Software Rejuvenation in the Cloud

Dario Bruneo, Francesco Longo,
Antonio Puliafito, Marco Scarpa
Dipartimento di Matematica, Università degli Studi di Messina
Viale F. Stagno d'Alcontres, 31 - 98166
Messina (ME), Italia
{dbruneo,flongo,apuliafito,mscarpa}@unime.it

Salvatore Distefano
Dipartimento di Elettronica e Informazione,
Politecnico di Milano
Via Ponzio, 34/5 - 20133
Milano (MI), Italia
distefano@elet.polimi.it

ABSTRACT

In this paper, we investigate how software rejuvenation can be used in a Cloud environment to increase the availability of a virtualized system composed of a single virtual machine monitor (VMM) on top of which a certain number of virtual machines (VMs) can be instantiated. We start from the assumption that the aging of a VMM increases with the number of VMs it is managing, thus characterizing the problem in terms of dynamic reliability. Therefore, by identifying the age of the VMM with its reliability and based on the conservation of reliability principle, we characterize the time to failure of the VMM through continuous phase type distributions. The system availability is thus modeled by an expanded continuous time Markov chain expressed in terms of Kronecker algebra in order to face the state space explosion and to keep memory of the age reached by the VMM in case the number of the hosted VMs change. Time-based rejuvenation is taken into consideration and the optimal timer is evaluated in order to maximize the VMM availability.

Keywords

Rejuvenation, Cloud Computing, Virtualized Environments, Dynamic Availability, Phase Type Distributions.

1. INTRODUCTION

Software rejuvenation is a proactive fault management technique aimed at cleaning up the system internal state to prevent the occurrence of more severe crash failures in the future [1],[2],[3],[4],[5]. It involves occasionally terminating an application or a system, cleaning its internal state and restarting it. Software rejuvenation is a cost effective technique for dealing with software faults that offers protection not only against hard failures, but against performance degradation as well. In a client-server type of application where the server is intended to run perpetually for providing a service to its clients, rejuvenating the server process periodically during the most idle time of the server increases the availability of that service. In a long-running computation-intensive application, rejuvenating the application periodically and restarting it at a previous checkpoint increases the likelihood of successfully completing the application execution.

Simutools 2012, March 19-23, Desenzano del Garda, Italy

DOI 10.4108/icst.simutools.2012.247772

The use of rejuvenation in Cloud systems is currently a hot research topic. By Cloud computing and its terminology, it is implied an infrastructure of shared hardware and software resources, deeply rooted on the premise of virtualization [6]. The elastic behaviour of Cloud systems translates into the availability of resources that always fit the requirements of users and their applications, bringing setup expenses to a minimum and putting a cap on monthly bills. We believe that software rejuvenation can be successfully applied to Cloud systems to mitigate the effect of aging that may affect the *Virtual Machine Monitor* (VMM), i.e., the software module that allows to execute the *Virtual Machines* (VMs).

Understanding the way in which VMMs age and how this can influence the service availability is extremely important, considering that Cloud services strongly involve the commercial world, where Service Level Agreement (SLA) and Quality of Service (QoS) play a strategic role. The stress condition of a VMM is also depending on the amount of work it is requested to do. This means that the VMM aging increases with the number of virtual machines it is managing. A workload-based rejuvenation and an analysis on the way the workload can be considered a critical factor to determine the system failure distribution can be found in [1]. In this paper, we address this problem through the definition of stochastic models that allow to capture the aging behavior just described.

In order to avoid VMM crashes, that would implicitly stop the execution of several VMs, VMM rejuvenation may represent a valid solution. We adopted a time-based rejuvenation technique [5] to investigate the optimal time to rejuvenation with the final goal to maximize the VMM availability.

The paper is organized as follows. In Section 2 we provide a description of Cloud computing systems. In Section 3 we formulate the problem. In Section 4 we describe the proposed model. Some details on the solution technique are given in Section 5 while in Section 6 we present the obtained results. Concluding remarks are presented in Section 7.

2. CLOUD COMPUTING

This section provides a global picture of the heterogeneous

Cloud scenario discussing goals and motivations, providing some well-known definitions, identifying related problems and issues and giving overview of existing solutions, challenges, and future directions.

2.1 Goals and motivations

The necessity of dealing with complex problems and the technological progress focused on mechanizing and automating processes and services brought to the exigence of more and more powerful computing systems. After the initial monolithic approach of implementing more complex (standalone) systems, the idea of decomposing and scattering computing tasks among distributed resources prevailed giving rise to parallel and distributed computing. Several distributed computing paradigms and infrastructures have been therefore specified, starting from client-server, clustering (also structured and hierarchical), peer to peer and others.

Before Cloud computing, the distributed computing scenario was dominated by the Grid, volunteer and P2P solutions, that were mainly bounded to academic, scientific, private, closed contexts and virtual organization. The Cloud unlocked the scenario, mainly contained within specific and well identified boundaries, disclosing a new world of services in which whatever can be considered as a service, also and in particular the computing resources. Anyone can rent resource from Cloud providers by negotiating and agreeing specific SLAs, that contain specific commitments for the parties.

Among the reasons behind the success of Cloud, besides the low costs, there are: the user-centric interface that acts as a unique, user friendly, point of access for user needs and requirements; on-demand service provision; the QoS guaranteed offer, and the autonomous system for managing hardware, software and data transparently to users [7]. But, on the other hand, there are different open problems in Cloud infrastructures that inhibit their use mainly concerning information security (confidentiality and integrity), trustiness, interoperability, reliability, availability and other QoS requirements specified in the SLA, only partially addressed or sometimes still uncovered.

2.2 Definitions and taxonomy

Among the several different definitions of "Cloud computing" available in the literature, one of the most authoritative is that provided by NIST [8]: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This Cloud model promotes availability [...]." It is important to remark that such a definition identifies the availability as a key concept of the Cloud paradigm. This concept has to be characterized into the broader class of QoS, SLA and related issues, which are topics of primary and strategic importance in the Cloud.

