context of SAN formalism, and to show the advantage of this technique in appropriately using power in servicing user requests to achieve efficiency. Finally, Subsection 6.C presents the output parameters of the proposed model, when its input parameters change, so we can analyze the sensitivity of final results to the variation of each input parameter.

A. *Comparing the Proposed Model with Two Baselines*

In order to use more realistic data, we use some real settings reported in related publications. The number of servers in each cluster of the cloud is considered to be 20 ($N = 20$). Several clusters in clouds are mentioned to work with about 20 servers, for example the clusters of the Leiden University (LU), University of Amsterdam (UvA), and Netherlands Institute for Radio Astronomy (ASTRON) with 16, 16 and 24 nodes, receptively, collaborating in DAS-4 project [1] which experiments Green Clouds [13]. The number of VMs on top of each server is typically 2, 4 or 8 [6, 13, 23], and we consider this number to be 4 in our experiments ($M = 4$). The size of the queue is 30 ($S = 30$), the typical size considered in many papers in this context is in the range between 10 and 50 [9, 10, 11, 19, 29, 34]. The power consumption and processing speed of a VM in its low state are $P_{low} = 1.1\ W$ and $\mu_{low} = 300\ req/sec$, respectively, and similarly for a VM in its high state we have $P_{high} = 1.3\ W$ (a variations of about 20% from P_{low}) and $\mu_{high} = 400\ req/sec$ [14]. The average power consumption of a server in warm state (P_{warm}) is assumed to be half as the power consumption when all VMs of the server is running at their low mode [34]. The rates of transitioning between day and night (α and β) are set to $1/12\ h^{-1}$. The mean wake-up time of a server ($\frac{1}{\gamma}$), which is the mean time required for a server to transit from the cold pool to the warm is set to 3 ms [35]. The value set for idle time-out when a server is in its idle mode is 20 ($\frac{1}{\delta} = 20\ min$), which is the default value for Microsoft web servers [2].

In this subsection, we compare the proposed SAN model with two baselines. In *Baseline* 1, all VMs on top of a physical server are considered to run in their low power/speed mode. Therefore, the power consumption and processing speed considered for all VMs are $P_{low} = P_{high} = 1.1\ W$ and $\mu_{low} = \mu_{high} = 300\ req/sec$, respectively. In *Baseline* 2, all VMs are considered to run in their high power/speed mode, so the power consumption and processing speed parameters for all VMs are equal to $P_{high} = P_{low} = 1.3\ W$ and $\mu_{high} = \mu_{low} = 400\ req/sec$, respectively. Since the proposed model runs a combination of VMs at high and low levels, so it consumes less power but provide degraded performance in comparison with the situation in which all VMs run in their high speed, and vice versa when all VMs run at low speed. Therefore, the aim of comparing the proposed model with these two baselines is to demonstrate the multi-purpose optimization of performance and power consumption.

It is worthwhile to mention that scale up and scale down, also known as vertical scaling, means increasing or decreasing the size of a VM in response to an existing workload. It is different from horizontal scaling, also referred to as scale out and scale in, where the number of VMs differs according to the workload. If we change the number of VMs for a specific user, it may need to power off/on physical servers, but by increasing or decreasing the size of a VM (VMs), when it is permitted, the old data of the VM(s) are retained without any need to deploy a new VM. Scaling up/down can be useful when: (1) a service built on VMs is under-utilized, e.g. at weekends, in which reducing the size of a VM can reduce the power consumption and monthly costs, and (2) increasing VM size to cope with large demands raised through executing an application without creating additional VMs. On the other hand, provisioning a VM or a set of VMs may need a physical server to be powered on if all capacity of already available servers is allocated to the existing users and there is no

enough VM(s) to be allocated to the new user(s). In this case, a server is selected from the pool of cold servers, and powered on to be allocated to the new user(s). In contrast with powering on a server when it is required, we can power off a server when all its VMs have been released. In this case, a free server is powered off to save power. As another example, we can mention the migration technique in a virtualized environment, which migrates the VMs of a physical server to another server to be able to power off an under-utilized server. In the proposed model, vertical and horizontal scalings are done by *DVFS* and *Servers* parts of the SAN shown in Fig. 3, respectively. The results obtained for the proposed model and two baselines are reported in the graphs of Fig. 4.

