Performability modelling and analysis of server virtualised systems subject to workload-dependent software aging

Zayneb Tayachi, Mohamed Escheikh, Kamel Barkaoui

Performability Modeling and Analysis of Server Virtualized Systems subject to Workload-dependent Software Aging

Zayneb Tayachi

Conservatoire National des Arts et Métiers, Cedric Lab, Paris, France
University of Tunis El Manar, ENIT, Syscom Lab, Tunis, Tunisia
E-mail: tayachi.zayneb@gmail.com

Mohamed Escheikh

University of Tunis El Manar, ENIT, Syscom Lab, Tunis, Tunisia
E-mail: mohamed.escheikh@enit.rnu.tn

Kamel Barkaoui

Conservatoire National des Arts et Métiers, Cedric Lab, Paris, France
E-mail: kamel.barkaoui@cnam.fr

Abstract: This paper tackles performability modeling and analysis of versatile server virtualized systems subject to workload-dependent software aging, failures and rejuvenation. We develop a modular modeling approach based on stochastic reward nets to investigate dependencies between several server virtualized system modules including virtual machine monitor, virtual machine, data intensive applications and workload-aware power management mechanism. Two case studies are considered, each of them accounts for a specific virtual machine monitor rejuvenation technique (Cold-VM or Migrate-VM). We show through numerical analysis how steady-state availability and power-performance metric are impacted by workload-dependent software aging and workload burstiness.

Keywords: Server virtualization systems, Workload-dependent software aging, Software rejuvenation, Performability, SRNs, Workload-aware power management.

Biographical notes:
Zayneb Tayachi received her engineering degree in telecommunications from National Engineering School of Tunis (ENIT), Tunisia, in 2013. She is currently pursuing her Ph.D. in ENIT. Her current research interests include queuing systems, petri nets, cloud computing, and power management.

Mohamed Escheikh received in 1992 the Diploma degree in electrical engineering. In 1994, he received a Master's degree and in 2001 a PhD degree with Distinction all in electrical engineering from National Engineering School of Tunis ENIT (Tunisia). In 2017, he has graduated a "Habilitation à Diriger des Recherches" from the University of Tunis El Manar. He is currently Maître de Conférences at ENIT since 2018, member of SYSCOM laboratory at ENIT since 1992 and associate researcher with VESPA Team Cedric CNAM Paris. His research interests include in particular dependability analysis, queuing

systems, petri nets, mobile networks optimization, cloud computing and power management.

Kamel Barkaoui full professor of Computer Science at Conservatoire National des Arts et Métiers (CNAM - Paris) since 2002. He holds a Ph.D in Computer Science (1988) and Habilitation à Diriger des Recherches (1998) from Université Paris 6 (UPMC). His most important research domains are about formal methods for specification, verification, control and performance evaluation of concurrent and distributed systems. He supervised more than 30 PhD thesis defended mostly on modelling and analysis of concurrent and distributed systems. He published 40 papers in International Journals, more than 100 peer-reviewed papers in international conferences and contributed to several books. He leaded or participated in more than 10 international research projects. He received the 1995 IEEE Int. Conf. on System Man and Cybernetics Outstanding Paper Award. Kamel Barkaoui served on PCs and as PC chair and OC chair of several international workshops and conferences in his areas of research. He was PC co-chair of the 3rd International Colloquium on Theoretical Aspects of Computing (ICTAC2006), General co-chair of the 18th International Symposium on Formal Methods (FM2012) and General chair of the 35th International Conference on Application and Theory of Petri Nets and Concurrency (Petri Nets2014) and of the 14th International Conference on Application of Concurrency to System Design (ACSD'2014). He is founder and SC chair of the International Conference on Verification and Evaluation of Computer and Communication Systems (VECoS). Kamel Barkaoui was a Guest Editor of Formal Aspects of Computing Journal (FACJ), Journal of Systems and Software (JSS), Innovations in Systems and Software Engineering (ISSE), ACM Transactions on Embedded Computing Systems (TECS) and referee for several international computer science journals He is currently Associate Editor for IEEE/CAA Journal of Automatica Sinica and the International Journal of Critical Computer-Based Systems (IJCCBS).

1 Introduction

Nowadays the massive investment of cloud computing service providers in infrastructure as a service (IaaS) around the world had significantly contributed to the spectacular growth of IT infrastructure and the large proliferation of data centers deployment. This trend is mainly boosted by increasingly rising demand for computing resources generated by modern services such as compute-intensive and scientific applications. In order to consolidate data centers' efficiency a great deal of progress remains to be achieved today and concerns mainly leveraging the full benefits of both virtualization technologies and dynamic power management (DPM) mechanisms. Virtualization enables both flexibility and operating cost benefits. This is mainly achieved through migration and consolidation techniques. DPM provides interesting opportunities to rationalize power consumption and to avoid dramatic resource wastage accordingly. This is primarily fulfilled through suitable power management (PM) mechanisms making the most of Advanced Configuration and Power Interface (ACPI) specifications and complying with green data center sustainability needs. Another requirement emerges today for modern data centers concerns providing high performance while ensuring high availability in order to support critical applications. Server virtualized systems (SVSs) are considered nowadays the cornerstone of data centers and their successful migration toward green architectures for data centers operators requires

to adopt cost-effective PM strategies. Such strategies are intended to minimize power consumption while meeting quality of service (QoS) and availability requirements. Efficient design and deployment of SVS is tributary of guarantying several requirements related to dynamic speed scaling processing, PM, availability, and performance in presence of transient failures whenever software fault tolerance techniques such as rejuvenation are deployed. These requirements should be jointly investigated to provide virtualized resource auto-scaling matching with sustainable and green computing purposes.

Investigating impact of time-varying workload with bursty nature such as data-intensive applications on SVS performability is of paramount importance. This is mainly interesting whenever versatile SVSs models are considered. Versatility may be concretized through accounting of workload-dependent software aging and including both proactive fault tolerance techniques such as software rejuvenation and DPM mechanism. We address in this paper performability modeling and analysis of SVS to evaluate several SVS properties through assessing performability metrics (Steady state availability, Mean waiting time, Mean loss rate), power metrics and performance-energy efficiency. The analysis is achieved for two kinds of software rejuvenation (i.e. Cold-VM and Migrate-VM) and for a given PM mechanism with fixed parameter (i.e. timeout) value. Notice that in previous works (Escheikh et al. (2017), Escheikh et al. (2014)), SVS performability is investigated for different timeout values for only one kind of rejuvenation (i.e. Cold-VM).

1.1 Research scope

The scope of this paper is on stochastic modeling and analysis of SVS handling cost-effective PM policy and subject to workload-dependent software aging. The main objective of our contribution is to evaluate workload impact on software aging and performability in SVSs. In this regard, we propose two versatile models based on non-Markovian SRNs of SVS handling data intensive applications of bursty nature. Each of these models deals with a specific virtual machine monitor (VMM) rejuvenation technique and a comparative performability analysis of these two models is investigated to highlight interactions and correlations between several SVS's entities involving:

1. VMM subject to workload dependent software aging, failure, and rejuvenation;
2. Virtual machine (VM) subject to workload dependent software aging, failure, and rejuvenation;
3. Power management component (PMC) implementing workload-aware PM mechanism;
4. resource provisioning with dynamic speed scaling.

1.2 Research challenges

The main challenges addressed through our contribution are:

1. How to develop versatile SVS stochastic reward nets (SRN) models taking into consideration workload-dependent software aging through a comprehensive modeling and modular approach involving several SVS entities namely VMM, VM, bursty workload, scalable resource provisioning, and workload-aware PM mechanism?

2. How to implement in SVS, a time-based workload-aware PM policy enabling energy savings for bursty workload?
3. How to make suitable and opportunistic decision about workload-aware switching between PMC soft states in order to minimize SVS energy consumption while keeping acceptable performance complying with service level agreement (SLA) constraints?
4. How to distinguish, from a design perspective, between different VMM rejuvenation techniques when conceiving versatile SVS SRN models?
5. How to define suitable reward-based measures in order to quantify dependencies between software aging, workload, and performability metrics (i.e. steady state availability, performability metrics and power metrics)?

The remainder of this paper is organized as follows. In Section 2 we address related work. In Section 3 we first examine workload handled by SVSs. We succinctly describe SVS issues such as software rejuvenation, software fault tolerance techniques and PM in SVSs. Lastly, we discuss in the same section correlations between workload, software aging, PM policy and power consumption in SVSs. Section 4 provides a detailed description of the proposed versatile SVS SRN models. Next, we illustrate the main contributions provided in this paper. In Section 5 we define some selected parameters and metrics of the versatile SVS SRN models cited above. In Section 6 we illustrate by means of numerical analysis how workload with bursty nature impacts software aging, steady-state availability, performability metrics, power metrics and power-performance metric in SVS. Finally Section 7 concludes this paper.

2 Related Work

In literature, different modeling approaches have been proposed to better describe the dynamic of different entities involved in virtualized systems. Authors in Cotroneo et al. (2014) and Cotroneo et al. (2011) propose surveys on the main studies given in literature on software aging and rejuvenation (SAR). In Alonso et al. (2013) authors present a comparative experimental study of six rejuvenation strategies, categorized in terms of granularity, to mitigate software aging effects. They show that performance overheads caused by software rejuvenation techniques are closely dependent on granularity level. In the same work, authors provide comprehensive guidelines to handle decision making related to rejuvenation scheduling algorithms and to select the suitable rejuvenation mechanism. Authors in Saito and Dohi (2016) present an approach based on semi-Markov models to investigate how software aging testing affects system behavior and its availability in the operational phase. It's worth mentioning the above models account for both aging-related bugs and non-aging-related bugs.
At PM level a set of works have been proposed. Chen et al. (2005) investigated trade-offs between energy consumption and performance based on SLA. They highlighted time overhead of switching on/off nodes and the relevant energy consumption on reliability. Nathuji et al. (2009) developed PM system for virtualized distributed architectures referred to as virtual power. They proposed the notion of soft power states and related mapping with hard power states to enable VM managing soft power with traditional techniques. Control theory is used successfully in Gaoa et al. (2013) and Beloglazov et al. (2012) to tackle issues on PM server. Elnozahy et al. (2003) applied two PM mechanisms

(Vary-On Vary-Off, VOVO) to explore power efficiency resource management problem in homogenous cluster hosting a single web application with SLAs requirements. In the above work, authors proposed five resource management policies to estimate the needed CPU frequency meeting response time requirements. At fault tolerant level, SRN modeling approach was used in Han and Xu (2013) and Machida et al. (2010) to build availability models for virtualized systems including VM and VMM subject to software rejuvenation. In Machida et al. (2010) authors present a comparative study between availability models of three VMM rejuvenation techniques referred to as Cold-VM, Warm-VM, and Migrate-VM. They show that rejuvenation trigger intervals of both VM and VMM need to be carefully chosen so as to reach high-level VM steady-state availability. In Wang et al. (2007) and Xie et al. (2004) time-based rejuvenation policies under varying workload have been developed to enhance the performability measure of cluster system. A comparative study of three different rejuvenation policies is achieved through analytical models using deterministic and stochastic Petri net (DSPN) models. VM consolidation techniques (Takeda and Takemura (2010), Ye et al. (2010)) and energy management policies (Feller et al. (2010)) are investigated to enable cloud computing energy saving. Authors in James and Ian (2013) introduce an energy-efficient technique for private cloud, namely workload mixes, for assigning tasks to servers to balance application performance and energy consumption. Gaganpreet and Anil (2015) propose a performance evaluation of power aware VM consolidation using live-migration. The objective is to assign workload to servers according to a function of their cost to operate. Ghosh et al. (2011) developed a scalable analytical model to quantify power-performance trade-off. The model enables to judiciously configure physical machine pools.

Recovery methods in IaaS context are investigated by Beloglazov et al. (2011) through a Markov chain model and a control algorithm for recognizing overloading problems during dynamic VM consolidation and to achieve load balancing through migration if needed. Recovery methods are also used by Bobroff et al. (2007) to avoid SLA violations. Also, power-performance tradeoff has been investigated by Escheikh et al. (2014) and Escheikh et al. (2017) through modeling a workload-aware PM mechanism for SVS subject to Cold-VM. Authors in Grottke and Schleich (2013) investigate the optimal energy requirement of a computing cluster in a proportional data center to maintain a specific workload and performance objectives. They show how optimizing switching between running and hibernating servers' states to fulfill a selected service level objective (SLO) with minimum energy overhead. Work in Araujo et al. (2014) investigates software aging impact in the Eucalyptus framework. It highlights potential harmful issues related to system dependability and performance, involving RAM memory, swap space exhaustion as well as highly excessive CPU utilization by the VM. In the same work, authors propose a predictive approach based on time series analysis to schedule rejuvenation, so as to reduce downtime. They show therefore that their approach outperforms in terms of availability the threshold-triggered rejuvenation method based on continuous monitoring of resources utilization. Authors in Kuehn et al. (2015) develop a versatile multi-server queuing model for data center servers enabling load-dependent server consolidation. They show through numerical results how to quantifying tradeoff between energy efficiency and QoS.

3 Performability analysis of SVSs

Performability is defined as a combination of system's performance and its dependability measure in the presence of faults. It can be thought of as the "quality of service (QOS), provided the system is correct" Meyer (1992). Dependability is an all-encompassing definition for reliability, availability, safety and security Laprie (1992). We focus in this paper on the combination of performance and availability measures of SVS.
In order to investigate SVSs performability analysis, it's recommended to consider several attributes such as PM, time-varying workload, dynamic speed scaling, software aging, failure, and rejuvenation. This enables better representation and understanding of different underlying potential interactions between all these attributes and to draw among them better performance-dependability trade-off. In the design and development processes of virtualized systems, many pertinent questions regarding time-varying workload impact on software aging, availability, performability, power-saving, and efficiency have to be considered. In order to achieve this purpose, performability techniques would be very useful to conceive appropriate models representing the desired behavior in a flexible, concise and modular way. Particularly modeling approach based on high level Petri nets formalism and their variants are among the most appropriate techniques. To tackle these issues SRNs formalism specifically involving non-Markovian distributions is commonly used. This enables to establish interesting dependability analysis, thanks to its ability to describe with relaxed assumptions more realistic models. In addition, using SRNs in virtualized system modeling would define and adopt other attributes fitting the best with green computing requirements. We propose in this paper a non-Markovian SRN-based modeling approach to investigate performability analysis of a SVS subject to VMM rejuvenation enabling elastic resource provisioning and relying on workload-aware PM mechanism. This approach describes and captures in a modular and concise way correlations and interactions between the following entities: *(i)* time-varying workload of bursty nature in SVS environments; *(ii)* VMM and VM subject to workload-dependent aging, failure, and rejuvenation; *(iii)* workload-aware PM scheme; *(iv)* on-demand service provisioning with dynamic speed scaling. The key contributions of this work are modeling: *(i)* how VMM or VM unavailability due to failure or rejuvenation affects service provisioning; *(ii)* PMC with multiple power states (having idle states (C-states) and operational states (P-states)) with dynamic and scalable resource provisioning; *(iii)* how workload tracking-based PMC scheme governs opportunistic transitions between PMC power states (P-states and C-states) to maximize efficiency. In what follows we detail the main performability attributes of SVS namely workload, fault tolerance, PMC, and workload-aware PM mechanism.