An interesting attempt to fix Cloud concepts and ideas is provided in [9] thorough an ontology which demonstrates a dissection of the Cloud into the five main layers shown in Figure 1, where higher layers services can be composed from the services of the underlying layers, which are:

1. *Cloud Application Layer:* provides interface and access management tools (Web 2.0, authentication, billing, SLA, etc.), specific application services, services mashup tools, etc. to the Cloud end users. This model is referred to as Software as a Service (SaaS).
2. *Cloud Software Environment Layer*: providers of the Cloud software environments supply the users Cloud applications' developers with a programming-language-level environment with a set of well-defined APIs. The services provided by this layer are referred to as Platform as a Service (PaaS).
3. *Cloud Software Infrastructure Layer*: provides fundamental resources to other higher-level layers. Services can be categorized into:
   - *Computational resources* - provides computational resources (VM) to Cloud end users. Often, such services are dubbed Infrastructure as a Service (IaaS).
   - *Data storage* - allows users to store their data at remote disks and access them anytime from any place. These services are commonly known as Data-Storage as a Service (DaaS)
   - *Communications* provides some communication capabilities that are service oriented, configurable, schedulable, predictable, and reliable. Towards this goal, the concept of Communication as a Service (CaaS) emerged to support such requirements, as well as network security, dynamic provisioning of virtual overlays for traffic isolation or dedicated bandwidth, guaranteed message delay, communication encryption, and network monitoring.
4. *Software Kernel*: provides the basic software management for the physical servers that compose the Cloud. OS kernel, hypervisor, virtual machine monitor, clustering and grid middleware, etc.
5. *Hardware and Firmware*: form the backbone of the Cloud. End users directly interacting with the Cloud at this layer are normally big enterprises with huge IT requirements in need of subleasing Hardware as a Service (HaaS).

2.3 Modeling challenges

In the design and management of a Cloud system, performance evaluation plays a key role allowing system administrators to evaluate the effects of different resource management strategies on the data center functioning and to predict the corresponding costs/benefits.

Cloud systems differ from traditional distributed systems. First of all, they are characterized by a very large number of resources that can span different administrative domains. Moreover, the high level of resource abstraction allows to implement particular resource management techniques such as VM multiplexing [10] or VM live migration [11] that, even if transparent to final users, have to be considered in the design of performance models in order to accurately understand the system behavior. Finally, different clouds, belonging to the same or to different organizations, can dynamically join each other to achieve a common goal, usually represented by the optimization of resources utilization. This mechanism, referred to as cloud *federation* [12], allows to provide and release resources on-demand thus providing elastic capabilities to the whole infrastructure.

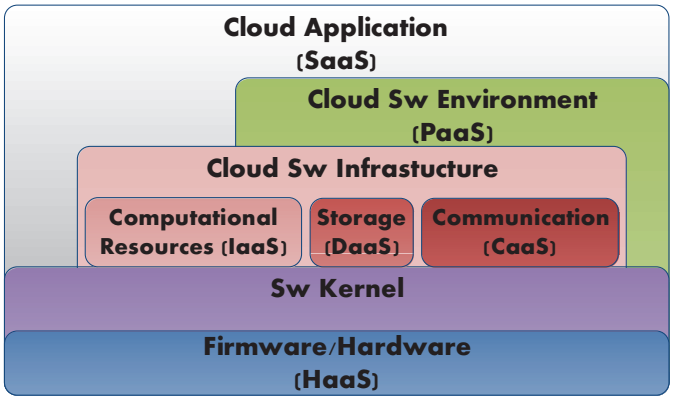

Figure 1: Cloud Taxonomy.

For these reasons, typical performance evaluation approaches such as simulation or on-the-field measurements can not be easily adopted. Simulation [13], [14] does not allow to conduct comprehensive analyses of the system performance due to the great number of parameters that have to be investigated. On-the-field experiments [15], [16] are mainly focused on the offered QoS, they are based on a black box approach that makes difficult to correlate obtained data to the internal resource management strategies implemented by the system provider. On the contrary, analytical techniques [17], [18] represent a good candidate thanks to the limited solution cost of their associated models.

The modeling scenario becomes more tricky when prevention techniques, like software rejuvenation, want to be applied to a Cloud infrastructure in order to obtain high availability degrees. In fact, in this case, also the analytical approaches, in particular state-space techniques, suffer from the complexity of the corresponding models that have to take into account different policies and operative conditions, also considering non-Markovian aspects. In such cases, techniques able to opportunely manage the state space explosion and to deal with generally distributed events have to be exploited in order to create effective and scalable models.

3. PROBLEM FORMULATION

In this section, in order to focus on the rejuvenation aspects in the area of IaaS Clouds, we describe the architecture of a simplified system discussing all the assumptions we made. A Cloud node offers a virtualized environment composed of a VMM on top of which one or more VMs can be instantiated. We indicate with N the maximum number of VMs that can be contemporaneously instantiated. User requests (in terms of new VMs) are scheduled by loading pre-built VM images from a storage system and by registering them to the VMM. Once instantiated, a VM can start its execution providing to the user an isolated execution environment.

The VMM is a software module subjected to software aging due to bugs. For this reason, the VMM can fail giving rise to critical effects on the running VMs, ranging from service interruption to data loss. The VMM failure can be detected for example by periodically inspecting the memory usage or the response time. When a failure is detected, the status of all the running VMs is saved (when possible) and the VMM is repaired. As soon as the repairing has been performed, the VMs can be reactivated. We assume that after a reparation the VMM age is reset.