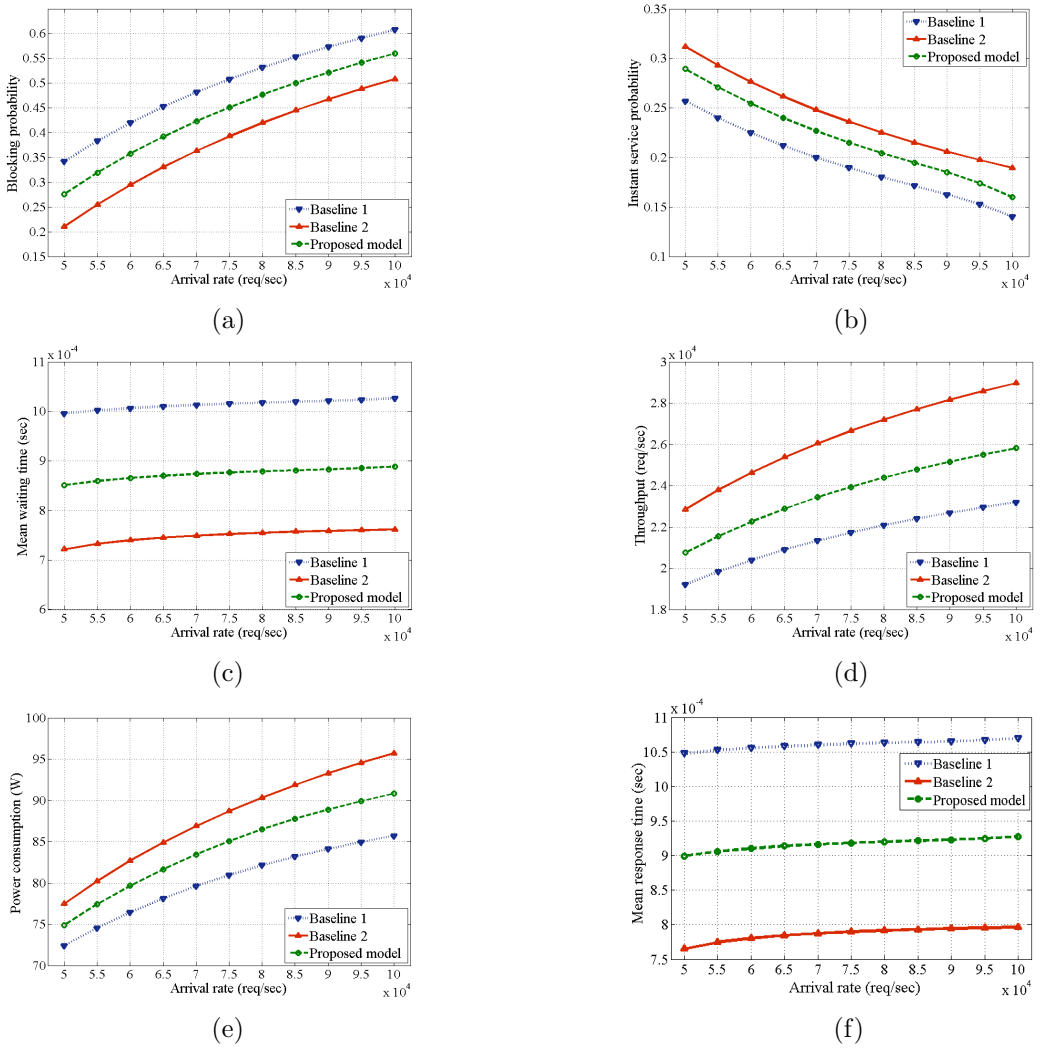


Figure 4: Results obtained for comparing the proposed model with two baselines considering the: (a) blocking probability; (b) instant service probability; (c) mean waiting time; (d) throughput; (e) power consumption; and (f) mean response time.

In all plots of Fig. 4, the arrival rate of requests in day hours (λ_{day}) is varied from $50,000$ to $100,000$ req/sec and the arrival rate in night hours is $\lambda_{night} = \frac{1}{5}\lambda_{day}$. As can be seen in Fig. 4, the measures related to the performance of the system (e.g. the blocking probability, instant service probability, mean waiting and response times, and system throughput) for *Baseline* 2 which runs all VMs in high power/speed mode, are much better than the related measures of *Baseline* 1 which runs all VMs in low power/speed mode. On the contrary, the power consumption of *Baseline* 2 is higher than that of *Baseline* 1, which is an expected result. Since the proposed model uses both high and low power/speed VMs to service user requests, the performance and power consumption values are between those of *Baseline* 1 and *Baseline* 2. Hence the results reported in Fig. 4 emphasize that scaling up/down running VMs can help cloud providers to decrease the power consumption of the data centers, with a penalty in terms of performance that can be modeled and assessed. This performance degradation, which is controllable should be acceptable.