3.1 Workload in SVSs

Intensive and/or delay sensitive applications handled by SVSs are usually on demand and require usually dynamic provisioning. To meet stringent requirements of such applications, there is a need to dynamically adapt and manage virtualized system capabilities (scaling up/down CPU processing rates, switching opportunistically between CPU power states). These requirements are usually expressed and specified in SLA form in terms of throughput, waiting time, response time, quality of service and power efficiency. From a modeling perspective, it's highly recommended to use versatile processes such as Markov Modulated Poisson Process (MMPP) to capture time-varying nature of SVS handled workload and to illustrate dynamic speed scaling of resource provisioning. This will be more detailed

in section 4. In the rest of this paper, we consider a SVS belonging to hypervisor-based configuration category (Fig. 1). Hypervisor, called also VMM, is a software implementation of server virtualization hosting guest operating systems (i.e VM).

3.2 Software rejuvenation in SVSs

To cope with software aging effects on VM and VMM, preventive maintenance (a.k.a. proactive fault management) techniques referred to as rejuvenation are used. Such techniques actually reduce unpredictable outage of cloud applications, postpone crash failures and improve software performance but at the expense of occasional and controlled unavailability. VMM software aging problem directly affects all hosted VMs. These latter need to be controlled since their execution environment is likely to be cleared just before VMM rejuvenation. Three time-based VMM rejuvenation techniques are considered and are distinguished in the way the hosted VMs are handled before triggering VMM rejuvenation: *(i) Cold-VM rejuvenation*: which shuts down all VMs and cleans all aging effects of both VMs and VMM; *(ii) Warm-VM rejuvenation*: which suspends all VMs and saves their executions states to avoid transaction running loss. As soon as VMM rejuvenation is achieved, VMs' executions are resumed; *(iii) Migrate-VM rejuvenation*: moves all VMs to other host server. We focus in the rest of this paper only on Cold-VM and Migrate-VM rejuvenations.

3.3 Fault tolerance in SVSs

When running modern applications (i.e. data intensive or delay sensitive applications) in highly dynamic computing environments such as virtualized datacenters characterized by a highly varying on-demand traffic, several disruptive events and undesirable conditions may occur. Among these events or incidents, we distinguish software aging manifested in different forms (errors, faults, ...) and becoming more pronounced as time progresses and workload becomes higher. The more solicited virtual software resources become the higher memory leakage risk. Memory leak concerns in computing science incorrect memory allocation management where memory is not released even if it's no longer needed. This may involve potential failures leading obviously to SVS unavailability. To cope with these problems preventive maintenance (a.k.a. proactive fault management) techniques referred to as rejuvenation are used to postpone or prevent system failures. Given that VMM hosts VM, failure or rejuvenation of the former affects accordingly the latter. Such techniques actually postpone crash failures and improve software performance but at the expense of occasional and controlled unavailability. For SVS, software aging affects potentially both VMM and VM and any relevant performability analysis should take into consideration such key factors.

In the rest of this section we present some attributes and characteristics related to PMC, next we detail principles of workload-aware PM mechanism in SVSs.

3.4 PMC in SVSs

The main PMC feature in SVSs is to allow multiple operation modes to achieve workload-aware power-performance trade-off. Actually having several operation modes increases flexibility and enables finer control. It may incur however additional costs in terms of delay or performance loss. In practice, the number of operation modes is limited. To that end, and

in order to reduce operating cost and to rationalize energy consumption, ACPI Hewlett-Packard Corporation et al. (2011) industry standard has defined performance and power component states. The main purpose is to allow operating systems to fully monitor and control aspects related to power saving of their underlying hardware. In the following, let us detail the above PMC power states.

- P-states: are referred to as performance states and denoted as (P_0,P_1, ..., P_n). They determine capabilities of a working (i.e. active /loaded) CPU to save power. It should be pointed out that the number of P-states is component specific. Switching from one P-state to another is enabled through scaling both frequency and voltage. The higher P-state number, the lower frequency. Therefore power consumption decreases with respect to P-state number's increase.

- C-states: they determine idle component capabilities to switch off unused components to avoid power wastage. C-states are split into active state (C_0) and sleep states (C_1, C_2, ..., C_n). The higher C-state number, the deeper the sleep state. Notice that deeper sleep states induce slower wake up times. Notice also that several components hold multiple sleep states. Components such as RAM, hard disk or CPU may support active, ready and standby states. PM of hardware component relies on several hardware states. This should preserve system performance while keeping relative low cost by minimizing energy consumption through opportunistic selection of suitable power states.

In order to achieve workload-aware PM for virtualized systems, related components should dynamically adapt their power states by switching from a given state to a more convenient state (among all possible states) in a timely manner. Timing of switching from one state to another should be conveniently chosen, otherwise it would involve unsuitable (premature or too late) transitions leading to unavoidably energy wastage and /or performance degradation. In this paper we consider that PMC has three kinds of states detailed as follows:

- Active states: They include wakeup, fastactive and slowactive states. Wakeup state describes a PMC transient state between sleep state and fastactive or slowactive state. Fastactive state is activated whenever workload exceeds a given threshold and slowactive state becomes active everytime workload falls below a specific threshold.

- Sleep states: They include a sleep state and an idle state. Sleep state is considered the least power consuming state but experiencing the longest delay. Idle state consumes more power, than sleep state, but needs shorter delay to transit to one of the active states.

- Off states: They include only one state referred to as off state which consumes no power.

Hence PMC's power states maybe classified in an ascending order of energy consumption as follows; off, sleep, idle, wakeup, slowactive and fastactive.

3.4.1 Workload-aware PM mechanisms in SVSs

For SVSs, different workload-aware PM schemes maybe adopted to cope with on-demand traffic of data intensive as well as latency-sensitive applications Beloglazov et al. (2010). Such applications often generate highly time-varying workload incurring several non-uniform unpredictable fluctuations and including workload burst periods of variable lengths.

From PM point of view, appropriate control schemes should be adopted in order to meet stringent SLA requirements. A carefully designed PM scheme should rely on coordinated solutions combining multiple policies to handle multi-objective adaptations. Adopting a PM policy (or a logic) enables a clear description of advantages, related costs and overheads of each adaptation alternative. This will be used to make the good decision at the right time about the right resources in a dynamic and flexible manner. Therefore, each adaptive alternative should be evaluated before making a decision but also reevaluated at each adaptation period. This could be achieved by building self-tuning schemes enabling self-configuring properties through adapting dynamically PMC power states according to workload variations without causing unbearable overheads. These overheads may result from several reasons often related to satisfying at the same time several conflicting needs in terms of efficiency, correctness and stability while complying with constraints.
Workload-aware PM encompasses DPM and dynamic voltage frequency scaling (DVFS) policies. DPM tries to fully exploit knowledge about real-time resource usage and applications' nature to minimize energy consumption. It includes timeout, predictive and stochastic PM policies and is usually used for system components having multiple power states. Whereas DVFS is convenient for components supporting multiple speed and voltage levels dynamically scaled up or down.
In timeout PM scheme, PMC is forced to sleep state whenever it's idle for more than a specified period of time (referred to as timeout). Note that this period should be at least equal to break-even time which corresponds to the minimum necessary time PMC has to hold in the low-power state to recover switching costs to and from low-power state. Thus, it is preconized to filter out relatively short idle periods. The more PMC responds faster the smaller filter intervals are. Authors in Simunic et al. (2000) found that the most appropriate range of filter intervals for hard disk is included in [0.5 s, 2 s], while for WLAN card filter intervals are considerably shorter (50 ms, 200 ms) since these devices respond much faster than the hard disk. Unlike timeout policies, predictive policies rely on predicting upcoming idle period duration and take shutdown decision straight after PMC becomes idle. Stochastic PM policy is a DPM policy where workload and PMC behavior are described by stochastic processes and where PM maybe formulated as a power-performance optimization problem. PMCs such as CPU may support the two policies cited above (DPM, DVFS). This enables virtualized resources to meet QoS requirements of handled applications while rationalizing power resource utilization. In order to face changing scenarios and to fully benefit from available resources, applications should include self-configuring capabilities. This enables to take appropriate decisions squeezing most out of competing adaptation mechanisms and avoiding potential oscillations whenever two or more correlated objectives need to be reached simultaneously. Design of dynamic adaptive PM mechanisms suitable for SVSs context needs to capture correlations between PMC behavior and workload's dynamic via appropriate modeling approach.

3.5 Correlations between workload, software aging, PM policy and power consumption in SVSs

Software aging is commonly accelerated with time and with respect to workload increase. This should be captured in our modeling approach by considering workload-dependent software aging. Also energy efficient PM policy should minimize energy consumption by judiciously adapting PMC power states with respect to workload variations. This may be accounted in our proposed models by considering a time-based workload-aware PM

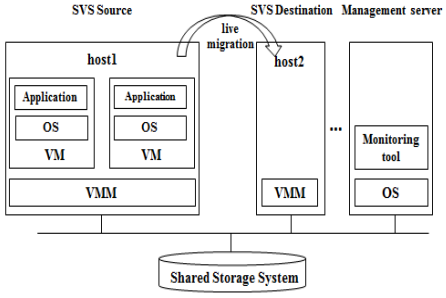


Figure 1: System configuration including a server virtualized system

policy, The more this minimization is tangible and effective the less aging is pronounced. This demonstrates also close correlation between on one hand this kind of PM policy and workload and on the other hand PM policy and workload-dependent aging. Given that PM policy aims to switch adaptively from one PMC power state to another, its impact on software aging is almost impalpable for both light and heavy workloads. Indeed in the first case PMC is almost always in sleep state whereas in the second case PMC is almost always in active state and therefore in both cases dynamic workload aware PM policy would not provide significant gains.

4 SRNs modeling of versatile SVS with workload-aware PM mechanism

From a point of view combining performance and dependability, VM needs to assign dynamically virtualized resources such as vCPU, vRAM, vDDR, in an adaptive way taking into consideration impairments cited above and aiming to meet QoS requirements while preserving low power costs. These resources should be managed according to efficient pre-established policy in order to achieve convenient processing.

In this section, we first recall SRN paradigm. We give next a detailed description of two proposed versatile SVS SRN models. The first model accounts for Cold-VM whereas the second takes into consideration Migrate-VM. We examine also tunable parameters in the two SVS models and we detail the main contributions in this paper.

4.1 SRN basic principles

Petri nets (PNs) represent a high level formalism providing the underlying Markov model specification and a visualization of system's dynamic Tadao Murata (1989). They give easier representation of computer system features such as concurrency, synchronization, sequencing and resource sharing through a set of places (represented by circles) and transitions (represented by rectangles). Each place represents a system local state and may contain zero or more tokens (represented by black dots or integers). Each transition represents a system event and is connected to places via oriented arcs. A transition is enabled if at least one token is deposited in each input place. The firing of an enabled transition removes a token from each of its input place(s) and puts one token in each of its output place(s). The number of tokens in each place (a.k.a. marking) characterizes a PN state. We distinguish two kinds of marking: vanishing and tangible. Tangible marking maybe assigned a reward rate. A transition with an assigned marking dependent guard is

enabled only if this guard is true. Notice also that transitions maybe immediate or timed. In this last case, transition firing time is a random variable. The use of non-Markovian transitions distributions such as deterministic or general, allows building more realistic models. Transitions which are never preempted are referred to persistent. However those which may be preempted are known as non-persistent. For non-Markovian non-persistent transitions, a firing policy has to be specified. The first kind of policies is known as preemptive resume (prs) whereas the second one is referred to preemptive different (prd). *prs* (resp. *prd*) policy means that when a transition is disabled without having fired, its performed work is maintained (resp. lost).
A more concise description of the stochastic PN (SPN) model may be obtained by using several extensions such as marking dependent arc multiplicity, priorities and guards (i.e. enabling function) with firing dependent firing rates (also called reward rates and formulated with respect to the number of tokens in place P (i.e. $\#P$)). Also associating reward description to PNs extends SPNs and is referred to as SRNs. The modeling paradigm based SRNs is recognized to be among the most powerful tools to investigate model-based performability analysis combining performance and dependability evaluations. SRNs facilitate both reward structure specification and Markov reward model automatic generation. They allow to specify and calculate a variety of qualitative and quantitative performance measures. SRNs enhance both modeling power and modeling convenience and specify output measures as a reward-based function. In SRNs a tangible marking may be associated with a reward rate so as to facilitate the computation of several output measures. Classical output measures namely mean (expected) number of tokens in a place, expected transition throughput and probability that an event occurs may be derived from a SRN. Also, reward functions may be used to define more complex output measures.

Output measures in SRNs: Assume X a random variable representing a reward rate and describing a given output measure of interest. In steady-state the expected reward rate expression is given by:

$$E[X] = \sum_{k \in T} n_k r_k$$

where T is the set of tangible markings, n_k (resp. r_k) is the steady-state probability (resp. reward rate) of tangible marking k.

4.2 Description of the proposed versatile SVS SRN models

We propose in this paper two versatile SVS SRN models representing SVS composed by one VMM hosting a VM which in turn handles data intensive applications with time-varying traffic. The first versatile SVS SRN model is subject to Cold-VM rejuvenation whereas the second one uses migrate-VM (i.e. live migration) as VMM rejuvenation technique. The two above models include a workload-aware PM mechanism enabling to judiciously adapt PMC's soft power states (a.k.a. P-states, C-states) in accordance to workload variations. In our modeling approach we assume that VM and VMM are subject during their operational processes to several impairments such as workload-dependent software aging, failure and rejuvenation. We assume likewise that VMM assigns dynamically to VM a PMC (i.e. virtualized resources such as vCPU) monitored by workload-aware PM policy. For illustrative purposes instead of using monolithic model we adopt a modular approach to better describe interactions between different sub-models of each proposed versatile SVS SRN model. This is justified by the need to better understanding complex SVS model

involving several attributes. For more convenience, we will use in the rest of this paper the following notations:

- The first (resp. second) versatile SVS SRN model subject to Cold-VM (resp. Migrate-VM) rejuvenation is referred to as model^1 (resp. model^2).
- Each of theses models encompasses two dependent sub-models. The first one describes VMM and VM interactions in SVS, whereas the second sub-model represents a workload-aware PM mechanism.
- Model^1 is composed by sub-model^{11} (Fig. 2, Tab. 1, Tab. 3, Tab. 6) and sub-model^{12} (Fig. 4, Tab. 2, Tab. 4, Tab. 7).
- Model^2 includes sub-model^{21} (Fig. 3, Tab. 1, Tab. 3, Tab. 6) and sub-model^{22} (Fig. 4, Tab. 2, Tab. 5, Tab. 7).
- Notice that sub-model^{11} (resp. sub-model^{21}) is similar to Cold-VM (resp. Migrate-VM) model given in Machida et al. (2010) and is used in this paper as a sub-model of model^1 (resp. model^2) with some guards modifications. These modifications are introduced in order to represent workload-dependent software aging (in VMM and VM) and to retain correlations with sub-model^{12} (resp. sub-model^{22}).

In the rest of this subsection, we provide a detailed description of model^1 and model^2. Notice that we adopt the same notation for naming transitions and places. For example T_vfp (resp. T_hfp) name abbreviates fail-prone transition of VM (resp. VMM)). We next investigate tunable parameters in the workload-aware PM SRN sub-model before defining some pertinent performability metrics of the two proposed versatile SVS SRN models.