In order to reduce the failure probability, a software rejuvenation task can be triggered. In this case, the running VMs are suspended in a safety way and the VMM is rebooted, thus resetting its aging state. The effects of a software rejuvenation can be considered less critical with respect to those of a failure. In fact, the time needed to recover a failed system is obtained by summing the time to detect the failure, the time to save the VMs status, the time to discover the failed component, the time to repair it, the time to reboot the system, and the time to restart the VMs. On the contrary, the rejuvenation duration time is composed of the time to suspend the VMs, the time to reboot the system, and the time to restart the VMs. Moreover, being the rejuvenation a scheduled task, all the countermeasures to reduce the QoS degradation (e.g., VM live migration, data saving, etc.) can be taken in advance. In the reminder of the paper, we focus on the aspects related to the aging and failure of the VMM and to its rejuvenation, thus neglecting the failure of other system components.

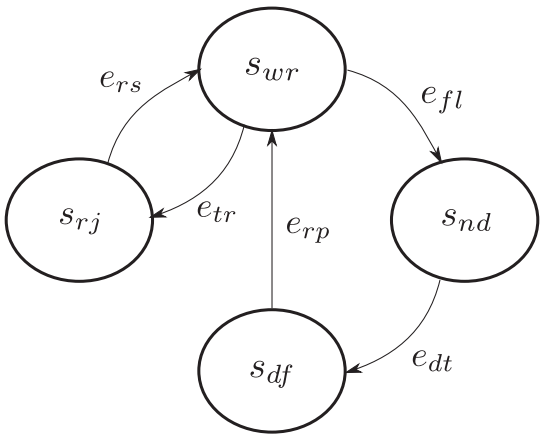

Figure 2: Operating states of a Virtual Machine Monitor.

From an operating point-of-view the VMM is characterized by four states, as shown in Figure 2. Once started, the VMM is in the *working* state s_{wr}. During the sojourn in this state, the VMM age increases due to software bugs. When a *failure* event e_{fl} occurs, the VMM enters the *non-*

detected failure state s_{nd}. In this state, the VMM is not able to accomplish its work but the failure has yet to be detected. Once the failure is detected (detection event e_{dt}), the VMM enters the *detected failure* state s_{df} and as soon as a reparation is completed (*repair* event e_{rp}) the VMM is rebooted thus re-starting from state s_{wr}. When a rejuvenation is needed, the *trigger* event e_{tr} fires and the VMM enters the *rejuvenation* state s_{rj}. The *resume* event e_{rs} allows the system to be rebooted reinitializing its software age.

3.1 VMM failure and repair

With respect to the system events, we make the following assumptions. Let t_{fl} be the random variable representing the time at which a VMM failure occurs.

In a Cloud system the load is highly variable due to request bursts and migration tasks. In order to satisfy the user requests, also in terms of QoS, the VMM has to implement different strategies to quick instantiate, suspend, and transfer VMs. Such strategies have a critical impact on the VMM behavior being strictly related to memory management tasks (such as memory allocation/deallocation). As a consequence, they could interfere with the VMM software aging process. For these reasons, we assume the failure time distribution to be dependent on the VMM load. Indicating with #VM the number of running VMs on the VMM, we express such dependency by defining the failure time cumulative distribution function (CDF) as:

$$F_{fl}(t,i) = Pr\{t_{fl} \leq t \mid \#VM = i\},\ \forall i : 0 \leq i \leq N \qquad (1)$$

Let t_{rp} be the random variable associated to the time to repair. We assume that a checkpoint service periodically save the VMM status by making a dump of its memory on the disk. Since the status of the n VMs being executed on top of the VMM at the time it crashed has to be recovered from the checkpoint database, we assume that event e_{rp} has a 2-variable CDF $F_{rp}(t,i)$ associated:

$$F_{rp}(t,i) = Pr\{t_{rp} \leq t \mid \#VM = i\}. \qquad (2)$$

Finally, we indicate with $F_{dt}(t)$ the CDF associated to the detection event e_{dt} whose duration, differently from the other events above reported, does not depend on the number of VMs managed by the VMM at the time the failure occurs.

Both event e_{fl} and e_{rp} depend on the number of VM currently instantiated on the VMM. In the following, we will indicate with e_{ar} and e_{dp} the event associated with the arrival of a new VM and the departure of an instantiated VM, respectively. $F_{ar}(t)$ and $F_{dp}(t,i)$ are the CDFs associated to such events. While the arrival event is independent from the number of VM currently running, the departure event depends on it.

3.2 VMM rejuvenation

In this work, we focus on a time-based rejuvenation technique. The simplest policy that can be applied to the VMM is to fix a constant timer T associated to the trigger event e_{tr}. Usually, such timer is fixed at the system startup and it is chosen in order to maximize the system availability.

Finally, let t_{rs} be the random variable associated to the time needed to carry out the VMM rejuvenation (resume event e_{rs}). Since the status of the VMs running on the VMM immediately before the rejuvenation has to be recovered from the checkpoint database, we assume that the CDF associated to event e_{rs} is a function of both time and load:

$$F_{rs}(t,i) = Pr\{t_{rs} \leq t \mid \#VM = i\}. \qquad (3)$$

The performance index we will investigate is the VMM availability $A(t)$ that can be defined as:

$$A(t) = Pr\{\text{VMM is in state } s_{wr} \text{ at time } t\}. \qquad (4)$$

In particular, we will study how to optimally set the rejuvenation timer in order to maximize the steady-state VMM availability $A(\infty)$, defined as:

$$A(\infty) = \lim_{t \to \infty} A(t). \qquad (5)$$

4. ANALYSIS CHALLENGES

According to the finite state machine of Figure 2, a VMM can be in four possible states with five possible transitions or events among such states. A state space model can be easily obtained from such finite state machine by stochastically characterizing the events through the corresponding random variables. However, as already described in Section 3, we want to take into account the impact of workload, represented by the number i of VMs managed by the VMM at a given instant t, on the software aging (eq. (1)) and, in general, on the time-to-event of some of the state transitions (eq. (2), (3)). The analysis of such a stochastic process is quite challenging given that it could be difficult to model and evaluate state space models with 2-variable CDFs.