B. Comparing the Proposed Model with Previously Presented Models

In this subsection, we use the values reported in [9] and [10] to compare the proposed model with the SRN model presented in those papers. The values for input parameters of the model are reported in Table 5. Since the model proposed in [10] has some differences in comparison with our model, in the sense of allocation mechanism, turning on/off VMs and switching off idle servers, we made the required modifications in our model for the purpose of comparison. As mentioned earlier, the model presented in [10] does not consider the DVFS technique and uses another formalism to solve the problem, so we only compare the results obtained from both models to show the effectiveness of DVFS technique introduced in the proposed SAN. The power consumption and two performance measures, blocking probability and throughput, obtained with the proposed SAN model and the model in [10] (the saturation strategy) are given in Fig. 5-(a) to Fig. 5-(c). In Fig. 5, the horizontal axis shows the arrival rate of requests to the system in day hours (λ_{day}) which varies from 0.1 to 1 req/min. The request arrival rate in night hours is $\lambda_{night} = \frac{1}{5}\lambda_{day}$. As it can be seen in Fig. 5-(a) and Fig. 5-(c), although the power consumption of the proposed model is lees than the model presented in [10], the throughput resulted from our model is more than the throughput of the model in [10]. Fig. 5-(b) shows that the blocking probability of both models increases with increasing the arrival rate of requests, however with negligible difference, the blocking probability of our model is slightly more than that of model presented in [10]. Since the result of other performance measures is very close to each other (like results presented in Fig. 5-(b)), we do not present them in Fig. 5. Comparing the results obtained with the proposed model and the model presented in [10] shows that using DVFS technique, dispatching requests among VMs with different power/speed modes and switching VMs among different states of power consumptions and processing speeds according to the system workload, we can reach a better performance with acceptable power usage.

In addition to comparing the proposed SAN model with two baselines, and the model presented in [10], we compare it with the system that does not use the virtualization mechanism, so it allocates each server to only one user. Although some performance measures such as the mean response time may be improved by a non-virtualized system, the power

Table 5: The configuration of the system considered in Subsection 6.B

Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
N	20	M	2	S	10
μ_{low}	$5/18\ req/h$	μ_{high}	$1/3\ req/h$	P_{low}	$7\ W$
P_{high}	$10\ W$	P_{warm}	$7\ W$	α	$1/12\ h^{-1}$
β	$1/12\ h^{-1}$	γ	$1/2\ min^{-1}$	δ	$1\ min^{-1}$

consumption of such a system is significantly higher than that of a virtualized system as shown in Fig. 5-(d). It is worthwhile to mention that since an idle physical server is switched off when there is no waiting request in the queue, the instant service probability of a non-virtualized system is *zero* for all arrival rates.

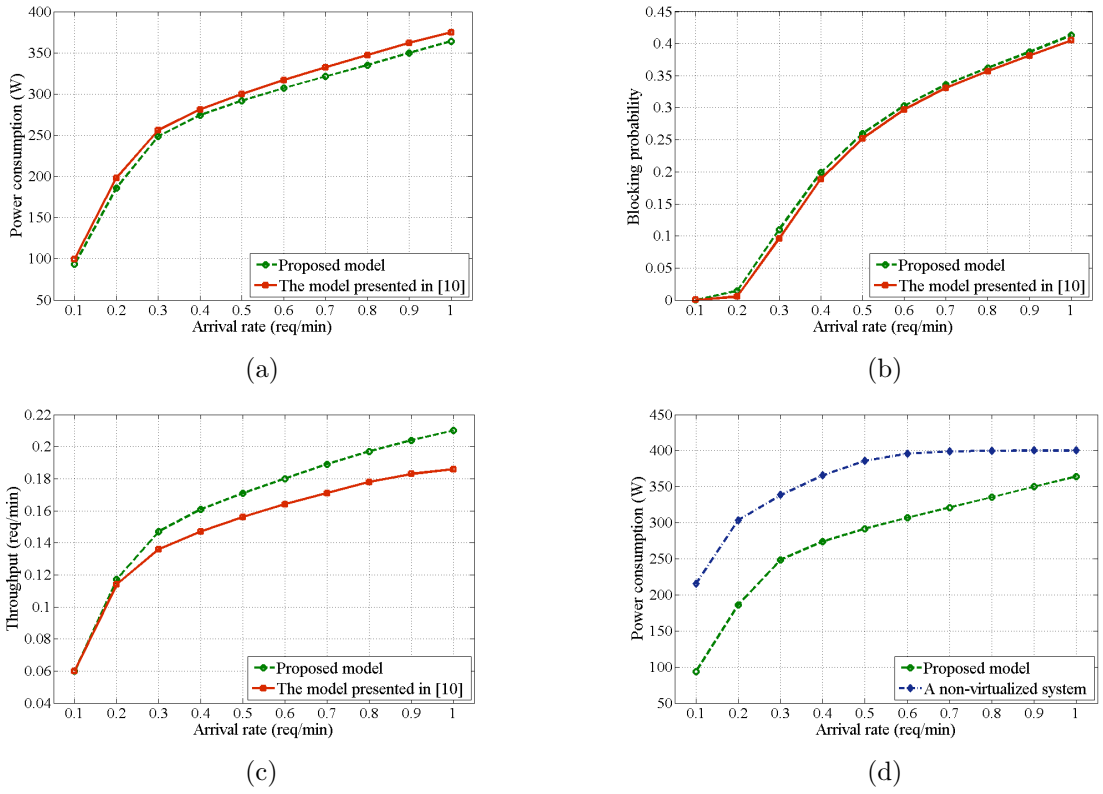


Figure 5: Results obtained for comparing our proposed model with the model presented in [10] considering the: (a) power consumption; (b) blocking probability; and (c) throughput. (d) Power consumption resulted from our model and a non-virtualized system.