4.2.1 SVS SRN sub-model with Cold-VM rejuvenation (sub-model^{11})

Sub-model^{11} (Fig. 2) encompasses a VMM model and a VM model. VMM model includes a VMM sub-model (Fig. 2a) and a VMM clock sub-model (Fig. 2b). Similarly VM model includes a VM sub-model (Fig. 2c) and a VM clock sub-model (Fig. 2d). VMM (resp. VM) clock sub-model is used to trigger VMM (resp. VM) time-based rejuvenation. The VM model is closely dependent of the VMM model. This is straightforward since failure or rejuvenation of VMM affects automatically the handled VM. Related dependencies between VM and VMM are illustrated through assigned guards in VM model. In what follows we recall detailed description given in Machida et al. (2010) for sub-model^{11}. We highlight also potential interactions between VMM and VM models.

VMM model description: As soon as the latest VMM boot time expires, deterministic transition $T\_hinterval$ with constant duration $1/\tau_h$ fires to trigger VMM rejuvenation as long as immediate transition $T\_hpolicy$ is enabled. Fig. 2a represents and describes failure, recovery and time-based rejuvenation processes in VMM. Whenever VMM software aging transition T_hfp fires, it deposits a token in P_hfp. T_hfail firing assigns a token to P_hfail. The above place indicates the state of VMM failure caused by software aging. T_hdet firing is conditioned by VMM failure detection whereas $T\_hrepair$ firing is only enabled if VMM recovery process (from failure state) is fully achieved. VMM clock (Fig. 2b) enables to trigger VMM rejuvenation phase and as soon as this event occurs, immediate transition T_hrejt or $T\_hfprejt$ can fire and moves a token from P_hfp to deposit another one in P_hrej. As soon as VMM rejuvenation process achieved, transition

T_hrej fires and deposits a token in P_hup.

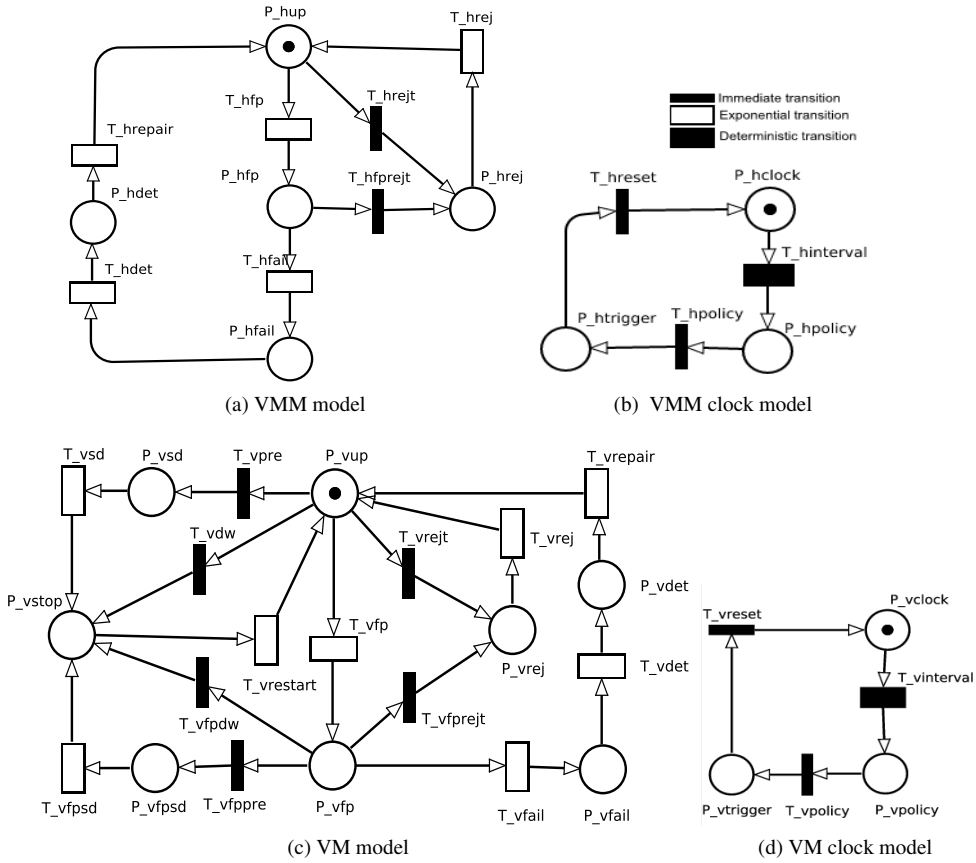
VM model description: $T\_vinterval$ (Fig. 2d)fires, after the latest VM start up time, and triggers VM rejuvenation whenever immediate transition $T\_vpolicy$ is enabled. $T\_vinterval$ duration (i.e. $1/\tau_v$) is considered constant and is represented by a deterministic transition. As soon as VM rejuvenation process finishes, T_vreset is enabled and puts a token in place P_vclock. Notice that VM model (Fig. 2c) includes different stages of failure, recovery and time-based rejuvenation. Notice also that since VM model closely depends on the underlying VMM model, whenever no token is present in P_hup nor in P_hfp, both immediate transitions T_vdw and T_vfpdw are enabled and a token is put in P_vstop. Getting a token in P_hup or P_fup allows transition $T\_vrestart$ to be enabled and consequently VM cannot be rebooted unless VMM is available again. Cold-VM rejuvenation consists in shutting down hosted VM before starting any VMM rejuvenation. This is expressed by immediate transitions T_vpre and T_vfppre and their associated guards' functions. Te above transitions are enabled as soon as a token is deposited in place $P\_hpolicy$ (Fig. 2b). Whenever T_vpre (resp. T_vfppre) fires a token is deposited in P_vsd (resp. P_vfpsd) and as a consequence transition T_vsd (resp. T_vfpsd) is enabled. In VM model (Fig. 2c), firing either T_vsd or T_vfpsd allows to deposit a token in P_vstop and to enable then immediate transition $T\_hpolicy$ which in turn triggers VMM rejuvenation process (Fig. 2b). Notice that VMM rejuvenation starting is assumed conditioned by the presence of a token in one of places, P_vup, P_vfp, P_vsd and P_vfpsd. Notice also that $T\_vinterval$ (resp. $T\_hinterval$) distribution is considered deterministic since it's used to model fixed VM (resp. VMM) rejuvenation trigger interval. All others timed transitions in Fig. 2 are assumed exponentially distributed.

4.2.2 Description of SVS SRN model with Migrate-VM (sub-model^{21})

Sub-model^{21} (Fig. 3, Tab. 1, Tab. 3 and Tab. 6) consists of VM sub-model, VM clock sub-model, VMM sub-model and VMM clock sub-model. Notice that representation, associated notations and guards of sub-models (Fig. 3a, Fig. 3b and Fig. 3d) included in model^{21}, are respectively exactly identical to those given in Fig. 2a, Fig. 2b and Fig. 3d (described above in subsection 4.2.1) $T\_hpolicy$ guard function (Tab. 3). Fig. 3c shows sub-model of a VM housed by VMM subject to Migrate-VM rejuvenation.
Migrate-VM rejuvenation uses live VM migration to move a running VM from the current VMM, just before triggering its rejuvenation, to a remote VMM. If a token is deposited in P_vup or P_vfp and immediate transitions T_vpre and T_vfppre are enabled, VM migration process to remote host is triggered. T_vpre and T_vfppre are enabled when a token is deposited in $P\_hpolicy$ (Fig. 3b). While a token is present in P_vmigd or $P\_vfpmigd$, VM execution on remote host will continue. As soon as VMM rejuvenation is completely achieved, immediate transitions T_vpost and $T\_vfppost$ will be enabled and VM returns back from remote host to original host, again by live VM migration. To keep SVS SRN model more tractable, we assume that no failure occurs throughout VMM rejuvenation and VM migration periods. VMM rejuvenation can be triggered unless a token is deposited in P_vup, P_vfp, P_vmig, P_vfpmig, P_vbac or P_vfpbac. If a token is deposited in P_vup, P_vfp, P_vmigd, or $P\_vfpmigd$, then VM becomes available and transition $T\_vinterval$ is enabled.

It's worth mentioning that authors in Machida et al. (2010) used 10-stage Erlang distributions to approximate deterministic distributions since model implementation is

done with software package SPNP (Muppala et al. (1994)) which uses only exponential timed distributions. Since in this paper we use extended deterministic stochastic Petri nets (eDSPNs) implemented through TimeNet 4.1 tool (Armin Zimmermann (2012)) above approximation is relaxed and deterministic distributions are effectively used.



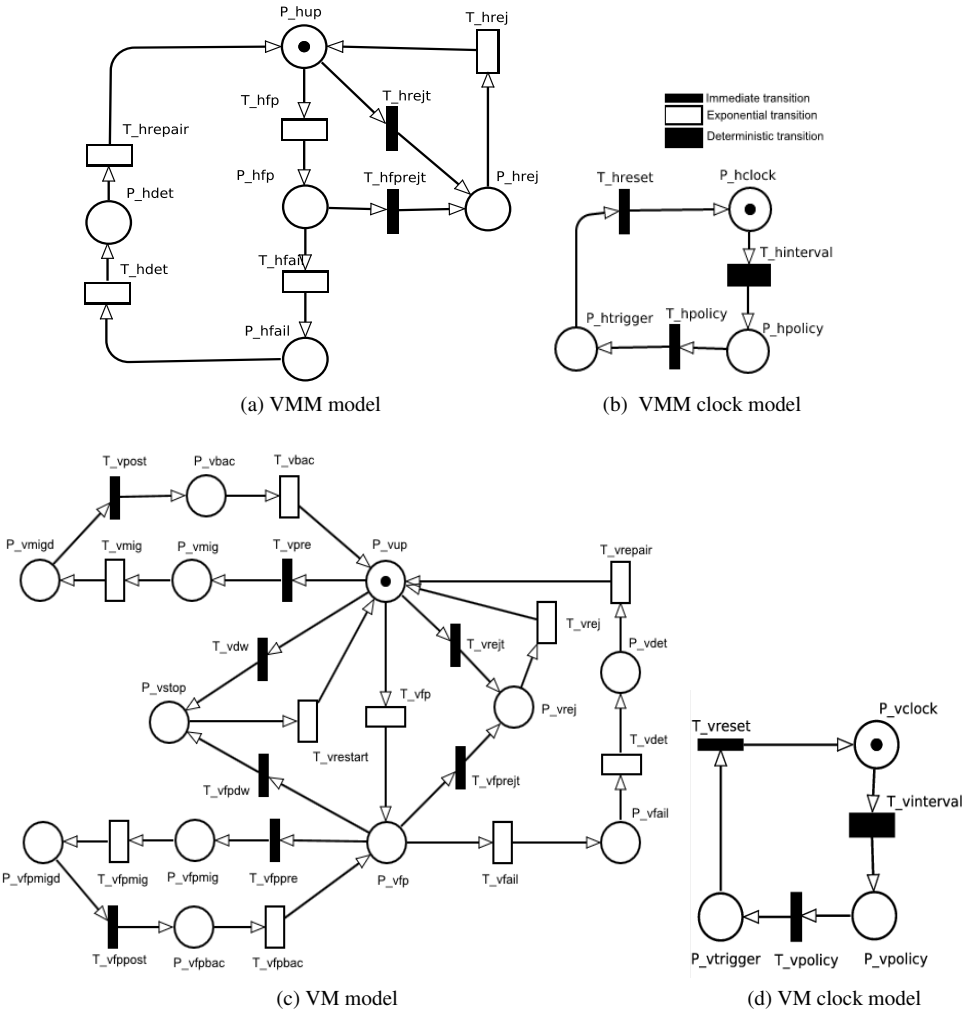
(a) VMM model (b) VMM clock model

(c) VM model (d) VM clock model

Figure 2: SVS SRN model with Cold-VM rejuvenation (sub-model^{11})

4.2.3 Detailed description of workload-aware PM SRN sub-model (sub-model^{12} and sub-model^{22})

Sub-model^{12} and sub-model^{22} have the same representation (Fig. 4, Tab. 2, Tab. 7) but they have different guards (Tab. 4, Tab. 5). Control scheme implementing PM mechanism is conceived to achieve dynamic adaptive PM policy relying on workload tracking process. Tracking process aims to make appropriate opportunistic workload-aware decisions enabling to wisely choose how and when to force PMC to transit from one power state to another. This is achieved by choosing appropriate PMC power states accordingly. Application-driven PM policies for unpredictable time varying workload are commonly of stochastic and non stationary nature and change over time according to workload variations.



(a) VMM model (b) VMM clock model

(c) VM model (d) VM clock model

Figure 3: SVS SRN model with Migrate-VM rejuvenation(sub-model^{21})

This means that workload fluctuations may lead to PMC state's change at any point in time. This is usually achieved in order to maintain low power costs while meeting SLA performance requirements. In what follows we give a detailed description of the proposed workload-aware PM SRN sub-model (Fig. 4, Tab. 2, Tab. 7) including a novel event driven PM mechanism for unpredictable time varying workload referred to as *On-demand transition to active state with timeout preempted upon arrival*. From a practical point of view PM policy makes decisions upon a request's occurrence or straight after a request processing is achieved. The description of the workload-aware PM SRN sub-model (Fig. 4, Tab. 2, Tab. 7) includes preliminary assumptions, request processing process and P-invariants specification.