A possible solution could be to duplicate the states of the finite state machines in Figure 2 thus generating a stochastic process whose events are associated to CDFs that only depend on time. Such a stochastic process is composed of $N+1$ sets of states each representing the evolution of the VMM life-cycle in the presence of a certain number of running VMs. Moreover, events e_{ar} and e_{dp} represent the arrival and departure of a VM, respectively.

In order to analytically solve the stochastic process under consideration, attention needs to be put in the following aspects. First of all, it is necessary to identify the VMM unreliability CDFs at each specific load condition in isolation, i.e., when the number of running VMs does not vary. In other words, referring to eq. (1), we need to identify CDFs $F_{fl}(t,i) = F^i_{fl}(t)$ thus obtaining a set of unreliability CDFs $\mathcal{F}_{fl} = \{F^i_{fl}(t) : 0 \leq i \leq N\}$. A similar operation needs to be done for the CDFs of the other system events whose behavior depends on the load (repair, resume, and departure events), thus obtaining the sets $\mathcal{F}_{rp} = \{F^i_{rp}(t) : 0 \leq i \leq N\}$, $\mathcal{F}_{rs} = \{F^i_{rs}(t) : 0 \leq i \leq N\}$, and $\mathcal{F}_{dp} = \{F^i_{dp}(t) : 0 \leq i \leq N\}$.

In order to represent load dependent conditions in analytic terms, one of the most widely used approach is the *proportional hazard model* (PHM) [19]. In a PHM, the changes in reliability corresponding to changes in load conditions are expressed in terms of a multiplicative effect on the failure rate. The simplest PHM expresses such effect by a constant thus obtaining for each possible load configuration a failure rate that is proportional to the one taken as a reference and usually associated to a certain amount of load that is considered as unitary. In the specific case of the system under analysis, assuming $\lambda^1_{fl}(t) = -\frac{d(F^1_{fl}(t))/dt}{F^1_{fl}(t)}$ the failure rate of the VMM on top of which one VM is running, we can express the failure rate of the VMM with $i > 1$

VMs as $\lambda^i_{fl}(t) = -\frac{d(F^i_{fl}(t))/dt}{F^i_{fl}(t)} = i \cdot \lambda^1_{fl}(t)$. Similarly, the case in which no VMs are running can be characterized by $\lambda^0_{fl}(t) = k\lambda^1_{fl}(t)$ with $0 \leq k \leq 1$.

Following a similar reasoning, given the rejuvenation rate $\eta^1_{rs}(t) = -\frac{d(F^1_{rs}(t))/dt}{F^1_{rs}(t)}$, the repair rate $\mu_{rp}(t) = -\frac{d(F^1_{rp}(t))/dt}{F^1_{rp}(t)}$, and the departure rate $\mu^1_{dp}(t) = -\frac{d(F^1_{dp}(t))/dt}{F^1_{dp}(t)}$ for a VMM with a single VM as load, the rates in the case of $i > 1$ VMs can be expressed as $\eta^i_{rs}(t) = \eta^1_{rs}(t)/i$, $\mu^i_{rp}(t) = \mu^1_{rp}(t)/i$, and $\mu^i_{dp}(t) = i \cdot \mu^1_{dp}(t)$ respectively. Assuming the repair, the resume, and the departure events to be exponentially distributed, we have $\eta^i_{rs}(t) = \eta^i_{rs} = \eta^1_{rs}(t)/i = \eta^1_{rs}/i$, $\mu^i_{rp}(t) = \mu^i_{rp} = \mu^1_{rp}(t)/i = \mu^1_{rp}/i$, and $\mu^i_{dp}(t) = \mu^i_{dp} = i \cdot \mu^1_{dp}(t) = i \cdot \mu^1_{dp}$.

Finally, events e_{dt} and e_{ar}, exponentially distributed and independent from the load as an assumption, will be associated to the rates λ_{dt} and λ_{ar}, respectively.

According to these assumptions, for a complete specification of the model, it is necessary to adequately specify what happens when a transition occurs from a workload condition to another one. During such transitions, two events are involved since they are concurrently enabled: the failure event e_{fl} and the trigger event e_{tr}.

With respect to event e_{fl}, we have to guarantee that the age level reached by the VMM, i.e., its level of degradation, is preserved during the transition thus applying the conservation of reliability principle [20, 21, 22]. In a similar way, the memory related to the trigger event e_{tr} has to be preserved according to the time-based rejuvenation policy applied.

5. MODEL EVALUATION

The stochastic process described in the previous section is in general non-Markovian and non-homogeneous in time [22]. More complex and powerful techniques than Markov chains or Markov reward processes are necessary to analyze such process, such as Markov additive processes [21], semi-Markov processes and renewal theory [23]. Starting from such model, in the following we describe a technique based on CPH distributions and Kronecker algebra. The unreliability and the timeout of the VMM, under different load conditions, are represented by CPHs, moving the problem towards the solution of an expanded CTMC. Kronecker algebra is therefore used in order to overcome the well-known state space explosion problem, since the infinitesimal generator matrix of the expanded CTMC is never generated nor stored as a whole, but it is algorithmically evaluated on-the-fly in a block-wise fashion when needed during the model solution. In this way, only few information about the stochastic process is permanently stored, with consequent memory saving.

In this section we recall the technique we proposed in [24] for the evaluation of reliability and availability of dynamic systems that will be used in order to manage the model introduced in Section 4.

5.1 Analysis Technique

The use of PH distributions was popularized in 1981 by Neuts [25] who formally defined a PH distribution as the distribution of the time until absorption in a finite state Markov chain with a single absorbing state. In case of CTMC, CPH distributions are characterized.