C. Sensitivity Analysis

In this subsection, we study the sensitivity of output parameters to the variation of input parameters. To reach this, each input parameter should be varied in a valid range, and then, the sensitivity of a result to the variation of that parameter is analyzed. For the sake of brevity, we only investigate the sensitivity of the results to the variation of *three* important parameters: (1) the rate of moving a cold physical server to the warm state, named wake-up rate shown by γ, (2) the rate of moving a server from the warm pool to the cold pool named time-out rate denoted by δ, and (3) the rate of transitioning between day and night hours denoted by α and β, respectively. However, other input parameters can also be included in the study. To achieve this, we consider a cloud system with the values reported in Subsection 6.A for the input parameters. Moreover, the arrival rate of requests in all scenarios studied in this subsection is set to $50,000$ req/sec.

In the first scenario, which studies the impact of variation of the wake-up rate (γ) on the final results, we change the value of mean wake-up time ($1/\gamma$) from 1 ms to 10 ms with a time step of 1 ms. It should be mentioned that in the system considered in Subsection 6.A, this parameter is set to 3 ms. According to the results obtained from the steady-state analysis of the proposed SAN, the measures *throughput* and *power consumption* of the network are almost fixed numbers, which do not change by modifying the mean wake-up time. The value of throughput and power consumption are around 20757 and 74.91, respectively, for all 10 variations of mean wake-up time. However, other output parameters change with changing the mean wake-up time of servers as shown in Fig. 6.

As can be seen in Fig. 6-(a), the blocking probability of requests increases when the mean wake-up time gets higher, which is an expectable result, because according to the proposed SAN shown in Fig. 3, increasing the mean wake-up time (decreasing the value of rate γ) causes more tokens to be queued inside place P_{queue}, resulting in the waiting queue of requests to reach the maximum size, so the newly arriving requests are blocked. Moreover, increasing the number of waiting requests in the queue and delaying in responding the waiting requests, which are the results of increasing the wake-up time of a server, cause the mean waiting time and mean response time of requests to increase as shown in Fig. 6-(c) and Fig. 6-(d). On the other hand, increasing the mean wake-up time of a server causes the probability of finding the waiting queue empty, and immediately servicing a newly arriving request to increase, which results in decreasing the instant service probability as shown in Fig. 6-(b).

In the second scenario, we fix the mean wake-up time to 3 ms, and vary the mean time-out ($1/\delta$) from 3 to 30 min, with a time step of 3 min. According to the results obtained from the steady-state analysis of the proposed SAN, the only measure which varies by varying the time-out rate is *power consumption*. The values of the blocking probability, instant service probability, mean waiting time, throughput, and mean response time are 0.276107, 0.289449, 8.5128×10^{-4}, 20756.627, and 8.9946×10^{-4}, respectively, for all 10 variations of time-out rate. The values of power consumption for $(1/\delta) = 3$ to 30 min are shown in Fig. 7.