Assumptions

Table 1 Transitions description of sub-model^{11} (Fig. 2) and sub-model^{21} (Fig. 3)

Common description for the two sub-models	
Transition	**Description**
$T\_hinterval$ (resp. $T\_vinterval$)	host (i.e. VMM) (resp.VM) inter-rejuvenation interval duration
$T\_hpolicy$ (resp.$T\_vpolicy$)	host (resp. VM) rejuvenation trigger interval duration
T_hreset (resp. T_vreset)	immediate transitions enabled when host (resp. VM) rejuvenation duration expires to trigger $T\_hinterval$ (resp. $T\_vinterval$)
T_hfp (resp. T_vfp)	host (resp. VM) fail prone duration
T_hfail (T_vfail)	host (resp. VM)) failure duration
T_hdet (resp. T_vdet)	host (resp. VM)) failure detection duration
$T\_hrepair$ (resp. $T\_vrepair$)	host (resp. VM) failure recovery duration
T_hrejt (resp. T_vrejt)	immediate transitions enabled to trigger host (resp. VM) rejuvenation
T_hrej (resp.T_vrej)	host (resp. VM) rejuvenation duration
$T\_hfprejt$ (resp.$T\_vfprejt$)	immediate transitions enabled to trigger host (resp. VM) rejuvenation
$T\_vrestart$	VM restart duration
T_vdw, T_vfpdw	VM shutdown duration
Specific description for each model	
Cold-VM (sub-model^{11})	
Transition	**Description**
T_vpre, T_vfppre	immediate transitions enabled when VM stopping process is initiated (as soon as host rejuvenation is achieved)
T_vsd, T_vfpsd	VM stopping time interval duration
Migrate-VM (sub-model^{21})	
Transition	**Description**
T_vpre, T_vfppre	immediate transitions enabled when VM Migration process to remote host is initiated (as soon as host rejuvenation is achieved)
T_vmig, T_vfpmig	VM migration duration
$T\_vpost, T\_vfppost$	immediate transitions triggering VM migration back to original host
T_vbac, T_vfpbac	duration VM migration back to original host

Table 2 Transitions description of workload-aware PM SRN sub-model (Fig. 4)

Transition	**Description**
T_h2l	time interval duration to switch MMPP traffic from high to low rate
T_l2h	time interval duration to switch MMPP traffic from low to high rate
$1/T\_hightraffic$	high traffic rate
$1/T\_lowtraffic$	low traffic rate
$1/T\_servfast$	fast service processing rate
$1/T\_servslow$	slow service processing rate
$1/T\_f2s$	transition rate to switch from fast to slow
$1/T\_s2f$	transition rate to switch from slow to fast
$1/T\_fast$	PMC fast service processing rate
$1/T\_slow$	PMC slow service processing rate
T_start	immediate transition enabled to trigger service processing
$T\_sleep2wakeup$	time interval duration for PMC to transit from sleep to wakeup
T_wakeup	time interval duration for PMC to transit from wakeup to idle
$T\_timeout$	time interval duration for PMC to transit from idle to sleep
$T\_lossjob$	immediate transition enabled when an arriving request is lost
$T\_rejdown1$	immediate transition enabled to transit from sleep to off
$T\_rejdown2$	immediate transition enabled to transit from idle to off
$T\_rejdown3$	immediate transition enabled to transit from fastactive to off
$T\_rejdown4$	immediate transition enabled to transit from slowactive to off
$T\_rejdown5$	immediate transition enabled to transit from wakeup to off
$T\_notrejdown$	time duration to transit from off state to sleep state

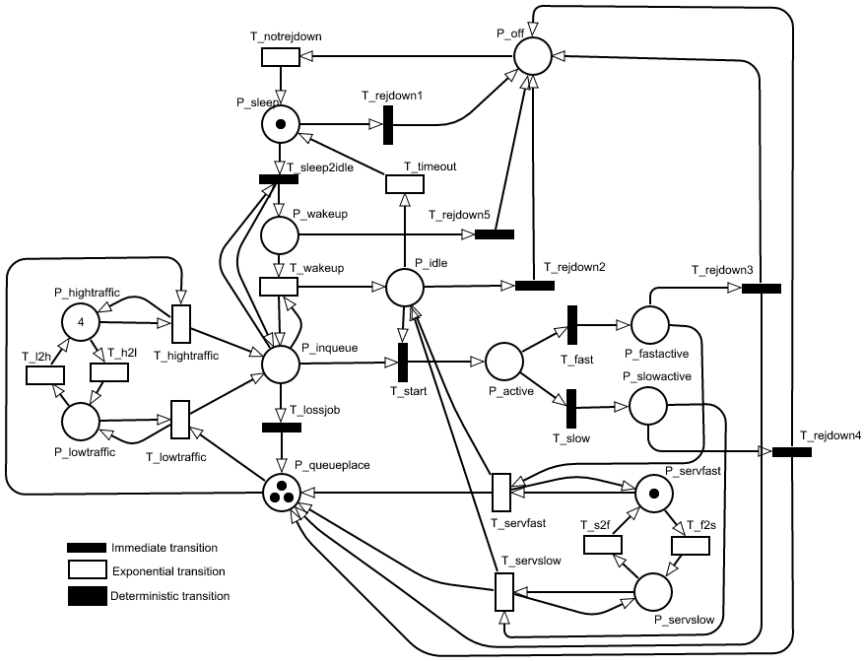


Figure 4: Workload-aware PM SRN sub-model

Assumptions considered in this paper are mainly related to initial marking, arrival process distribution, service process distribution and PMC transitions from sleep to wakeup. We particularly assume that:

- transitions distributions are either immediate (imm.), exponential (exp.) or deterministic (det.);
- PMC has the following power states: off state (P_off), active or operational states ($P\_fastactive$, $P\_slowactive$) and idle states (P_idle, P_wakeup ,P_sleep). The lower power state the more time it experiences to come back to active state;
- PMC is initially in sleep state (P_sleep);
- SVS is initially idle (#$P\_queueplace$ = K = 3, where K is buffer capacity of the SVS model);
- PMC service processing follows a two states MMPP, this allows to capture dynamic speed scaling of PMC;
- PMC service processing is workload-dependent;
- request arrival follows a two states MMPP, this enables to capture traffic burstiness;
- there are 4 traffic sources (X high traffic sources and Y low traffic sources with $X + Y = 4$);
- initially there are 4 low traffic sources;

- PMC is initially in sleep state. According to this assumption and as a request arrival occurs, PMC service initiation (i.e. request processing) is not granted at once. Instead,

Table 3 Transitions with assigned guards of SVS SRN sub-models (sub-model^{11}, sub-model^{21})

Common guards for sub-model^{11} and sub-model^{21}	
Transitions	**Guards**
T_hrej	*P_hclock* == 1
T_hinterval	(*P_hup* == 1) || (*P_hfp* == 1)
T_hrejt	*P_htrigger* == 1
T_hfprejt	*P_htrigger* == 1
T_hreset	*P_hrej* == 1
T_vpolicy	(*P_vup* == 1) || (*P_vfp* == 1)
T_vdet	(*P_hup* == 1) || (*P_hfp* == 1)
T_vrepair	(*P_hup* == 1) || (*P_hfp* == 1)
T_vrej	(*P_vclock* == 1) && (*P_hup* == 1) || (*P_hfp* == 1)
T_vrejt	*P_vtrigger* == 1
T_vfprejt	*P_vtrigger* == 1
T_vpre	*P_hpolicy* == 1
T_vdw	*P_hfail* == 1
T_vfpdw	*P_hfail* == 1
T_vfppre	*P_hpolicy* == 1
T_vrestart	(*P_hup* == 1) || (*P_hfp* == 1)
T_vreset	*P_vrej* == 1
Specific guards to each model	
Cold-VM (sub-model^{11})	
Transitions	**Guards**
T_hpolicy	(*P_vstop* == 1) || (*P_vfail* == 1) || (*P_vdet* == 1) || (*P_vrej* == 1)
T_vinterval	(*P_vup* == 1) || (*P_vfp* == 1) || (*P_vsd* == 1) || (*P_vfpsd* == 1)
Migrate-VM (sub-model^{21})	
Transitions	**Guards**
T_hpolicy	(*P_vfail* == 1) || (*P_vdet* == 1) || (*P_vrej* == 1) || (*P_vmigd*== *1*)|| (*P_vfpmigd* == 1)
T_vinterval	(*P_vup* == 1) || (*P_vfp* == 1) || (*P_vmigd* == 1) || (*P_vfpmigd* == 1)

Table 4 Transitions with assigned guards of workload-aware PM SRN sub-model (Cold-VM rejuvenation)

Transitions	**Guards**
$T\_servfast$	(*P_vup* == 1) || (*P_vfp* == 1) || (*P_vsd* == 1) || (*P_vfpsd* == 1)
$T\_servslow$	(*P_vup* == 1) || (*P_vfp* == 1) || (*P_vsd* == 1) || (*P_vfpsd* == 1)
T_f2s	(*P_hightraffic* == 0) && (*P_vup* == 1) || (*P_vfp* == 1) || (*P_vsd* == 1) || (*P_vfpsd* == 1)
T_s2f	(*P_hightraffic* > 0) && (*P_vup* == 1) || (*P_vfp* == 1) || (*P_vsd* == 1) || (*P_vfpsd* == 1)
T_fast	(*P_servfast* == 1)
T_slow	(*P_servslow* == 1)
T_start	(*P_vup* == 1) || (*P_vfp* == 1) || (*P_vsd* == 1) || (*P_vfpsd* == 1)
T_sleep2wakeup	(*P_vup* == 1) || (*P_vfp* == 1) || (*P_vsd* == 1) || (*P_vfpsd* == 1)
T_wakeup	(*P_vup* == 1) || (*P_vfp* == 1) || (*P_vsd* == 1) || (*P_vfpsd* == 1)
$T\_timeout$	(*P_vup* == 1) || (*P_vfp* == 1) || (*P_vsd* == 1) || (*P_vfpsd* == 1)
$T\_lossjob$	(*P_vup* == 0) && (*P_vfp* == 0) && (*P_vsd* == 0) && (*P_vfpsd* == 0)
$T\_rejdown1$	(*P_vup* == 0) && (*P_vfp* == 0) && (*P_vsd* == 0) && (*P_vfpsd* == 0)
$T\_rejdown2$	(*P_vup* == 0) && (*P_vfp* == 0) && (*P_vsd* == 0) && (*P_vfpsd* == 0)
$T\_rejdown3$	(*P_vup* == 0) && (*P_vfp* == 0) && (*P_vsd* == 0) && (*P_vfpsd* == 0)
$T\_rejdown4$	(*P_vup* == 0) && (*P_vfp* == 0) && (*P_vsd* == 0) && (*P_vfpsd* == 0)
$T\_rejdown5$	(*P_vup* == 0) && (*P_vfp* == 0) && (*P_vsd* == 0) && (*P_vfpsd* == 0)
$T\_notrejdown$	(*P_up* == 1) || (*P_vfp* == 1) || (*P_vsd* == 1) || (*P_vfpsd* == 1)

it's deferred and triggered after a given delay has elapsed. This delay corresponds to a given time (T_wakeup) needed by PMC to switch from sleep state to active state;

- PMC power consumption along T_wakeup duration is maximum and equal to $Powerfastactive$.

Request processing process description

As soon as a request occurs (a token is deposited in $P\_inqueue$), and since there is initially a token in P_sleep, transition $T\_sleep2wakeup$ is enabled. Forthwith, a token is deposited in P_wakeup and transition T_wakeup is thereby enabled. Note that whenever T_wakeup is enabled and before its firing, further request arrivals may enter the system and as much tokens are deposited in $P\_inqueue$. The number of tokens in $P\_inqueue$ represents the number of requests waiting to be processed whereas $P\_queueplace$ represents the available

Table 5 Transitions and assigned guards of workload-aware PM SRN sub-model (Migrate-VM rejuvenation)

Transitions	**Guards**
T_servfast	(*P_vup* == 1) || (*P_vfp* == 1) || (*P_vmigd* == 1) || (*P_vfpmigd* == 1)
T_servslow	(*P_vup* == 1) || (*P_vfp* == 1) || (*P_vmigd* == 1) || (*P_vfpmigd* == 1)
T_f2s	(*P_hightraffic* == 0) && (*P_vup* == 1) || (*P_vfp* == 1) || (*P_vmigd* == 1) || (*P_vfpmigd* == 1)
T_s2f	(*P_hightraffic* > 0) && (*P_vup* == 1) || (*P_vfp* == 1) || (*P_vmigd* == 1) || (*P_vfpmigd* == 1)
T_fast	(*P_servfast* == 1)
T_slow	(*P_servslow* == 1)
T_start	(*P_vup* == 1) || (*P_vfp* == 1) || (*P_vmigd* == 1) || (*P_vfpmigd* == 1)
T_sleep2wakeup	(*P_vup* == 1) || (*P_vfp* == 1) || (*P_vmigd* == 1) || (*P_vfpmigd* == 1)
T_wakeup	(*P_vup* == 1) || (*P_vfp* == 1) || (*P_vmigd* == 1) || (*P_vfpmigd* == 1)
T_timeout	(*P_vup* == 1) || (*P_vfp* == 1) || (*P_vmigd* == 1) || (*P_vfpmigd* == 1)
T_lossjob	(*P_vup* == 0) && (*P_vfp* == 0) && (*P_vmigd* == 0) && (*P_vfpmigd* == 0)
T_rejdown1	(*P_vup* == 0) && (*P_vfp* == 0) && (*P_vmigd* == 0) && (*P_vfpmigd* == 0)
T_rejdown2	(*P_vup* == 0) && (*P_vfp* == 0) && (*P_vmigd* == 0) && (*P_vfpmigd* == 0)
T_rejdown3	(*P_vup* == 0) && (*P_vfp* == 0) && (*P_vmigd* == 0) && (*P_vfpmigd* == 0)
T_rejdown4	(*P_vup* == 0) && (*P_vfp* == 0) && (*P_vmigd* == 0) && (*P_vfpmigd* == 0)
T_rejdown5	(*P_vup* == 0) && (*P_vfp* == 0) && (*P_vmigd* == 0) && (*P_vfpmigd* == 0)
T_notrejdown	(*P_vup* == 1) || (*P_vmigd* == 0) && (*P_vfpmigd* == 0)

Table 6 Parameters and corresponding values used of the two versatile SVS SRN models

Symbol	**Description**	**Value**	**Mean time**
λ_{FPv}	VM aging rate	0.005952381	1 week
λ_v	VM failure rate after aging	0.013888889	3 days
δ_v	VM failure detection rate	12	5 mins
μ_v	VM failure recovery rate	2	30 mins
β_v	VM rejuvenation rate	60	1 min
r_v	VM restart rate	120	30 secs
σ_v	VM shutdown rate	120	30 secs
ω_v	VM migration rate	3600	1 sec
τ_v	VM rejuvenation trigger rate	0.041666667	1 day
λ_{FPh}	VMM aging rate	0.001388889	1 month
λ_h	VMM failure rate after aging	0.005952381	1 week
σ_h	VMM failure detection rate	12	5 mins
μ_h	VMM failure recovery rate	1	1 hour
β_h	VMM rejuvenation rate	30	2 mins
τ_h	VMM rejuvenation trigger rate	0.005952381	1 week

system capacity.
Forthwith T_wakeup fires, PMC transits from wakeup state to idle state and a token is deposited in P_idle. This enables, straight after, the immediate transition T_start since at least a token is present in place $P\_inqueue$. Once T_start fires, it moves one token from each of places $P\_inqueue$ and P_idle and forwards a token to P_active. Here indeed the actual request processing begins. The dynamic speed scaling of PMC following a MMPP process is expressed by transitions $T\_servfast$ and $T\_servslow$ alternating between $P\_servfast$ and $P\_servslow$ (using transitions T_f2s and T_s2f). According to assigned guards one of the immediate transitions T_fast and T_slow is enabled. When T_fast (resp. T_slow) fires, it puts a token in $P\_fastactive$ (resp. in $P\_slowactive$). Adapting service processing rate to workload variations is achieved through marking dependent firing rates. This adaptation is enabled by assigning guard(s) to one or both of transitions T_s2f and T_f2s. Once request is fully processed, through firing of $T\_servfast$ or $T\_servslow$, PMC switches to idle state (P_idle) and triggers at the same time a timer with constant duration ($T\_timeout$). If any new request arrival occurs before timer expires then transition $T\_timeout$ is canceled (preempted with policy prd) and PMC comes back from idle state (P_idle) to active state ($P\_slowactive$ or $P\_fastactive$) after T_start firing. Therefore a new request processing begins immediately. Otherwise if this timer expires without detecting any new request arrival, then $T\_timeout$ fires and the PM mechanism forces PMC to transit immediately back to sleep state (P_sleep). Finally since our versatile SVS SRN models deal with performability analysis it's obvious to hold failure occurrences. Thus if at any time VM or VMM becomes unavailable due to either failure or rejuvenation, PMC power state switches to off state (P_off). This is captured in PM SRN model by immediate transitions $T\_rejdown1$, $T\_rejdown2$, $T\_rejdown3$, $T\_rejdown4$ and $T\_rejdown5$. These latter are enabled with the same assigned guard specifying that VM or VMM is unavailable. Indeed in such case, one of the above transitions fires, moves a token from one of the following places: P_sleep, P_idle, $P\_fastactive$, $P\_slowactive$, P_wakeup and immediately forwards a token in place P_off.
To summarize the above process let us consider time interval T_{int} between instant when