More specifically, let us consider a CTMC χ with n transient states and a single absorbing state (labeled $n+1$) whose infinitesimal generator matrix $\hat{\mathbf{G}}$ of dimensions $n+1 \times n+1$ is in the form:

$$\hat{\mathbf{G}} = \left[\begin{array}{c|c} \mathbf{G} & \mathbf{U} \\ \hline \mathbf{0} & 0 \end{array}\right]$$

where $\mathbf{G}$ is a $n \times n$ matrix that describes the transient behavior of the CTMC and $\mathbf{U}$ is a n-dimension column vector grouping the transition rates to state $n+1$. Moreover, let us suppose that the chain is started with an initial probability vector $\boldsymbol{\pi}(0) = [\boldsymbol{\alpha}, \alpha_{n+1}]$ such that $\alpha_{n+1} = 1 - \sum_{i=1}^{n} \alpha_i$. Then, we say that a random variable T is distributed according to the CPH distribution with representation $(\boldsymbol{\alpha}, \mathbf{G})$ and order n if its cumulative distribution function (CDF) $H_T(t)$ is the probability to reach the absorbing state of χ; a well known result is that $H_T(t)$ is expressed as:

$$H_T(t) = 1 - \boldsymbol{\alpha} \cdot e^{\mathbf{G}t} \cdot \mathbf{1}, \quad t \geq 0 \tag{6}$$

The model we want to analyze is characterized by dependencies among events. In fact, when a new VM request is submitted to the VMM or when a running VM completes its computation and leaves the system, the distributions describing the age process of the VMM (the expiration of the rejuvenation timer) changes and the reliability level (timer level) needs to be preserved. In the context of dynamic reliability systems, we say that when a *driver* event occurs it can affect the behavior of a *target* event by changing its CDF in the new reached state. Such type of interaction between events can be described in terms of CPH distributions, as well.

In particular, for a matter of semplicity, let us focus on the case in which the behavior of a dependent event is characterized by a bivariate CDF, as follows:

$$H_{XY}(x, y) = \begin{cases} H'_X(x) & x \leq y \\ H''_X(x) & x > y \end{cases} \tag{7}$$

where X is the random variable characterizing the target event, while Y is the random variable associated to the driver event that triggers the dependency at time instant y. $H'_X(x)$ and $H''_X(x)$ are the CDFs associated to the dependent event in isolation and under dependency, respectively and, as stated above, are expressed by CPH distributions.

If the random variable X is a time to failure and the *conservation of reliability principle* [20, 21] has to be applied, then at change point y the reliability level that has been reached has to be preserved. Without losing in generality, being the same reasoning applicable to other quantities, let us suppose that at time instant y the time-to-failure CDF of a component C changes from $H'(t)$ to $H''(t)$ and let $(\boldsymbol{\alpha}', \mathbf{G}')$ and $(\boldsymbol{\alpha}'', \mathbf{G}'')$ be the $n' \times n'$ and $n'' \times n''$ matrices associated to the corresponding CPHs, respectively, such that:

$$H'(t) = 1 - \boldsymbol{\alpha}' \cdot e^{\mathbf{G}'t} \cdot \mathbf{1}'$$

and

$$H''(t) = 1 - \boldsymbol{\alpha}'' \cdot e^{\mathbf{G}''t} \cdot \mathbf{1}''.$$

In other words, the aging process of the component evolves with CPH $(\boldsymbol{\alpha}', \mathbf{G}')$ for $t \leq y$. At time instant $t = y$, the driver trigger event occurs and the aging process of the component changes its evolution to CPH $(\boldsymbol{\alpha}'', \mathbf{G}'')$ for $t > y$.

Keeping memory on *age* for C at change point y means to find a translation in time of CDF $H''(t)$ such that

$$H'(y) = H''(y+\tau) \tag{8}$$

and such that the aging process of the component can continue to evolve according to $H''(t)$, preserving the *age* level reached so far. In terms of CPH distributions, we need to find an equivalent time $y+\tau$ such that:

$$\boldsymbol{\alpha}' \cdot e^{\mathbf{G}'y} \cdot \mathbf{1}' = \boldsymbol{\alpha}'' \cdot e^{\mathbf{G}''(y+\tau)} \cdot \mathbf{1}'' \tag{9}$$

Since the proposed technique is based on matrices manipulation, as we will describe in the following, it is necessary to specify a *memory matrix* $\mathbf{M}(y)$ that represents the transition from the CPH representing $H'(t)$ to the one representing $H''(t)$, at time instant y, and such that the eq. (9) holds due to the conservation of age principle. Expressing this in terms of probability vectors $\boldsymbol{\alpha} \cdot e^{\mathbf{G}t}$ we have:

$$\boldsymbol{\alpha}' \cdot e^{\mathbf{G}'y} \cdot \mathbf{M}(y) = \boldsymbol{\alpha}'' \cdot e^{\mathbf{G}''(y+\tau)} \tag{10}$$

The computation of the memory matrix is not straightforward, especially considering that it is time dependent. However, in case the two CPHs have the same number of stages and equal initial vectors ($\boldsymbol{\alpha}' = \boldsymbol{\alpha}''$), a possible solution of eq. (10) is given by eq. (11),

$$e^{\mathbf{G}'y} \cdot \mathbf{M}(y) = e^{\mathbf{G}''(y+\tau)} \Rightarrow \mathbf{M}(y) = e^{-\mathbf{G}'y} \cdot e^{\mathbf{G}''(y+\tau)} \tag{11}$$

where τ is computed according to eq. (8) by $\tau = H''^{(-1)}(H'(y)) - y$.

5.2 From CPH to Kronecker Algebra

Let us consider a discrete-state discrete-event model and let S be the system state space and $\boldsymbol{\varepsilon}$ the set of CPH distributed system events. Following the state space expansion approach [25], the stochastic process can be represented by an expanded CTMC. Such CTMC is composed of $||S||$ macro-states and it is characterized by a $||S|| \times ||S||$ block infinitesimal generator matrix $\mathbf{Q}$ in which:

- the generic diagonal block $\mathbf{Q}_{n,n}$ ($1 < n < ||S||$) is a square matrix that describes the evolution of the CTMC inside the macro-state related to state n, and it depends on the possible events that are enabled in such state;
- the generic off-diagonal block $\mathbf{Q}_{n,m}$ ($1 < n < ||S||$, $1 < m < ||S||$) describes the transition from the macro-state related to state n to that related to state m, and it depends on the events occurred in state n and on the possible events that are still able to occur in state m.