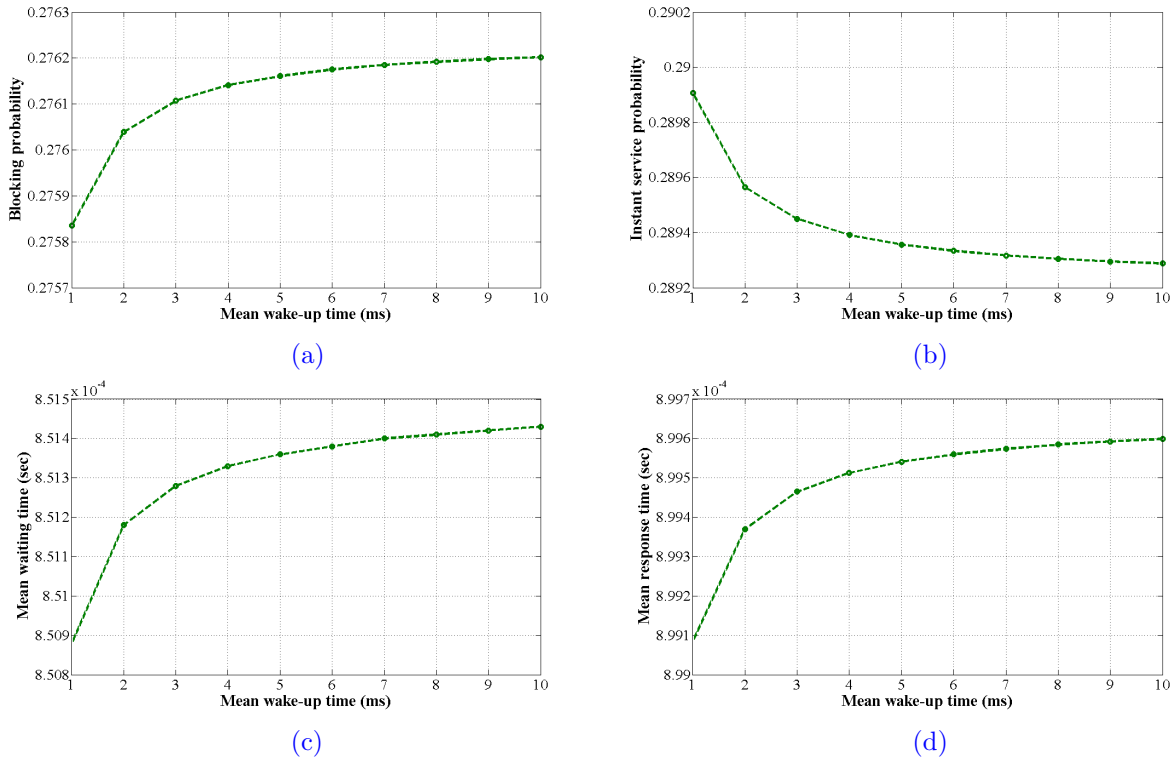


Figure 6: Results showing the impact of variation of mean wake-up time of servers on the: (a) blocking probability; (b) instant service probability; (c) mean waiting time; and (d) mean response time.

As can be observed in Fig. 7, increasing the value of time-out (decreasing the time-out rate) increases a bit the power consumption. This is a reasonable conclusion because powering a server off reduces power consumption of a cloud data center, and if it takes more time to power off a server, the server will consume more power. Analyzing the SAN model

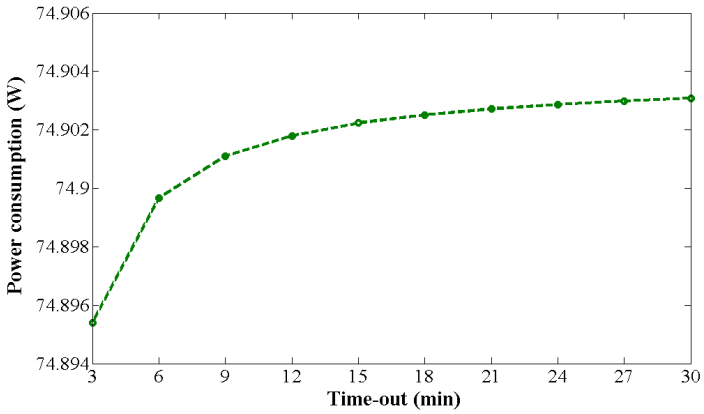


Figure 7: The impact of variation of time-out value on power consumption of an IaaS cloud data center

Table 6: Results obtained from the variation of transitioning rate between day and night hours

$1/\alpha$ (h)	P_b	P_i	W (sec)	Throughput (req/sec)	R (sec)	P (W)
8	0.276104	0.289454	8.5128×10^{-4}	20756.511	8.99460×10^{-4}	74.902315
9	0.276105	0.289452	8.5128×10^{-4}	20756.539	8.99462×10^{-4}	74.902416
10	0.276105	0.289451	8.5128×10^{-4}	20756.572	8.99463×10^{-4}	74.902505
11	0.276106	0.289450	8.5128×10^{-4}	20756.606	8.99464×10^{-4}	74.902582
12	0.276107	0.289449	8.5128×10^{-4}	20756.627	8.99464×10^{-4}	74.902662
13	0.276107	0.289448	8.5128×10^{-4}	20756.657	8.99466×10^{-4}	74.902741
14	0.276108	0.289447	8.5129×10^{-4}	20756.677	8.99467×10^{-4}	74.902809
15	0.276109	0.289445	8.5129×10^{-4}	20756.708	8.99468×10^{-4}	74.902888
16	0.276109	0.289444	8.5129×10^{-4}	20756.731	8.99469×10^{-4}	74.902968

presented in Fig. 3, we can also get the same result, because if the value of time-out gets higher (the rate δ decreases), tokens spend more time inside place P_{warm} resulting in more power consumption according to Eq. 8.