Table 7 Timed transitions' Values of workload-aware PM SRN sub-models

transition	**Value**
$T\_h2l = \frac{1}{\sigma_{T\_h2l}}$	0.1
$T\_l2h = \frac{1}{\sigma_{T\_l2h}}$	0.01
$T\_hightraffic = \frac{1}{\lambda_{T\_hightraffic}}$	[0.0015,...,250]
$T\_lowtraffic = \frac{1}{\lambda_{T\_lowtraffic}}$	[0.0003,...,50]
$T\_servfast = \frac{1}{\lambda_{Tservfast}}$	0.05
$T\_servslow = \frac{1}{\lambda_{Tservslow}}$	0.002
$T\_f2s = \frac{1}{\sigma_{T\_f2s}}$	0.1
$T\_s2f = \frac{1}{\sigma_{T\_s2f}}$	0.01
$T\_wakeup = \frac{1}{\lambda_{T\_wakeup}}$	0.01
$T\_timeout = \frac{1}{\lambda_{T\_timeout}}$	0.02

Table 8 PMC power measurements

Mode	$sleep$	$idle$	$slowactive$	$fastactive$	$wakeup$
Power (watt)	2.5	60	80	95	95

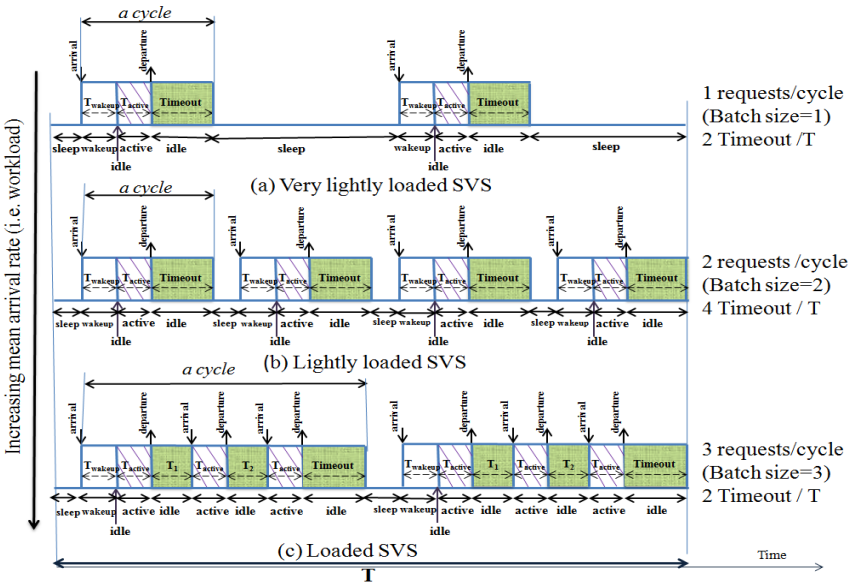


Figure 5: Illustration of PMC states' evolution and request processing vs time (for different mean arrival rates) in the SVS

previous request is fully processed and instant when current request occurs. If T_{int} is strictly shorter than a given time (corresponding to $T\_timeout$ delay) then as soon as the processing of the previous request is achieved, PMC immediately transits from active state ($P\_fastactive$ or $P\_slowactive$) to idle state (P_idle). PMC remains in P_idle until T_{int} time interval is elapsed and straight after it switches to active state to initiate current request processing. Inversely if T_{int} is greater than $T\_timeout$ duration and as soon as this latter is elapsed, PMC switches from idle to sleep state. In the following we will discuss what kind of transitions' delays would be adaptively tuned according to workload's dynamic to achieve a workload-aware PM Policy.

A graphical illustration of request processing request is provided in Fig. 5 to better understand how PMC states evolve and how requests are processed vs time. For more explanation of the above process, let's define a *cycle* (Fig. 5) as the time period between instant of triggering T_wakeup and $T\_timeout$ expiry. For each *cycle* corresponds one firing occurrence of both T_wakeup and $T\_timeout$. Workload's increase raises certainly departure frequency of SVS whereas it may increase, decrease or keep constant the number of T_{wakeup} periods (T_{wakeup} firing frequency) during time interval T. This is highlighted in Fig. 5 where three kinds of SVS workloads are considered. Notice that workload increase from very light level (Fig. 5.a) to light level (Fig. 5.b) increases F_{T_wakeup} whereas workload evolution from light level (Fig. 5.b) to loaded level (Fig. 5.c) decreases F_{T_wakeup}. Notice also that according to our modeling approach the request processing process maybe considered as a particular queuing system where the waiting time consists of three cumulative delays:

- the first one, *D1*, is due to *T_wakeup*. In the best case *D1*=0 (if the request arrival occurs and finds PMC in idle state) and in the worst case *D1=T_wakeup* duration (if request arrival occurs and finds buffer empty and PMC in sleep state). If an arrival occurs and

transition *T_wakeup* is already triggered (by a previous arrival) without being fired then *D1* is equal to residual time of *T_wakeup*. In this last case, the following equation holds $0 < D1 < T\_wakeup$;

- the second, *D2*, is due to the waiting time in the SVS buffer (i.e queue) before beginning the service;

- the third delay, *D3*, corresponds to the expected request processing time (mean service duration).

The aforementioned analysis enables to better explain the waiting time accumulated by a request from its arriving instant until its departure from the SVS. This allows also to better understand the overall SVS model.

Place invariants (P-invariants)

In what follows we detail P-invariants of workload-aware PM SRN sub-model (Fig. 4): the first P-invariant corresponds to system property that at most K buffer places are available (i.e. $\#P\_queueplace$ + $\#P\_inqueue$ + $\#P\_active$ + $\#P\_fastactive$ + $\#P\_slowactive$ = K); second P-invariant means that PMC is in one of possible states (i.e. $\#P\_off$ + $\#P\_sleep$+$\#P\_wakeup$+$\#P\_idle$+$\#P\_active$+$\#P\_slowactive$+$\#P\_fastactive$ = 1); third P-invariant (i.e. $\#P\_servslow$ + $\#P\_servfast$ = 1) describes property that PMC, whenever is active, is either serving with fast or slow rate. The property that $\#P\_hightraffic$ + $\#P\_lowtraffic$ = n = 4 means that arrival traffic is composed of n sources.

4.3 Tunable parameters in the proposed Workload-aware PM Policy

In workload-aware PM SRN model (Fig. 4), we distinguish between two kinds of timed transitions:

- Transitions with non-tunable delay ($T\_notrejdown$ and T_wakeup): They are assumed exponentially distributed. The corresponding delays of such transitions are inherent to PMC technology. $T\_notrejdown$ corresponds to the PMC necessary time to transit from off state to sleep state. Whereas T_wakeup duration is the time needed by PMC to transit from sleep state to idle state.

- Transitions with tunable delay according to the context and depending on workload are $T\_timeout$, $T\_servfast$, $T\_servslow$, T_s2f and T_f2s. Tuning delays of such transitions in opportunistic way may be a key factor to make appropriate decision of a given PM policy relying on workload attributes in order to reach good power-performance trade-off without incurring unbearable oscillations. Oscillatory behavior may be avoided by choosing $T\_timeout$ value beyond a given threshold. This threshold corresponds to the break-even time defined in previous sections and expressed by the needed time to switch to and from an active state.

In the following subsection we will define some SVS performability and efficiency metrics that will be investigated later through numerical results with respect to timeout parameter.

4.4 Main contributions of the proposed versatile SVS SRN models

In this section, we discuss the strong points and limitation of the proposed models when compared with similar ones presented in previous works (Escheikh et al. (2014), Escheikh et al. (2016), Escheikh et al. (2017)) and Escheikh et al. (2018):

- In Escheikh et al. (2014) and Escheikh et al. (2017) authors proposed a versatile SVS SRN model for Cold-VM rejuvenation. Whereas in Escheikh et al. (2016) and Escheikh et al. (2018) authors had considered only Migrate-VM rejuvenation scenario.
- In Escheikh et al. (2014) and Escheikh et al. (2017), VM and VMM sub-models are abstracted as a simplified and unified model with reduced state space to avoid untractable calculation. However this simplification may hide detailed specification useful to accurately describe both VMM and VM behaviors. This drawback is circumvented in this work through keeping these detailed specifications (given in previous work Machida et al. (2010)). This is mainly useful for more concrete numerical investigations.
- In the present paper we are concerned in this paper with workload-dependent software aging instead of workload-independent software aging for both VMM and VM (investigated in Escheikh et al. (2014) and Escheikh et al. (2017)).
- We have introduced in the workload-aware PM mechanism presented in this paper (compared with works given in Escheikh et al. (2014) and Escheikh et al. (2017)) a new PMC power state accounting for PMC switching between wakeup state and idle state. This provides more clarification and accuracy in describing PM mechanism.
- Numerical investigations and defined performability metrics in Escheikh et al. (2014) and Escheikh et al. (2017) focus on PM impact regarding performance (i.e. mean waiting time), power usage as well as power-performance efficiency when using time varying bursty workload. The objective is to find opportunistic timeout value (the tunable parameter of the PM mechanism) providing optimal power-performance trade-off for a given workload. In this paper, we are rather concerned with comparative study between SVS subject to Cold-VM rejuvenation and SVS subject to Migrate-VM rejuvenation for different SVS metrics. These metrics highlight the impact of workload-dependent aging, workload and workload burstiness on steady-state availability, performability metrics, power metrics and power-performance metric.
- Another distinction, in Escheikh et al. (2014) and Escheikh et al. (2017), the proposed versatile SVS SRN models are solved using TimeNet 4.1 tool (Armin Zimmermann (2012)) whereas in this work we propose two versatile SVS SRN models and their performability analysis is achieved through SPNica tool (German and Heindl (1999)).

4.5 Discussion about the proposed SVS models

In this subsection we enumerate first the main efforts of using the proposed SVS models. We address then the related limitations and eventual extensions.

1. **Main efforts of using the proposed SVS models**
   The main efforts of using the proposed SVS models is to:

- concisely describe SVS behavior through SRN as a degradable system;
- adopt modular design approach providing more scalability and less complexity than monolithic approach.
- judiciously specify output measures as reward-based functions;
- capture more SVS properties through using extensive marking dependencies without leading to untractable model resolution;
- describe dependencies between on one hand workload and on the other hand software aging and PM mechanism is essentially useful for variable traffic.
- highlight how workload burstiness increases or decreases SVS performability;

2. ***Limitations and eventual extensions of the proposed SVS models:***
The considered SVS models:

- consist of a hosting server handling a VMM with one hosted VM. This latter is assumed running one or more applications over a given operating system. Such assumption may be relaxed by considering a SVS handling several VMs on one VMM;
- assume only software failures;
- are examined for fixed values of *Timeout* time interval (The tunable parameter of the proposed workload-aware PM mechanism) and rejuvenation period (of either VM or VMM). Further investigations of the proposed SVS models may be done to optimize these parameters with respect to workload;
- are analysed only through steady state analysis. Further studies based on transient analysis may also be conducted for the same models.

5 Parameters and metrics of the proposed versatile SVS SRN models

We detail in this section some selected SVS parameters and metrics (traffic metrics, aging factor, performability metrics, power and power-performance metrics) defined from the proposed versatile SVS SRN models. If this is not explicitly stated otherwise we consider that the following metrics are valid by default for model[1] and model[2].

5.1 Traffic parameters

Mean arrival rate (λ): is the mean requests' traffic arriving to SVS and is given by:

$$\lambda = \frac{\lambda_{T\_hightraffic}.\sigma_{T\_l2h} + \lambda_{T\_lowtraffic}.\sigma_{T\_h2l}}{\sigma_{T\_h2l} + \sigma_{T\_l2h}} \quad (1)$$

Index of dispersion for counts (IDC): measures burstiness of requests' traffic arriving to SVS and is given by:

$$IDC = 1 + \frac{2(\lambda_{T\_hightraffic} - \lambda_{T\_lowtraffic})^2 \sigma_{T\_h2l}.\sigma_{T\_l2h}}{(\sigma_{T\_h2l} + \sigma_{T\_l2h})^2 (\lambda_{T\_hightraffic}.\sigma_{T\_l2h} + \lambda_{T\_lowtraffic}.\sigma_{T\_h2l})} \quad (2)$$

5.2 SVS Aging Factor (AF)

We define aging factor as an aging rate weight for both VM and VMM. Such definition is used to better express workload fluctuation impact on software aging rate. Notice that workload-independent software aging depends only on time whereas workload-dependent software aging depends both on time and workload.
Definition: Let AF (Eq. 3) be a workload-dependent dimensionless index defined as the ratio between on one hand powers weighted sum of all PMC states (sleep, idle, slowactive, fastactive, wakeup) and on the other hand power corresponding to PMC in idle state. We choose this latter as a reference power state. Notice that the power weighted sum cited above encompasses the sum of each power value of a given PMC state with assigned weight equals to the probability of the PMC to be in this state. Notice that according to the above choice, AF may be smaller, equals to one or greater than one. For AF<1, AF enables to slow down aging and then enhance availability. For AF=1, aging and availability of SVS are insensitive to aging factor and aging rate is workload independent. For AF>1, aging is speeded up and availability decreases as well as workload increases.

$$AF = E\{\#P\_idle\} + \frac{Powersleep}{Poweridle}.E\{\#P\_sleep\} + \frac{Powerwakeup}{Poweridle}.E\{\#P\_wakeup\}+ \frac{Powerslowactive}{Poweridle}.E\{\#P\_slowactive\} + \frac{Powerfastactive}{Poweridle}.E\{\#P\_fastactive\} \quad (3)$$

5.3 Performability metrics

- **Steady-state availability (As)**

(Cold-VM)
$$As = E\{\#P\_vup\} + E\{\#P\_vfp\} + E\{\#P\_vsd\} + E\{\#P\_vfpsd\} \quad (4)$$

(Migrate-VM)
$$As = E\{\#P\_vup\} + E\{\#P\_vfp\} + E\{\#P\_vmigd\} + E\{\#P\_vfpmigd\} \quad (5)$$

- **Mean waiting time (W) (s)**: is the ratio between the mean number of requests N in SVS (Eq. 7) and SVS throughput Th (Eq. 8)

$$W = N/Th \quad (6)$$

- **Mean number of requests (N)** in the system is given by:

$$N = E\{\#P\_inqueue\} + E\{\#P\_slowactive\} + E\{\#P\_fastactive\} \quad (7)$$

- **SVS throughput (Th) (requests/second)**

$$Th = E\{\#T\_servfast\} + E\{\#T\_servslow\} \quad (8)$$

- **Mean firing frequency of $T\_lossjob$ (requests/second)**: quantifies the mean request loss rate in SVS.

$$F\_Tlossjob = (E\{\#Phightraffic\}.\lambda_{Thightraffic} + E\{\#Plowtraffic\}.\lambda_{Tlowtraffic}).(E\{\#Pvstop\} + E\{\#Pvpre\} + E\{\#Pvfail\}) \quad (9)$$