The state space expansion approach could be easily used to represent the VMM model assuming the system events are associated to CPH distributions. The main drawback of the state space expansion approach is the explosion of the state space. This limits the applicability of the approach, thus reducing its benefits and potentialities. A possible solution to overcome such problems is to recur to the use of Kronecker algebra [25, 26, 27], a well known and effective technique able to provide a compact representation of CTMCs.

By exploiting Kronecker algebra, matrix $\mathbf{Q}$ does not need to be generated and stored as a whole, but can be symbolically represented through Kronecker expressions and algorithmically evaluated on-the-fly when needed with consequent extreme memory saving [28, 29]. Moreover, this approach is particularly suitable in models where the conservation of age principle has to be applied, since the overall CTMC model associated to such a system is generally non-homogeneous in time due to the presence of the time-dependent memory matrices $\mathbf{M}(y)$ representing the age process. In fact, by applying the proposed technique, the values of $\mathbf{M}(y)$ are evaluated on-the-fly only when required, and so the algorithm works for non-homogeneous CTMC, as well as in the homogeneous case. Going into details of matrix $\mathbf{Q}$, its blocks have the following form:

$$\mathbf{Q}_{n,n} = \bigoplus_{1<e<||\varepsilon||} \mathbf{Q}_e \tag{12}$$

$$\mathbf{Q}_{n,m} = \bigotimes_{1<e<||\varepsilon||} \mathbf{Q}_e \tag{13}$$

In other words, the diagonal blocks are computed as Kronecker sums (off-diagonal blocks are computed as Kronecker products) of a series of matrices $\mathbf{Q}_e$, each associated to one of the system events.

5.3 The VMM model

According to the state space expansion approach, each state in the model described in previous sections can be expanded by introducing the CPH distributions associated to the system events and the corresponding interactions. Moreover, Kronecker algebra could be used in order to deal with the state space explosion problem. In particular, the expanded CTMC associated to the VMM model can be described by an $(N+1)\cdot 4$ infinitesimal generator block matrix $\mathbf{Q}$, where the generic block $\mathbf{Q}_{n,m}$ ($n, m = 0, \ldots, (N+1)\cdot 4-1$) can be represented by a Kronecker sum or product of basic matrices each of which is associated to a system event.

The state space model of the VMM is characterize by the presence of $N+1$ sets of states each of which is associated to a particular number of VMs instantiated on top of the VMM under analysis. The i^{th} set of states, associated to the presence of i VMs, is composed of 4 states, say s^i_{wr}, s^i_{nd}, s^i_{df}, s^i_{rj}.

In the following, we indicate with ($\boldsymbol{\alpha}^i_{fl}$,$\mathbf{G}^i_{fl}$) the CPH representation of the CDF $F^i_{fl}(t)$ associated to the failure event e_{fl} in case of i instantiated VMs. ($\boldsymbol{\alpha}^i_{dt}$,$\mathbf{G}^i_{dt}$) will indicate the CPH representation of the CDF $F^i_{dt}(t)$ associated to the detection event e_{dt}, while ($\boldsymbol{\alpha}^i_{rp}$,$\mathbf{G}^i_{rp}$) is the CPH representation of event e_{rp}'s CDF $F^i_{rp}(t)$. Moreover, we indicate with ($\boldsymbol{\alpha}^i_{tr}$,$\mathbf{G}^i_{tr}$) the CPH representation of the CDF $F^i_{tr}(t)$ associated to the trigger event e_{tr} and with ($\boldsymbol{\alpha}^i_{rs}$,$\mathbf{G}^i_{rs}$) the CPH associated to the resume event e^i_{rs}'s CDF $F^i_{rs}(t)$. Finally, ($\boldsymbol{\alpha}^i_{ar}$,$\mathbf{G}^i_{ar}$) and ($\boldsymbol{\alpha}^i_{dp}$,$\mathbf{G}^i_{dp}$) are the CPH representation of the CDFs $F^i_{ar}(t)$ and $F^i_{dp}(t)$ associated to event e_{ar} and e_{dp}, respectively. The aim of this section is to give some examples about how to obtain matrices $\mathbf{Q}_{n,m}$ in such states.

Let us consider state s^i_{wr}. In such state, four events concurrently evolve accordingly to their associated CDFs: e_{fl}, e_{tr}, e_{ar}, e_{dp}. The other events are disabled. For such a reason, matrix block $\mathbf{Q}_{4i,4i}$ exhibits the form:

$$\mathbf{Q}_{4i,4i} = \mathbf{G}^i_{fl} \oplus [0] \oplus [0] \oplus \mathbf{G}^i_{tr} \oplus [0] \oplus \mathbf{G}^i_{ar} \oplus \mathbf{G}^i_{dp} \tag{14}$$

where matrix [0] (i.e., a matrix with a single element equal to 0) associated to events e_{dt}, e_{rp}, and e_{rs} is the neutral element

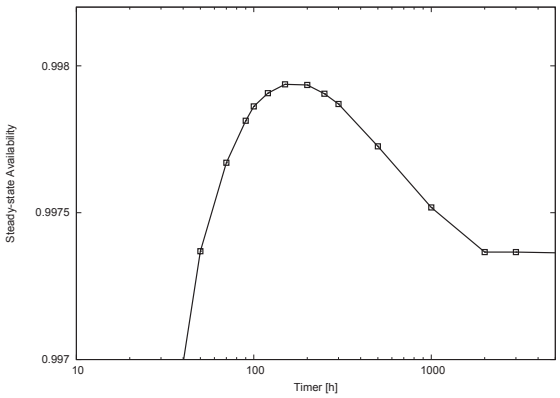

Figure 3: Steady state availability $A(\infty)$ of the VMM.

for the Kronecker sum operator, while matrices $\mathbf{G}^i_{ar} = [\lambda_{ar}]$ and $\mathbf{G}^i_{dp} = [i \cdot \lambda_{dp}]$ because we can assume that such events are exponentially distributed.