In the third scenario, to study the impact of variation of the transitioning rates between day and night hours (α and β) on final results, we fix the mean wake-up time and time-out values to 3 ms and 20 min, respectively, and vary the rates α and β. In the experiment considered in Subsection 6.A, the values of rates α and β are considered to be the same and equal to $1/12$ h^{-1}, which shows day and night have nearly exactly the same length (12 hours). Here, we assume the day hours to vary from 8 to 16, and consequently, the night hours vary from 16 to 8. Hence, the value of $1/\alpha$ changes from 8 to 16 and $1/\beta$ changes from 16 to 8. The output parameters resulting from the steady-state analysis of the proposed SAN, including the blocking probability (P_b), instant service probability (P_i), mean waiting time (W), throughput, mean response time (R), and power consumption (P) of the new setting, are reported in Table 6.

As can be observed in Table 6, changing the transitioning rate between day and night hours changes all output parameters, but the modifications are really negligible. Since we considered, in all experiments, the rate of arrivals in day hours to be 5 times bigger than that of night hours, it is expected that decreasing the parameter α will increase the arrival rate of requests. Increasing the arrival rate of requests to the system accordingly increases the blocking probability, mean waiting time, throughput, mean response time, and power consumption, and deceases the instant service probability. This can be seen in Table 6, when we traverse the table from the first row to the last in which parameter α decreases ($1/\alpha$ increases).

7. Conclusions and Future Work

Virtualization is one of the techniques which can be applied to a data center to reduce power consumption. It helps providers to consolidate several virtual servers on a single physical server. Using Virtual Machines (VMs) on top of a single physical server reduces the amount of hardware in use, and consequently, reduces the cost. Recently emerging technology, cloud computing, leverages the virtualization of computing resources aiming at allowing customers to provision resources on-demand over the internet on a pay-as-you-go pricing strategy. Cloud providers try to deliver reliable QoS to the users in terms of Service Level Agreements (SLAs) specifying QoS targets (e.g. throughput, response time and so forth) and economical penalties associated to SLA violations. Hence, to evaluate the performance delivered by each managing mechanism, and its related power consumption and cost, cloud providers have to deal with power-performance trade-off as aggressive consolidation of VMs can lead to performance loss.

To fulfill the need for analytical models to assess the power consumption and performance of virtualized servers in clouds, we propose a Stochastic Activity Network (SAN) model to evaluate power and performance of resource management techniques in Infrastructure-as-a-Service (IaaS) clouds. The proposed model considers two levels of power optimization techniques, one at the physical server layer and the other at the VM level. It models the Dynamic Voltage and Frequency Scaling (DVFS) technique which is a mechanism that dynamically adjusts the voltage and frequency to save power according to the workload of the system. The results obtained from solving the proposed stochastic analytical model show that our proposed model can be applied to real systems, and the optimization approach used in its body shows that better results are achieved in comparison to the models previously proposed. One interesting extension which can be considered as a future work, would be to use colored extensions of SANs and Petri Nets (PNs) to model the virtualized servers for making it possible to handle different user requests with different requirements in the network. If we could use different colors of tokens inside places representing different user requests and VMs, it would be possible to assign a given request type to only some predefined types of VMs. In the proposed model, all user requests have the same type, and the number of VMs on top of a physical server is a fixed number for all servers. Moreover, all VMs are considered to be of a single type. Using colored extensions of SANs and PNs (e.g Colored Petri Nets (CPNs)), one can model different types of requests and VMs in the system.

Another interesting extension to the SAN model presented in this paper is to use Markov Decision Process (MDP) in the $DVFS$ part of the proposed SAN. Although the mechanism applied to the input and output gates of the $DVFS$ part of the proposed SAN finally leads to appropriate power consumption and performance, there is no guarantee to say that this mechanism is the optimal one. If we use MDP in this part of the network, and define the goal of the network as minimizing the power consumption, we may reach better results. There have been proposed some formalisms to combine MDPs and PNs like Markov Decision Petri Nets (MDPNs) and Markov Decision Well-formed Nets (MDWNs), which may be used for this purpose.Using interactive sub-models to overcome the scalability problem raised by the proposed SAN, when a large number of physical servers or VMs is set, can be mentioned

as another guideline for future work. Since the proposed model is a monolithic model, it eventually encounters the state space explosion problem like other monolithic models previously presented in this area. Although the proposed model can easily handle real settings of cloud systems, dividing it into interactive sub-models and approximating the monolithic model by some good approximation models can decrease the number of states of the underlying Markov chain.