- **Mean firing frequency of** T_wakeup $(F\_Twakeup)$

  Cold-VM:

$$F\_Twakeup = \frac{A}{B}.\lambda_{T\_wakeup} \tag{10}$$

  Migrate-VM:

$$F\_Twakeup = \frac{C}{B}.\lambda_{T\_wakeup} \tag{11}$$

  where

$$A = (E\{\#P\_vup\} + E\{\#P\_vfp\} + E\{\#P\_vsd\} + E\{\#P\_vpsd\}) + E\{\#P\_wakeup\}.E\{\#P\_inqueue\} \tag{12}$$

$$B = E\{\#P\_inqueue\} + E\{\#P\_queueplace\} + E\{\#P\_active\} + E\{\#P\_slowactive\} + E\{\#P\_fastactive\} \tag{13}$$

$$C = (E\{\#P\_vup\} + E\{\#P\_vfp\} + E\{\#P\_vmigd\} + E\{\#P\_vfmigd\}) + E\{\#P\_wakeup\}.E\{\#P\_inqueue\} \tag{14}$$

5.4 Power metrics

- **Mean consumed power by PMC** (P) **(watt)**

$$P = Powersleep.E\{\#P\_sleep\} + Poweridle.E\{\#P\_idle\} + Powerwakeup.E\{\#P\_wakeup\} + Powerslowactive.E\{\#P\_slowactive\} + Powerfastactive.E\{\#P\_fastactive\} \tag{15}$$

- **Power utilization ratio** (U): is defined as the ratio between $FoTPused$ and $FoTPreq$:

$$U = FoTPused/FoTPreq \tag{16}$$

where $FoTPused$ is the fraction of time active power states are used (i.e. PMC is in an active state) given by:

$$FoTPused = E\{1_{\#P\_fastactive=1}\} + E\{1_{\#P\_slowactive=1}\} \tag{17}$$

and $FoTPreq$ is the fraction of time PMC is in a power-consuming state (i.e. slowactive, fastactive, wakeup, idle, sleep) is given by:

$$FoTPreq = E\{1_{\#P\_fastactive=1}\} + E\{1_{\#P\_slowactive=1}\} + E\{1_{\#P\_wakeup=1}\} + E\{1_{\#P\_idle=1}\} + E\{1_{\#P\_sleep=1}\} = E\{1_{\#P\_off=0}\} \tag{18}$$

5.5 Power-performance metric

- **Efficiency (E) (joules/request)**: is the ratio between mean consumed power P (Eq. 15) and throughput Th (Eq. 8)

$$E = \frac{P}{Th} \tag{19}$$

6 Numerical results

Investigation of the versatile SVS SRN models (model[1] and model[2]) may be achieved at two scales. The first one corresponds to SVS with idle-dominated workload where PM is mainly dominated by timeout tuning (Escheikh et al. (2014) and Escheikh et al. (2017)), whereas the second, known as computationally intensive workload, where PM is dominated by a heavy load asking the highest voltage/frequency setting. We make this distinction to focus separately on each kind of parameters enabling adaptive switching between PMC power states. In idle-dominated workload case timeout delay value (characterizing transition $T\_timeout$) is the main parameter considered. Whereas in computationally intensive workload case the key parameters are transition rates describing PMC serving process ($T\_servfast$, $T\_servslow$, T_f2s, T_s2f). In this section we investigate some numerical results related to the versatile SVS SRN models (model[1] and model[2]). We particularly focus on how workload with bursty nature impacts the behavior of these models. To get insight into workload influence on SVS parameters and metrics defined in the previous section we perform several set of experiments based on analytical analysis using SPNica tool (German and Heindl (1999)). In what follows, we investigate numerical analysis performability and power metrics of SVS:

6.1 Workload impact on SVS aging factor (AF)

Fig. 6 shows the trend of aging factor AF versus workload (λ). Obviously aging factor is an increasing function of workload. According to the plot (Fig. 6), three well-differentiated phases can be appreciated. The first one corresponds to low traffic where AF is slowly increasing. In the second, AF increases sharply with respect to workload evolution. In the last phase, corresponding to overloaded SVS (beyond a certain load), aging factor tends to converge towards a saturation value where it is almost insensitive to workload variations and where PMC power consumption is maximum.

6.2 Workload impact on versatile SVS SRN models' metrics

6.2.1 Workload impact on versatile SVS SRN models' performability metrics

- **SVS Availability (As) (Eq. 4 and Eq. 5)**: We observe from curves in Fig. 7 that workload impact on steady state availability (As) is clearly the most significant for a workload ranging between 0.001 and 1. Outside this range, As is almost insensitive to workload variations. Indeed light workload's impact is so small so that it can not significantly affect software aging and consequently availability, whereas for heavy workload the aging rate is at its highest value and more workload increase will not further affect availability. Notice also that for both rejuvenation techniques (Cold-Migrate and Migrate-VM) As keeps pace with AF. This behavior is relatively straightforward

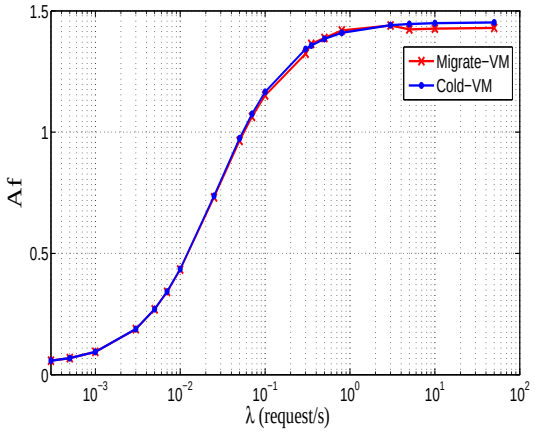


Figure 6: Aging factor index (AF) vs λ

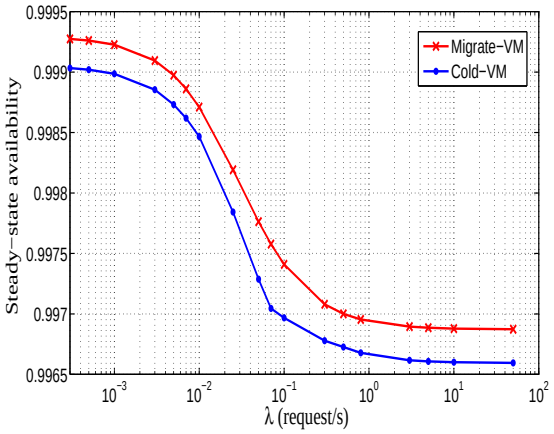


Figure 7: Steady state availability (As) vs λ

inasmuch as availability evolves proportionally to aging factor. It's worth mentioning also that model[2] exhibits a better availability than model[1] for a given workload. This result is predictable since SVS unavailability corresponding to the time spent during VM migration process (for Migrate-VM) is significantly less than the necessary time for VMM rejuvenation (for Cold-VM rejuvenation).

- **Mean waiting time (W) (Eq. 6)**: Fig.8 exhibits the mean waiting time of request per second with respect to workload. It shows that as workload increases W increases in the two versatile SRN SVS models. This is obvious since a greater load in SVS gives rise to longer queue and more delay in request processing.

- **Mean firing frequency of $T\_lossjob$ ($F\_Tlossjob$) (Eq. 9)**: Fig.9 highlights $F\_Tlossjob$ evolution as a function of workload. We observe that $F\_Tlossjob$ increases as a function of the mean arrival rate. This is also straightforward since the more load in SVS the

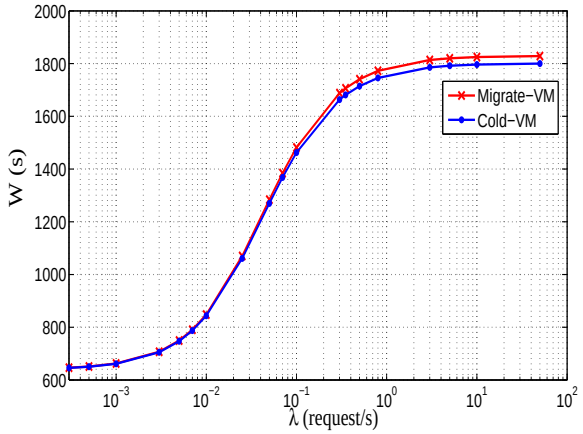


Figure 8: Mean waiting time vs λ

greater requests losses. Noticing also that model[1] suffers more losses than model[2] for any workload. This may be justified by the same arguments given above for the behavior of SVS steady state availability (Fig. 7) with respect to workload.

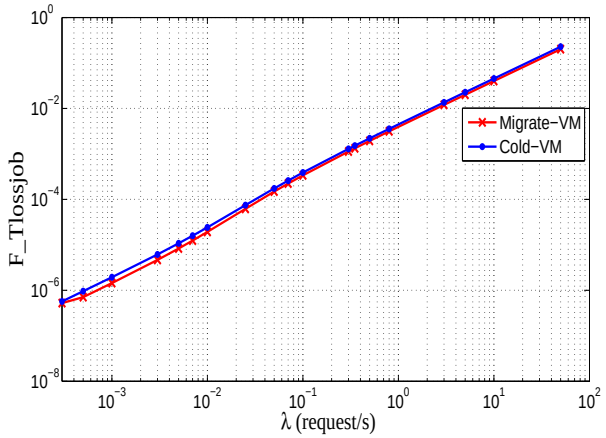


Figure 9: Mean loss rate vs λ

- **Firing frequency of T_wakeup ($F\_Twakeup$) (Eq. 11, Eq. 10)**: $F\_Twakeup$ is an increasing function of workload (Fig.10). Several parameters may be involved in the explanation of such behavior. T_wakeup delay ($\frac{1}{\lambda_{T\_wakeup}}$, Tab. 7) represents transition time from sleep state to idle state. According to Fig. 4 notice that three conditions must be fulfilled simultaneously to enable T_wakeup: *(i)* VM is available (see T_{wakeup} guard in Tab. 4, Tab. 5), *(ii)* there is at least one token in $P\_inqueue$, *(iii)* there is a token in P_sleep (resulting from firing of either $T\_notrejdown$ or $T\_timeout$). Hence T_wakeup firing is closely correlated to the firing of a set of transitions namely $T\_notrejdown$,

$T\_timeout$, $T\_hightraffic$ and $T\_lowtraffic$. Firing frequency of the above transition is intimately related to SVS load level. Firing frequency evolution of $F\_Twakeup$ may be interpreted at different levels of workload.

- For very light traffic (λ <10^{-2}), workload increase does not affect significantly firing frequency of $T\_hightraffic$ or $T\_lowtraffic$ (which remains very low) and keeps $P\_inqueue$ almost empty. In such case even if there is often a token in P_wakeup, T_wakeup will very rarely fire.
- For medium traffic ($\lambda \in [10^{-2}, 10^{2}]$), workload increase increases PMC switching frequency between different ACPI states (sleep, wakeup idle, fastactive, slowactive and off). It increases also SVS alternating frequency between availability and unvailability states (which increases $T\_notrejdown$). All these events are potentially involved in the explanation of $F\_Twakeup$ increase with respect to workload.
- For heavy traffic ($\lambda > 10^{4}$), workload increase increases the number of tokens in $P\_inqueue$ (if it is not full) and accordingly T_wakeup firing frequency depends almost only on token presence in P_wakeup. This last condition is nearly equivalent to having a token in P_sleep which in turn depends on the firing of $T\_notrejdown$ or $T\_timeout$.

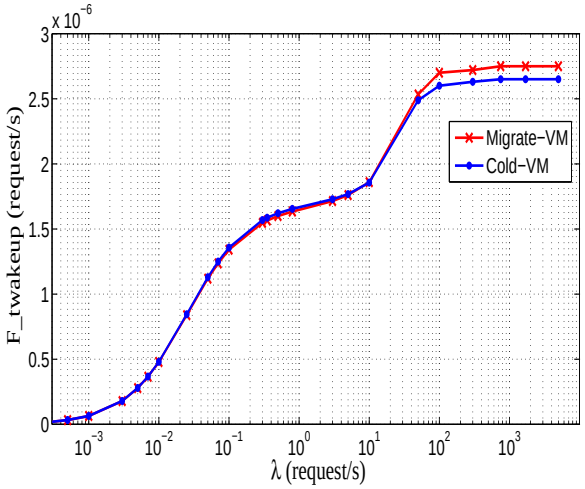


Figure 10: $F_{Twakeup}$ vs λ

6.2.2 *Workload impact on ACPI power PMC states*

Fig.11 shows an increasing probability of the PMC to be in state wakeup ($\#P\_wakeup$) as function of workload. Note that the corresponding curves dynamic follows aging factor trend with respect to workload.

Fig.12 captures probability trend of the PMC to be in state idle (*#P_idle*) as function of workload. Notice that $\#P\_idle$ evolves, for light traffic (i.e. workload), with a positive

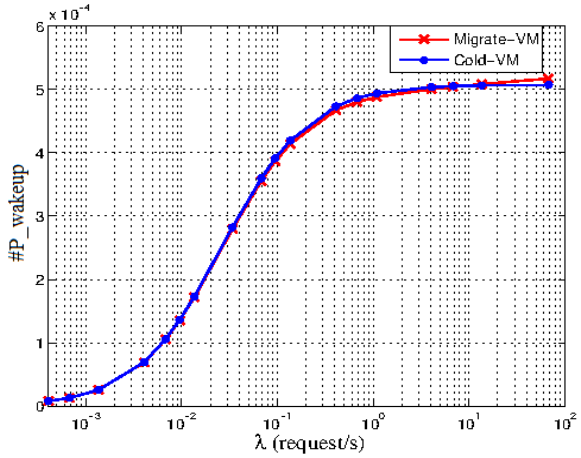


Figure 11: #P_wakeup vs λ

slope until reaching a maximum beyond which the curve begins to decline. Indeed, since PM mechanism, included in the proposed versatile SVS SRN models, is time-based for relative low traffic (requests inter-arrival time is much greater than timeout), each request processing is likely to be followed by a full timeout period. Hence, for light traffic, as

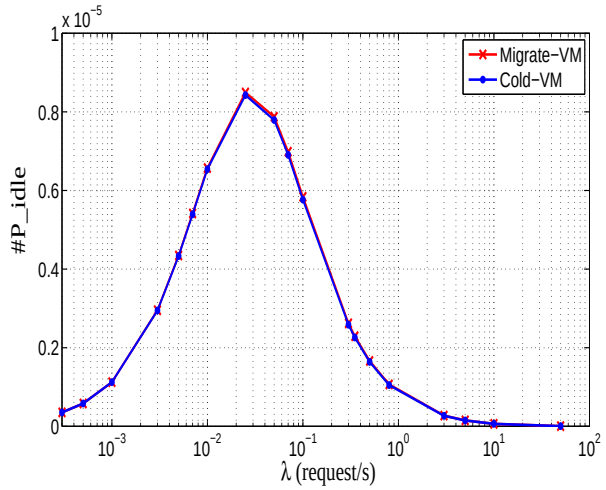


Figure 12: #P_idle vs λ

workload increases the mean inter-request time decreases leading to more requests and more frequent associated idle periods. Beyond a given workload threshold (3.10^{-2}), the inter-request time falls often under timeout value and in such case the next request will be likely processed before timeout expiry. For heavy traffic inter-request time becomes much smaller than timeout and consequently the PMC would process very probably a batch of requests before experiencing the full timeout period. Request batch size increases as

workload increases.
Thereby for light traffic each request processing corresponds, on average, to one timeout period whereas this latter is likely to correspond to a batch size greater than one for heavy traffic.
Obviously as workload increases, PMC's probability to be in state sleep decreases (Fig.13). Indeed workload increase extends time periods where PMC is busy. This consequently shortens time periods where PMC is in sleep state.
Fig.15 and Fig.14 illustrate how ACPI power PMC states evolve for different workloads.