Particularly interesting is the case of the non-diagonal blocks of matrix $\mathbf{Q}$, given that the conservation of age principle needs to be taken into consideration while setting the corresponding matrices. In particular, matrices associated to the transitions from state s^i_{wr} to states s^{i-1}_{wr} or s^{i+1}_{wr} deserve specific attention. In the first case, event e_{dp} is enabled and it is the one that causes the transition, being enabled again in the arrival state. Thus, the stochastic process leaves the phases of the CPH associated to event e_{dp} with the rates specified in matrix $\mathbf{U}^i_{dp}$ and enters into the corresponding phases in the arrival state with the initial probabilities given by vector $\boldsymbol{\alpha}^{i-1}_{dp}$. However, we can assume that event e_{dp} is exponentially distributed with rate $i \cdot \lambda_{dp}$ so the associated matrix is simply $[i \cdot \lambda_{dp}]$ i.e., a matrix with a single element equal to the rate associated to the event in the leaving state.

On the other hand, events e_{ar}, e_{fl}, and e_{tr} are enabled in both s^i_{wr} and s^{i-1}_{wr}, but their distributions change according to the load variation. As described earlier, in this case a memory matrix is necessary to keep memory of the quantity coded by the CPH phases associated to the events. In the case of event e_{fl} such quantity is the reliability, while in the case of event e_{tr} the quantity is represented by the level reached by the rejuvenation timer. In the case of event e_{ar}, we assume an exponential distribution and the associated matrix is simply [1]. Finally, events e_{dt}, e_{rp}, and e_{rs} are not involved in the transitions and the associated matrix is equal to [1]. For the above reported reasons, matrix block $\mathbf{Q}_{4i,4i-1}$ exhibits the form:

$$\begin{aligned}\mathbf{Q}_{4i,4(i-1)} &= \mathbf{M}_{fl}^{4i\to 4(i-1)}(t) \otimes [1] \otimes [1] \otimes \\ &\quad \mathbf{M}_{tr}^{4i\to 4(i-1)}(t) \otimes [1] \otimes [1] \otimes [i \cdot \lambda_{dp}] \quad (15)\end{aligned}$$

Other matrices can be easily computed by applying similar reasoning.

6. RESULTS

This section intends to show the effectiveness of the proposed technique by applying it to a specific VMM configuration. In such example, we decided to limit the maximum number of VMs the VMM can accept to 4 (i.e., $N = 4$). In this way, five load conditions can be identified for the VMM: 0, 1, 2, 3, and 4 VMs.

We adopted the parameters shown in Table 1. With regards to the requests, exponentially distributed arrivals and departures are considered with rates γ and δ, respectively.

Symbol	*Description*	*Value*
α	Failure CDF scale parameter	1000
β	Failure CDF shape parameter	1.3
μ	Repair rate	1
η	Detection rate	1
γ	Arrival rate	1
δ	Departure rate	1

Table 1: Parameters of the case study.

Since one of the most used CDF in reliability context is the Weibull distribution, we choose a Weibull CDF to represent the VMM aging, fixing the parameters $\alpha = 1000$ and $\beta = 1.3$. In this way, the CDF characterizing the VMM aging with only one VM is identified.

As discussed in Section 4, a PHM is considered to characterize the VMM aging in the different load conditions. More specifically, we assume that $\lambda_i(t) = i * \lambda(t)$ for $i = 1, .., 4$ and therefore, since the failure rate for a Weibull CDF is $\lambda(t) = \beta/\alpha \left(\frac{t}{\alpha}\right)^{\beta-1}$ and assuming $\beta_i = \beta$, we can easily obtain that the PHM condition can be expressed just in terms of the scale parameter as $\alpha_i = \frac{\alpha}{i^{1/\beta}}$. Moreover, since the recovery and the rejuvenation rates are constant, we have that $\eta_i = \eta/i$ and $\mu_i = \mu/i$ with $i = 1, .., 4$.

In case there are no VM instantiated in the VMM ($i = 0$) we assume that the VMM does not degrade and therefore it is necessary to just keep memory of the age or of the level of degradation reached at the changing point from the VMM load condition of 1 VM to 0 VM. In numerical terms this means that $\lambda_0(t) = \eta_0(t) = \mu_0(t) = 0$.

The results, in terms of the steady state availability $A(\infty)$, obtained by evaluating the model when varying the rejuvenation timer are shown in Figure 3. The trend initially increases by increasing the timer till a maximum value of the steady state availability, and therefore slowly decreases to an asymptotic value; this value represents the steady state availability considering the repair only without any rejuvenation policy.

The graphs clearly highlight that the steady state availability obtained by applying the time-based rejuvenation

policy is higher than the one obtained by simply repairing the system when a failure occurs. The obtained results thus justify the interest in the topic and encourage future works. Moreover, we demonstrated how our analytic technique is able to manage complex scenarios with non-exponentially distributed failure rates.

7. CONCLUSIONS

We proposed an innovative approach to model software aging in Cloud systems. Time and load dependent degradations have been modeled through CPH type distributions and a compact and efficient representation was given through Kronecker algebra. Preliminary results were provided by analytically solving the resulting model.

This work is one of the first attempt to apply the concept of software rejuvenation to Cloud computing and paves the way toward an entire set of new investigations to fully characterize Cloud systems thus allowing to deliver more reliable and available QoS-guaranteed services. We believe that the proposed approach is very innovative both at the Cloud infrastructure level and from the modeling point-of-view, as the proposed analytical technique can be used to characterize software aging at large.