Considering the networking aspects of an IaaS cloud data center, and taking the topology of the network and the connectivity of the resources into account are other interesting and important extensions to the current work. In the architecture considered in this paper, the physical servers are not connected to each other, and only their connection with central manager has been considered in the investigation. Since the performance and energy consumption of a data center depend on the structure of the network too, modeling a more realistic architecture of the system can help to reach more dependable results. As another interesting work which can be done in this research area is modeling different resource allocation mechanisms inside an IaaS cloud, in order to reduce power consumption. Although DVFS is a typical technique for enhancing energy efficiency of a computing system, IaaS cloud data centers often also employ other techniques to reduce power consumption. Since resource allocation technologies can be applied to consolidate VMs in order to minimize the number of physical servers for hosting VMs, they have direct impact on both performance and energy consumption. Therefore, modeling DVFS technique together with appropriate resource allocation mechanisms using SANs can provide even more realistic view of an IaaS cloud data center.

Acknowledgments

This work was supported by national funds through Fundacao para a Ciencia e a Tecnologia (FCT) with reference UID/CEC/50021/2013, and project PTDC/EEI-ELC/3152/2012.

References

[1] DAS-4. http://www.cs.vu.nl/das4/. Accessed: January 2016.

[2] Microsoft TechNet: Library − > Configure Request-Processing for a Web Server − > Configure Idle Time-out Settings for an Application Pool. https://technet.microsoft.com/. Accessed: January 2016.

[3] Natural Resources Defence Council. http://www.nrdc.org/energy/data-center-efficiency-assessment.asp. Accessed: January 2016.

[4] The NIST Definition of Cloud Computing, Information Technology Laboratory, National Institute of Standards and Technology, United States Department of Commerce. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf. Accessed: January 2016.

[5] M. Ali, S. U. Khan, and A. V. Vasilakos. Security in cloud computing: Opportunities and challenges. *Information Sciences*, 305(1):357–383, 2015.

[6] J. Bi, Z. Zhu, R. Tian, and Q. Wang. Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center. In *The IEEE 3rd International Conference on Cloud Computing*, pages 370–377, Miami, FL, USA, July 5–10, 2010.

[7] A. E. H. Bohra and V. Chaudhary. VMeter: Power modelling for virtualized clouds. In *The IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum*, pages 1–8, Atlanta, GA, USA, April 19–23, 2010.

[8] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. John Wiley and Sons, second edition, 2006.

[9] D. Bruneo, A. Lhoas, F. Longo, and A. Puliafito. Analytical evaluation of resource allocation policies in green IaaS clouds. In *The 3rd International Conference on cloud and green computing*, pages 84–91, Karlsruhe, Germany, September 30–October 2, 2013.

[10] D. Bruneo, A. Lhoas, F. Longo, and A. Puliafito. Modeling and evaluation of energy policies in green clouds. *IEEE Transactions on Parallel and Distributed Systems*, 26(11):3052–3065, 2015.

[11] D. Bruneo, F. Longo, R. Ghosh, M. Scarpa, A. Puliafito, and K. S. Trivedi. Analytical modeling of reactive autonomic management techniques in IaaS clouds. In *The IEEE 8th International Conference on Cloud Computing*, pages 797–804, New York City, NY, USA, June 27–July 2, 2015.

[12] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599–616, 2009.

[13] Q. Chen, P. Grosso, K. van der Veldt, C. de Laat, R. Hofman, and H. Bal. Profiling energy consumption of VMs for green cloud computing. In *The IEEE 9th International Conference on Dependable, Autonomic and Secure Computing*, pages 768–775, Sydney, Australia, December 12–14, 2011.

[14] K. Choi, W. Lee, R. Soma, and M. Pedram. Dynamic voltage and frequency scaling under a precise energy model considering variable and fixed components of the system power dissipation. In *The IEEE/ACM International Conference on Computer Aided Design*, pages 29–34, San Jose, CA, USA, November 7–11, 2004.

[15] D. Daly, D. D. Deavours, J. M. Doyle, P. G. Webster, and W. H. Sanders. Möbius: an extensible tool for performance and dependability modeling. In B.R. Haverkort, H.C. Bohnenkamp, and C.U. Smith, editors, *Computer Performance Evaluation: Modelling Techniques and Tools*, volume 1786 of *Lecture Notes in Computer Science (LNCS)*, pages 332–336. Springer, 2000.

[16] G. Dhiman, G. Marchetti, and T. Rosing. vGreen: A system for energy-efficient management of virtual machines. *ACM Transactions on Design Automation of Electronic Systems*, 16(1):6:1–6:27, 2010.

[17] R. Entezari-Maleki, A. Mohammadkhan, H. Y. Yeom, and A. Movaghar. Combined performance and availability analysis of distributed resources in grid computing. *The Journal of Supercomputing*, 69(2):827–844, 2014.

[18] R. Entezari-Maleki and A. Movaghar. Availability modeling of grid computing environments using SANs. In *The 19th International Conference on Software, Telecommunications and Computer Networks*, pages 1–6, Dubrovnik, Croatia, September 15-17, 2011.