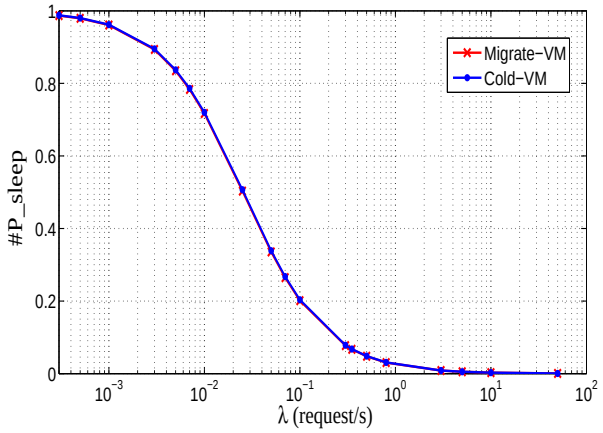


Figure 13: #P_sleep vs λ

In fact for light traffic, sleep is the dominant state and for heavy traffic, slow active and fast active are the most visited PMC's states.

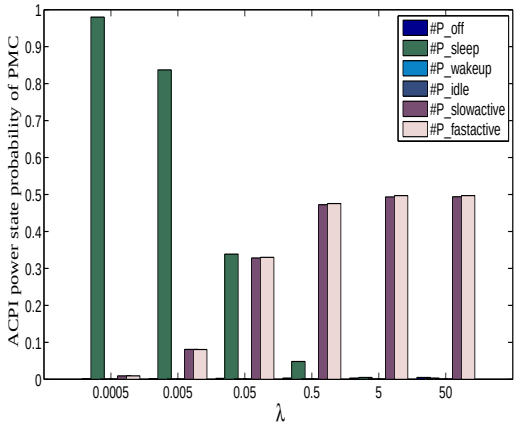


Figure 14: ACPI power states of PMC (Cold-VM)

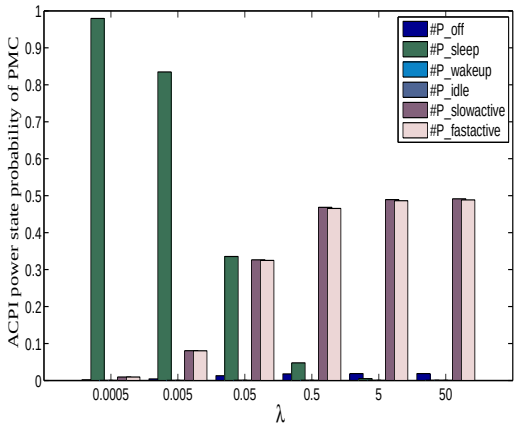


Figure 15: ACPI power states of PMC (Migrate-VM)

6.2.3 Workload impact on versatile SVS SRN models' power metrics

- **Mean consumed power (*P*) (Eq. 15)**: PMC's power consumption against a changing mean arrival rate is depicted in Fig. 16. For small λ ($\lambda < 10^{-3}$) P is low since PMC is nearly always in sleep state consuming by far the smallest power value. Gradually as λ increases, PMC tends to go more to active states and thereby P raises and tends to converge toward a fixed value (this corresponds to maximum power threshold where PMC is almost always in active state). The above explanation is valid for both model[1] and model[2].

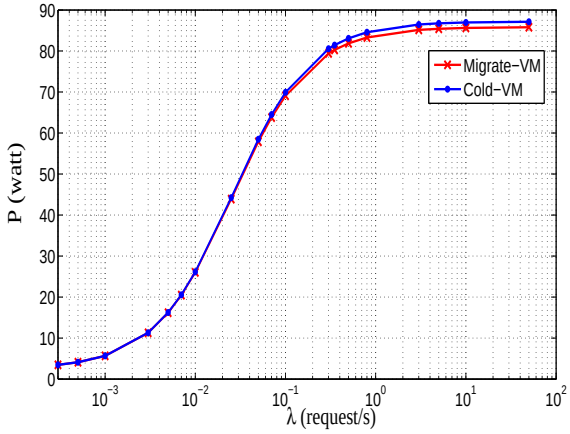


Figure 16: Mean consumed power (P) vs λ

- **Power utilization ratio (*U*) (Eq. 16)**: measures the probability that PMC is effectively in one of the active states while it's not off. Fig. 17 depicts U vs λ. It's worth noting that

for light workload, PMC remains for long periods in sleep state. Instead, for heavy traffic PMC remains most of the time in active states and thereby as workload becomes higher U converges to unity.

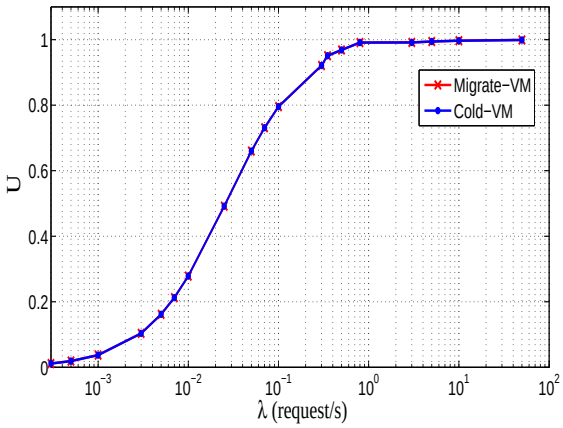


Figure 17: Power utilization ratio (U) vs λ

6.2.4 Workload impact on versatile SVS SRN models' power-performance metric

Efficiency (E) (Eq. 19): Fig. 18 shows efficiency evolution as function of λ. E decreases with respect to workload. As workload increases, sleep period becomes more reduced and requests batch size per cycle increases (for a fixed timeout, Fig.5). Consequently consumed energy per request decreases with respect to workload.
For very light traffic (very lightly loaded SVS) efficiency (i.e. consumed energy in joules per request) is relatively high since it includes consumed energy during a cycle and a sleep period (Fig.5.a). For heavy traffic (loaded SVS) sleep time period becomes very short and the corresponding consumed energy per request becomes negligible. Hence the consumed energy per request is almost equal to the consumed energy during a cycle divided by batch size. This is obvious since it's more economical to process a batch of requests per cycle (batch size =3, Fig.5.c) than to process each request separately (batch size =1, Fig.5.a).

6.3 IDC impact on versatile SVS SRN models' metrics

As IDC increases, traffic burstiness increases. This yields correlated traffic deepening traffic fluctuations. Since in our modeling approach we adopt a workload-aware PM mechanism, processing more bursty traffic is equivalent to serve in the same cycle greater request batch size. This is likely to be achieved by PMC with less time in idle state and more time in sleep state. Since idle state consumes much more power than sleep state, batch processing enables better performability.
Fig.19 shows, for a given λ (corresponding to AF>1), that SVS steady state availability is an increasing function of IDC and that model^2 provides better availability than model^1. This increase begins slowly as IDC increases until reaching a certain IDC value beyond

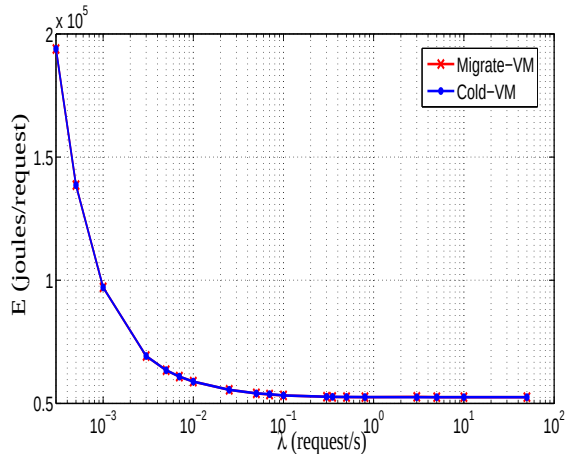


Figure 18: Efficiency (E) vs λ

which it becomes significantly more sharper especially for loaded SVS (λ=10).
For a given mean arrival rate and as IDC increases mean waiting time (W) (Fig.20), mean

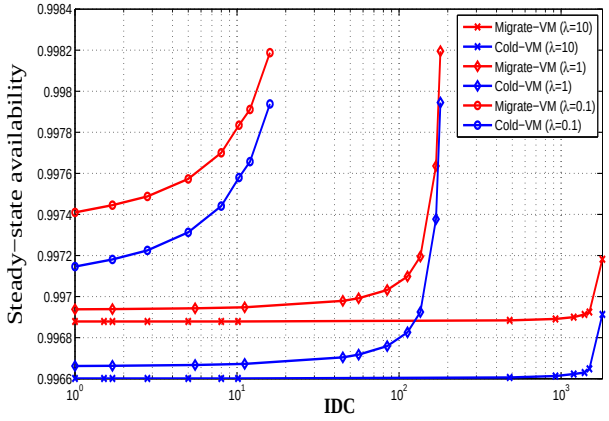


Figure 19: Steady state availability (As) vs IDC (λ = 0.1, 1, 10)

loss rate (Fig.21) and mean firing frequency of Twakeup (Fig.22) decrease. This decrease begins slowly as IDC increases until reaching a certain IDC value beyond which it becomes more acute especially for loaded SVS (λ=10) where batch size becomes greater and sleep period becomes larger. This explains in large part the curve shape for traffic where burstiness is enough high.

For a given mean arrival rate and as IDC increases *#P_wakeup* (Fig.23) decreases.This decrease starts slowly and beyond a given threshold becomes much more pronounced especially for high workload (λ=1, 10). Investigation of #P_idle (Fig.24) with respect to IDC shows two phases, an increasing phase and/or a decreasing phase. The same arguments detailed for Fig.12 may be used here to explain this behavior. For the same reasons cited

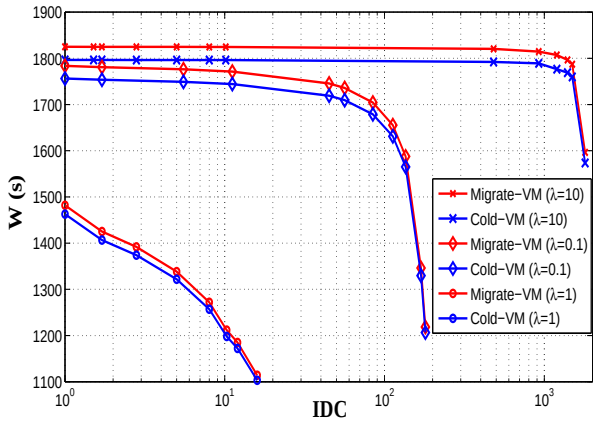


Figure 20: Mean waiting time vs IDC (λ = 0.1, 1, 10)

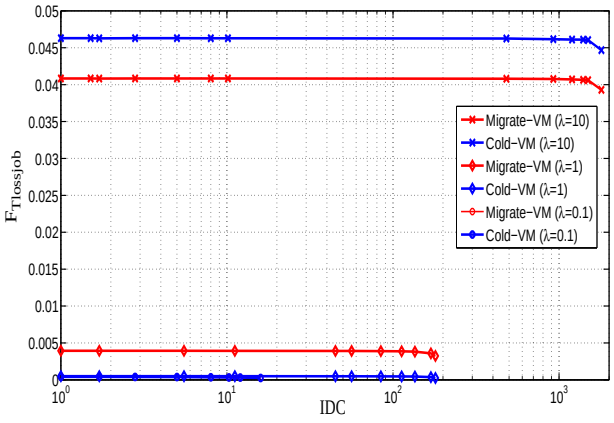


Figure 21: $F\_Tlossjob$ vs IDC (λ = 0.1, 1, 10)

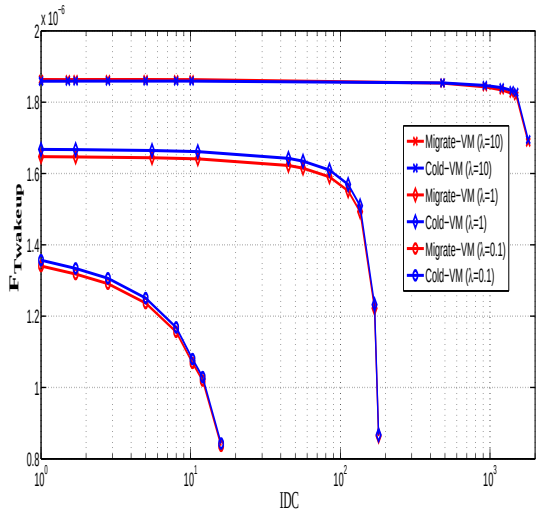


Figure 22: $F\_Twakeup$ vs IDC (λ = 0.1, 1, 10)

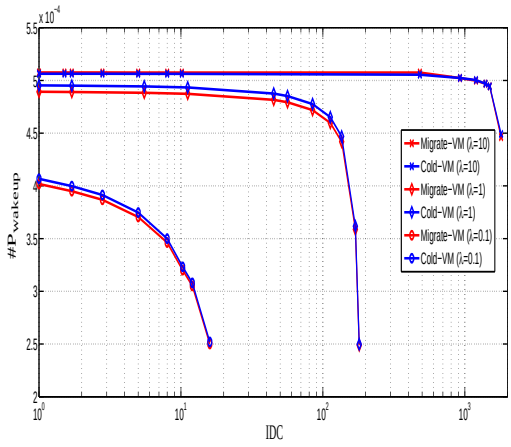


Figure 23: *#P_wakeup* vs IDC (λ = 0.1, 1, 10)

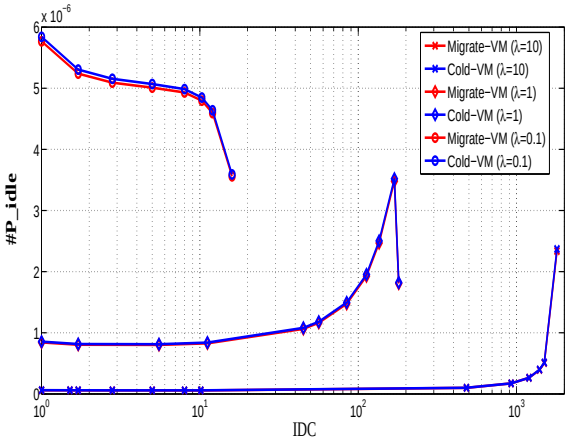


Figure 24: *#P_idle* vs IDC (λ = 0.1, 1, 10)

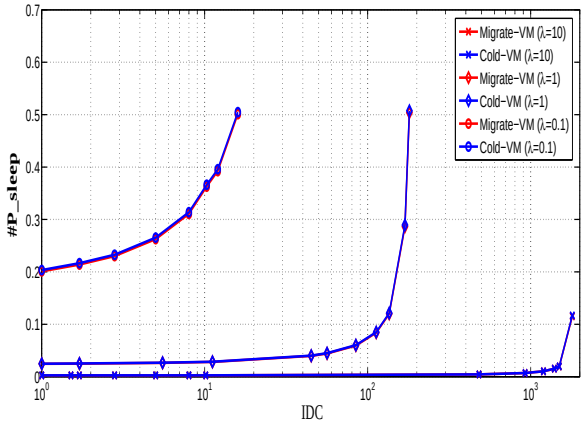


Figure 25: *#P_sleep* vs IDC (λ = 0.1, 1, 10)

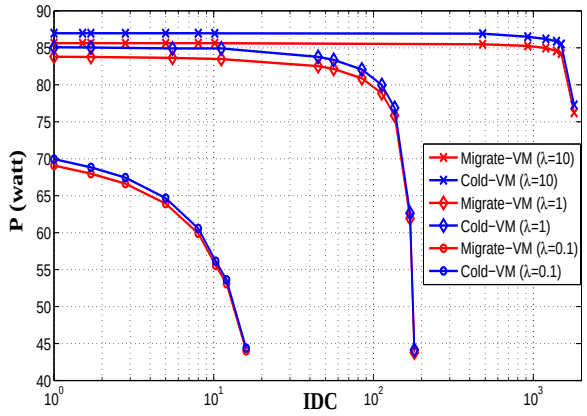


Figure 26: Mean consumed power (P) vs IDC (λ = 0.1, 1, 10)

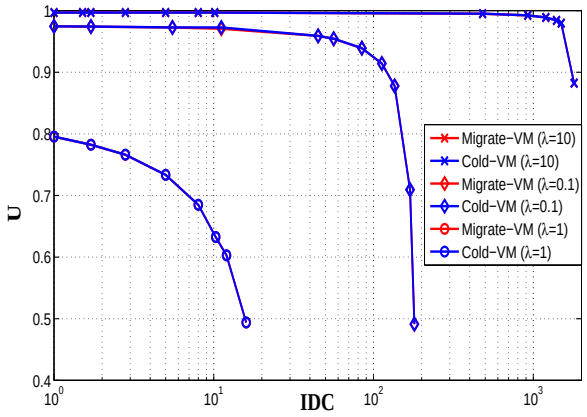


Figure 27: Power utilization ratio (U) vs IDC (λ = 0.1, 1, 10)

above in this subsection and beyond a given IDC value the following metrics, power consumed by PMC (P) (Fig. 26), power utilization ratio (U) (Fig. 27), efficiency (E) (Fig. 28), decrease sharply with respect to IDC. In conclusion, we can deduce that for a fixed average request rate (workload) and for aging factor $AF > 1$, burstiness enables to aggregate requests in batch forms. As IDC increases, batch size increases and this improves significantly all performability SVS attributes investigated in our modeling analysis. Fig. 29 clearly illustrates how IDC increase influences on one hand PMC states' evolution and request processing vs time (for different IDC and constant λ) in the SVS and improves on other hand the performability metrics of SVS.