8. REFERENCES

[1] K. Vaidyanathan and K. S. Trivedi, "A comprehensive model for software rejuvenation," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 124–137, 2005.

[2] S. Garg, A. Puliafito, M. Telek, and K. Trivedi, "Analysis of preventive maintenance in transactions based software systems," *Computers, IEEE Transactions on*, vol. 47, no. 1, pp. 96 –107, Jan. 1998.

[3] K. Vaidyanathan and K. Trivedi, "A measurement-based model for estimation of resource exhaustion in operational software systems," in *Software Reliability Engineering, 1999. Proceedings. 10th International Symposium on*, 1999, pp. 84 –93.

[4] S. Garg, A. Puliafito, M. Telek, and K. Trivedi, "Analysis of software rejuvenation using markov regenerative stochastic petri net," in *Software Reliability Engineering, 1995. Proceedings., Sixth International Symposium on*, Oct. 1995, pp. 180 –187.

[5] S. Garg, A. van Moorsel, K. Vaidyanathan, and K. Trivedi, "A methodology for detection and estimation of software aging," in *Software Reliability Engineering, 1998. Proceedings. The Ninth International Symposium on*, Nov. 1998, pp. 283 –292.

[6] P. Mell and T. Grance, "The nist definition of cloud computing," *National Institute of Standards and Technology*, vol. 53, no. 6, p. 50, 2009. [Online]. Available: http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc

[7] L. Wang, J. Tao, M. Kunze, A. Castellanos, D. Kramer, and W. Karl, "Scientific cloud computing: Early definition and experience," in *Int. Conf. on High Performance Computing and Communications, 2008 (HPCC '08).*, sept. 2008, pp. 825–830.

[8] P. Mell and T. Grance, "The nist definition of cloud computing," NIST Special Publication, Tech. Rep. 800-145, Jan. 2011.

[9] L. Youseff, M. Butrico, and D. Da Silva, "Toward a unified ontology of cloud computing," in *Grid Computing Environments Workshop, 2008. GCE '08*, Nov. 2008, pp. 1–10.

[10] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via vm multiplexing," in *Proceedings of the 7th international conference on Autonomic computing*, ser. ICAC '10. New York, NY, USA: ACM, 2010, pp. 11–20. [Online]. Available: http://doi.acm.org/10.1145/1809049.1809052

[11] H. Liu, H. Jin, X. Liao, C. Yu, and C.-Z. Xu, "Live virtual machine migration via asynchronous replication and state synchronization," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 12, pp. 1986 –1999, dec. 2011.

[12] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman, E. Levy, A. Maraschini, P. Massonet, H. MuÃś andoz, and G. Tofetti, "Reservoir - when one cloud is not enough," *Computer*, vol. 44, no. 3, pp. 44 –51, march 2011.

[13] R. Buyya, R. Ranjan, and R. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," in *High Performance Computing Simulation, 2009. HPCS '09. International Conference on*, june 2009, pp. 1 –11.

[14] A. Iosup, N. Yigitbasi, and D. Epema, "On the performance variability of production cloud services," in *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, may 2011, pp. 104 –113.

[15] V. Stantchev, "Performance evaluation of cloud computing offerings," in *Advanced Engineering Computing and Applications in Sciences, 2009. ADVCOMP '09. Third International Conference on*, oct. 2009, pp. 187 –192.

[16] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing," in *Cloud Computing*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 34, ch. 9, pp. 115–131.

[17] H. Khazaei, J. Misic, and V. Misic, "Performance analysis of cloud computing centers using m/g/m/m + r queueing systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2011.

[18] R. Ghosh, K. Trivedi, V. Naik, and D. S. Kim, "End-to-end performability analysis for infrastructure-as-a-service cloud: An interacting stochastic models approach," in *Dependable Computing (PRDC), 2010 IEEE 16th Pacific Rim International Symposium on*, dec. 2010, pp. 125 –132.

[19] D. R. Cox, "Regression Models and Life-Tables," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 34, no. 2, pp. 187–220, 1972.

[20] D. Kececioglu, *Reliability Engineering Handbook (Vol. 1 and 2)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1991.

[21] M. S. Finkelstein, "Wearing-out of components in a variable environment," *Reliability Engineering & System Safety*, vol. 66, no. 3, pp. 235 – 242, 1999.

[22] S. Distefano, F. Longo, and M. Scarpa, "Availability assessment of ha standby redundant clusters," in *Reliable Distributed Systems, 2010 29th IEEE Symposium on*, nov. 2010, pp. 265–274.

[23] D. R. Cox, *Renewal Theory*. London, England: Methuen & Co. Ltd, 1967.

[24] S. Distefano, F. Longo, and M. Scarpa, "Symbolic representation techniques in dynamic reliability evaluation," *High-Assurance Systems Engineering, IEEE International Symposium on*, pp. 45–53, 2010.

[25] M. F. Neuts, *Matrix-geometric solutions in stochastic models: an algorithmic approach*. Johns Hopkins University Press, Baltimore, 1981.

[26] M. Scarpa, "Non Markovian Stochastic Petri Nets with Concurrent Generally Distributed Transtions," Ph.D. dissertation, University of Turin, 1999.

[27] R. Pérez-Ocón and J. E. R. Castro, "Two models for a repairable two-system with phase-type sojourn time distributions," *Reliability Engineering & System Safety*, vol. 84, no. 3, pp. 253–260, 2004.

[28] F. Longo and M. Scarpa, "Applying symbolic techniques to the representation of non-markovian models with continuous ph distributions," in *EPEW '09: Proceedings of the 6th European Performance Engineering Workshop on Computer Performance Engineering*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 44–58.

[29] A. Bobbio and M. Scarpa, "Kronecker representation of stochastic petri nets with discrete ph distributions," in *Proc. Third IEEE Ann. Int'l Computer Performance and Dependability Symp. (IPDS '98)*, 1998.