[19] R. Entezari-Maleki, K. S. Trivedi, and A. Movaghar. Performability evaluation of grid environments using stochastic reward nets. *IEEE Transactions on Dependable and Secure Computing*, 12(2):204–216, 2015.

[20] R. Ghosh, F. Longo, F. Frattini, S. Russo, and K. S. Trivedi. Scalable analytics for IaaS cloud availability. *IEEE Transactions on Cloud Computing*, 2(1):57–70, 2014.

[21] R. Ghosh, F. Longo, V. K. Naik, and K. S. Trivedi. Modeling and performance analysis of large scale IaaS clouds. *Future Generation Computer Systems*, 29(5):1216–1234, 2013.

[22] R. Ghosh, K. S. Trivedi, V. K. Naiky, and D. S. Kim. End-to-end performability analysis for Infrastructure-as-a-Service cloud: An interacting stochastic models approach. In *The16th IEEE Pacific Rim International Symposium on Dependable Computing*, pages 125–132, Tokyo, Japan, December 13–15, 2010.

[23] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya. Virtual machine power metering and provisioning. In *The 1st ACM symposium on Cloud computing*, pages 39–50, Indianapolis, IN, USA, June 10–11, 2010.

[24] D. Le and H. Wang. An effective memory optimization for virtual machine-based systems. *IEEE Transactions on Parallel and Distributed Systems*, 22(10):1705–1713, 2011.

[25] F. Longo, R. Ghosh, V. K. Naik, and K. S. Trivedi. A scalable availability model for Infrastructure-as-a-Service cloud. In *The IEEE/IFIP 41st International Conference on Dependable Systems and Networks*,

pages 335–346, Hong Kong, June 27-30, 2011.

[26] J. F. Meyer, A. Movaghar, and W. H. Sanders. Stochastic activity networks: Structure, behavior, and application. In *The International Workshop on Timed Petri Nets*, pages 106–115, Torino, Italy, 1985.

[27] A. Movaghar. Stochastic activity networks: A new definition and some properties. *Scientia Iranica*, 8(4):303–311, 2001.

[28] A. Movaghar and J. F. Meyer. Performability modeling with stochastic activity networks. In *The 1984 Real-Time Systems Symposium*, pages 215–224, Austin, TX, USA, 1984.

[29] M. Roohitavaf, R. Entezari-Maleki, and A. Movaghar. Availability modeling and evaluation of cloud virtual data centers. In *The 19th IEEE International Conference on Parallel and Distributed Systems*, pages 675–680, Seoul, South Korea, September 15-17, 2013.

[30] P. Samimi, Y. Teimouri, and M. Mukhtar. A combinatorial double auction resource allocation model in cloud computing. *Information Sciences*, (Available online 13 February 2014), 2014.

[31] W. H. Sanders and J. F. Meyer. Stochastic activity networks: Formal definitions and concepts. In E. Brinksma, H. Hermanns, and J.-P. Katoen, editors, *Lectures on Formal Methods and Performance Analysis*, volume 2090 of *Lecture Notes in Computer Science (LNCS)*, pages 315–343. Springer, 2001.

[32] E. Le Sueur and G. Heiser. Dynamic voltage and frequency scaling: the laws of diminishing returns. In *The International Conference on Power Aware Computing and Systems*, pages 1–8, Vancouver, BC, Canada, October 3, 2010.

[33] Y. Tian, C. Lin, Z. Chen, J. Wan, and X. Peng. Performance evaluation and dynamic optimization of speed scaling on web servers in cloud computing. *Tsinghua Science and Technology*, 18(3):298–307, 2013.

[34] Y. Wang, S. Chen, H. Goudarzi, and M. Pedram. Resource allocation and consolidation in a multi-core server cluster using a markov decision process model. In *The 14th International Symposium on Quality Electronic Design*, pages 635–642, Santa Clara, CA, USA, March 4–6, 2013.

[35] Y. Wang, Y. Liu, S. Li, D. Zhang, B. Zhao, M. F. Chiang, Y. Yan, B. Sai, and H. Yang. A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops. In *The 38th European Solid State Circuits Conference*, pages 149–152, Bordeaux, France, September 17–21, 2012.

[36] L. D. Xu. *Enterprise Integration and Information Architecture: A Systems Perspective on Industrial Information Integration.* Auerbach Publications, first edition, 2014.

[37] D. Ye, Q. He, H. Chen, and J. Che. A framework to evaluate and predict performances in virtual machines environment. In *The IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, pages 375–380, Shanghai, China, December 17–20, 2008.

[38] K. Ye, X. Jiang, Q. He, X. Li, and J. Chen. Evaluate the performance and scalability of image deployment in virtual data center. In C. Ding, Z. Shao, and R. Zheng, editors, *Network and Parallel Computing*, volume 6289 of *Lecture Notes in Computer Science (LNCS)*, pages 390–401. Springer, 2010.