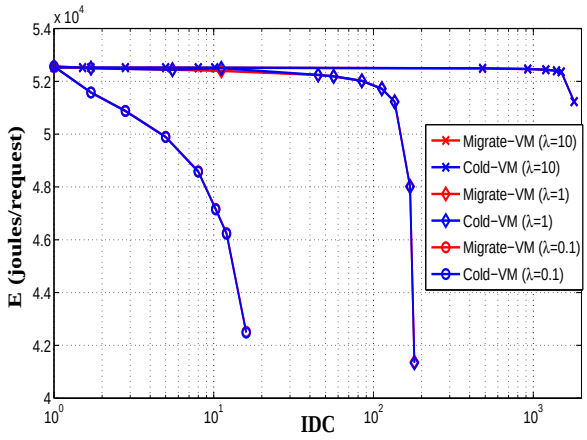


Figure 28: Efficiency (E) vs IDC (λ = 0.1, 1, 10)

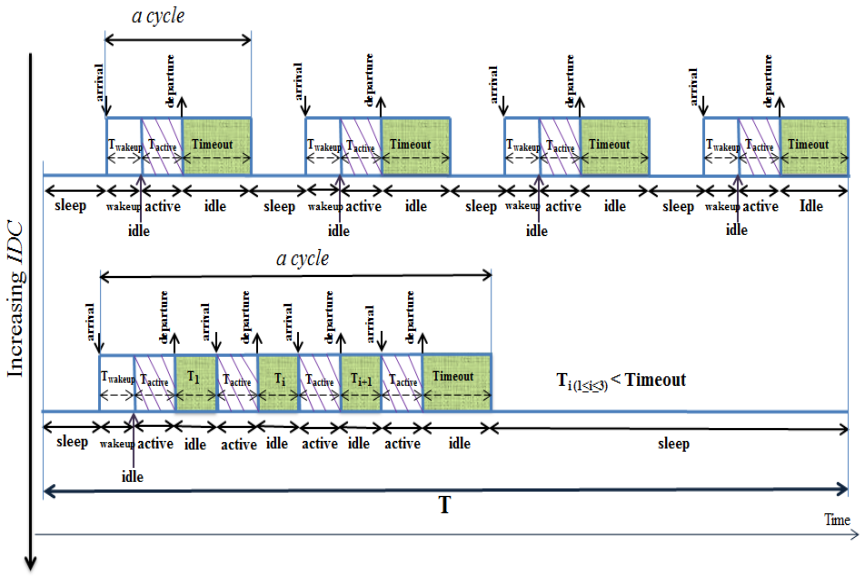


Figure 29: Illustration of PMC states' evolution and request processing vs time (for different IDC and constant λ) in the SVS

7 Conclusion

We proposed in this paper a performability modeling and analysis based on non-Markovian SRNs of versatile SVS hypervisor-based incorporating workload-aware PM mechanism and accounting for workload-dependent software aging. The considered analysis concerns two kinds of VMM rejuvenation namely Cold-VM and Migrate-VM. It relies on modular approach involving several attributes in SVS modeling and enables to capture bursty workload impact on SVS performability metrics. We have defined and investigated numerically various quantitative and qualitative metrics such as steady-state availability, mean consumed power, power utilization ratio, and efficiency. These metrics are used through numerical investigations to show how performance availability, power usage, and efficiency of SVS are impacted by workload with bursty nature for SVS either subject to Cold-VM or Migrate-VM. In this direction, the performability metrics dynamic had been investigated on one hand with respect to SVS workload and on the other hand with respect to IDC. The obtained results confirm that for SVS subject to workload-dependent aging as well as traffic becomes more bursty SVS becomes more available and more power efficient.

References

Bing Wei, Chuang Lin, Xiangzhen Kong. (2011) 'Dependability Modeling and Analysis for the Virtual Data Center of Cloud Computing', *IEEE 13th International Conference on High Performance Computing and Communications (HPCC), Banff, AB,* pp.784-789. [DOI:10.1109/HPCC.2011.111]

Beloglazov, A., Buyya,R., Lee, Y. C., et al.,(2010) 'A taxonomy and survey of energy-efficient data centers and cloud computing systems', *Adv.Comput*,82: pp. 47-111 [DOI:10.1016/B978-0-12-385512-1.00003-7]

Chen, Y., Das, A., Sivasubramaniam, W. Qin., et al., (2005) 'Managing Server Energy and Operational Costs in Hosting Centers', *Proceeding SIGMETRICS '05 Proceedings of the 2005 ACM SIGMETRICS international conference on measurement and modeling of computer systems, New York, NY, USA,* pp.303-314. [DOI:10.1145/1064212.1064253]

Meyer, J. F., (1992) 'Performability: a retrospective and some pointers to the futur', Performance evaluation, Vol. 14 No. 3-4, pp.139-156.

Laprie, J. C., (1992) 'Dependability: Basic concepts and terminology', Dependability: Basic Concepts and Terminology, pp.3-245.

Kuehn,P. J., and Mashaly, M. E., (2015) 'Automatic energy efficiency management of data center resources by load-dependent server activation and sleep modes', Ad Hoc Networks, Vol. 25, pp. 497–504.

Alonso, J., Matias, R., Vicente, E., Maria, A., and Trivedi, K. S., (2013) 'A comparative experimental study of software rejuvenation overhead', Performance Evaluation, Vol. 70 No. 3, pp. 231-250.

Cotroneo, D., Natella, R., Pietrantuono, R., and Russo, S., (2014) 'A survey of software aging and rejuvenation studies', ACM Journal on Emerging Technologies in Computing Systems (JETC), Vol. 10 No. 1, pp. 8.

Cotroneo, D., Natella, R., Pietrantuono, R., and Russo, S., (2011) 'Software aging and rejuvenation: Where we are and where we are going', *Software Aging and Rejuvenation (WoSAR), 2011 IEEE Third International Workshop on*, pp. 1-6.

Araujo, J., Matos, R., Alves, V., Maciel, P., Souza, F., and Trivedi, K. S., (2014) 'Software aging in the eucalyptus cloud computing infrastructure: characterization and rejuvenation',ACM Journal on Emerging Technologies in Computing Systems (JETC), Vol. 10 No. 1, pp. 11.

Saito, Y., and Dohi, T., (2014) 'Predicting software reliability via completely monotone nonparametric estimator with grouped data', Journal of Systems and Software, Vol. 117, pp. 296–306.

Grottke, M., and Schleich, B., (2014) 'How does testing affect the of aging software systems?', Performance Evaluation, Vol. 70 No. 3, pp. 179-196.

Nathuji, R., England, P., Sharma, P. et al., (2009) 'Feedback Driven QoS Aware Power Budgeting for Virtualized Servers', *EuroSys'10 Proceedings of the* 5^{th} *European conference on computer systems, Paris, France,* pp.237-250. [DOI:10.1145/1755913.1755938]

Tadao Murata., (1989) 'Petri nets: Properties, analysis and applications', *Proceedings of the IEEE*, Vol.77 No.4, pp:541-580.

Gaoa, Y., Qia, Z., Wanga, B.,et al., (2013) 'Quality of Service Aware PM for Virtualized Data Centers', *Journal of Systems Architecture - Embedded Systems Design*, Vol.59, pp:245-259. [DOI:10.1016/j.sysarc.2013.03.007]

Beloglazov, A., Abawajy, J., Buyya, R., (2012) 'Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing', *Journal of Future Generation Computer Systems*, Vol.28, pp.755-768. [DOI:10.1016/j.future.2011.04.017]

Elnozahy, E., Kistler, M., Rajamony, R., (2003) 'Energy-efficient server clusters in Power-Aware Computer Systems', *Second International Workshop, PACS 2002 Cambridge, MA, USA,* pp.179-197. [DOI:10.1007/3-540-36612-1-12]

Han, L., and Xu, J., (2013) 'Availability Models for Virtualized Systems with Rejuvenation', *Journal of Computational Information Systems*, Vol.9, No.20, pp:8389-8396. [DOI:10.12733/jcis8586]

Machida, F., Kim, D. S., Trivedi, K. S., (2010) 'Modeling and Analysis of Software Rejuvenation in a server virtualized system', *IEEE Second Int. Workshop on Software Aging and Rejuvenation (WoSAR), San Jose, CA,* pp.1-6. [DOI:10.1109/WOSAR.2010.5722098]

Wang, D., Xie, W., Trivedi, K. S., (2007) 'Performability analysis of clustered systems with rejuvenation under varying workload', *Perform, Eval.*, pp.247-265. [DOI:10.1016/j.peva.2006.04.002]

Xie, W., Hong, Y., Trivedi, K. S., (2004) 'Software rejuvenation policies for cluster systems under varying workload', *Proceedings,* 10^{th} *IEEE Pacific Rim International Symposium on Dependable Computing*, pp.122-129. [DOI:10.1109/PRDC.2004.1276563]

Takeda, S., Takemura, T., (2010) 'A rank-based VM consolidation method for power saving in datacenters', *IPSJ Transactions on Advanced Computing System*, Vol.3, No.2, pp.138-146. [DOI:10.2197/ipsjtrans.3.88]

Ye, K., Huang, D., Jiang, X., et al., (2010) 'Virtual machine based energy efficient data center architecture for cloud computing: A performance perspective', *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Intl Conference on and Intl Conference on Cyber, Physical and Social Computing (CPSCom), Hangzhou,* pp.171-178. [DOI:10.1109/GreenCom-CPSCom.2010.108]

Feller, E., Morin, C., Leprince, D., (2010) 'State of the art of power saving in clusters and results from the EDF case study', *Institut National de Recherche en Informatique et en Automatique (INRIA).* [DOI:inria-00542889v2]

James William Smith and Ian Sommerville. '2013) 'Understanding tradeoffs between Power Usage and Performance in a Virtualized Environment', *IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA,* pp.725-731. [DOI:10.1109/CLOUD.2013.138]

Gaganpreet Kaur Sehdev and Anil Kumar, (2015) 'Performance Evaluation of Power Aware VM Consolidation using Live Migration', *International Journal of Computer Network and Information Security(IJCNIS)*, Vol.7, No.2, pp.67-76. [DOI:10.5815/ijcnis.2015.02.08]

Ghosh, R., Naik, V.K., Trivedi, K.S., (2011) 'Power-Performance Trade-offs in IaaS Cloud: A Scalable Analytic Approach', *IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN), Hong Kong,* pp.152-157. [DOI:10.1109/DSNW.2011.5958802]

Beloglazov, A., and Buyya, R., (2013) 'Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints', *IEEE Transactions on Parallel and Distributed Systems*, Vol.24, No.7 pp.1366-1379. [DOI:10.1109/TPDS.2012.240]

Bobroff, N., Kochut, A., Beaty, K., (2007) 'Dynamic placement of virtual machines for managing sla violations', *IFIP/IEEE* 10^{th} *International Symposium on Integrated Network Management, Munich,* pp.119-128. [DOI:10.1109/INM.2007.374776]

Escheikh, M., Jouini, H., Barkaoui, K., (2014) 'A versatile traffic and power aware performability analysis of server virtualized system', *IEEE* 22^{nd} *International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems. MASCOTS'2014, Paris,* pp.207-212. [DOI:10.1109/MASCOTS.2014.34]

Escheikh, M., Tayachi, Z., Barkaoui, K., (2016) 'Workload-Dependent Software Aging Impact on Performance and Energy Consumption in Server Virtualized Systemsi, *IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pp. 111-118. [DOI:10.1109/ISSREW.2016.31]

Escheikh, M., Barkaoui, K., Jouini H., (2017) 'Versatile workload-aware power management performability analysis of server virtualized systems', *Journal of Systems and Software*, Vol. 125, pp.365-379.[DOI:10.1016/j.jss.2016.12.037]

M. Escheikh, Z. Tayachi, K. Barkaoui. (2018)'Performability evaluation of server virtualized systems under bursty workload', *14th IFAC Workshop on Discrete Event Systems WODES, Sorrente, Italy*, May 2018, Vol. 51(7), pp.45-50, [DOI: 10.1016/j.ifacol.2018.06.277]

Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation,and et al., (2011) *Advanced configuration and power interface specification revision 5.0. URL: http://www.acpi.info/spec50.htm*

Simunic, T., Benini,L., Glynn, P., et al., (2000) 'Dynamic power management for portable systems', *Proceedings of International Conference on Mobile Computing and Networking, MobiCom'00*, pp. 11-19. [DOI:10.1145/345910.345914]

German R. and Heindl A., (1999) 'Performance evaluation of IEEE 802.11 wireless LAN's with stochastic Petri nets', *The* 8^{th} *International Workshop on Petri Nets and Performance Models, Zaragoza,* pp.44-53. [DOI:10.1109/PNPM.1999.796531]

Zimmermann, Armin, (2012) 'Modeling and evaluation of stochastic Petri nets with TimeNET 4.1', *The* 6^{th} *IEEE International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS)* pp.54-63. [DOI: 10.4108/valuetools.2012.250263]

Muppala, Jogesh and Ciardo, Gianfranco and Trivedi, Kishor (1994) 'Stochastic reward nets for reliability prediction', *Journal of communications in reliability, maintainability and serviceability*, vol.1, No.2, pp. 9-20.