Chapter 1

Aging in Virtualized Environments

Roberto Pietrantuono, Domenico Cotroneo, Stefano Russo

Dipartimento di Ingegneria Elettrica e delle Tecnologie dell’Informazione
Università degli Studi di Napoli Federico II, Napoli, Italy
{roberto.pietrantuono, cotroneo, stefano.russo}@unina.it

With virtualization technologies becoming a predominant paradigm of computing, the analysis of long-running performance and aging issues in virtualized environments is a strong need. Software aging in such environments can have a severe impact on millions of customers of business-critical applications, as well as on end-users of more critical (e.g., mission-critical) systems where virtualization is also vastly being adopted.

This chapter discusses the main existing approaches to study the aging phenomena in virtualized environments, and the corresponding rejuvenation techniques. It targets those contexts where virtual machines are massively adopted—such as cloud computing platforms—as well as relevant virtualization technologies.

The chapter covers first model-based methods for the analysis of resource consumption and/or performance degradation, as well as for the optimal scheduling of rejuvenation in virtualized systems. Then, measurement-based approaches are discussed, where the statistical analysis of monitored health indicators is exploited to detect and forecast aging trends in the considered contexts.

1. Introduction

This chapter explores the main approaches to deal with software aging analysis and with rejuvenation in virtualized environments. Virtualization is a key technology for today’s IT organizations. Virtualization solutions greatly ease the build and deploy of applications, increase their runtime availability, and alleviate several manual tasks—saving people time to deliver greater value. Moreover, by reducing the number of physical machines, it drastically lowers the infrastructure and power consumption cost, the maintenance and IT staffing cost, the backup and recovery cost. As for instance, a Forrester study\textsuperscript{a} found that Red Hat Virtualization speeds up virtualization tasks and improves performance: \textit{i}) return on investment of 103\% over 3 years; \textit{ii}) 10\% to 20\% of infrastructure developer’s time saved through increased virtualization task and process efficiency; \textit{iii}) payback period of 5.6 months.

\textsuperscript{a} “The Total Economic Impact of Red Hat Virtualization”, available at: \url{www.redhat.com/it/engage/efficiency-of-virtualization}.
A main advantage of virtualization is its ability of making better use of existing resources, enabling multiple systems (i.e., operating system, system-level software and application-level software) running on the same physical machine, sharing and dynamically acquiring/releasing the same set of resources (e.g., CPU, memory, storage) depending on the runtime need. Systems/services making massive use of virtualization in this sense are those based on cloud computing. In this case, virtualization is implemented by means of an intermediate software layer called hypervisor or Virtual Machine Monitor (VMM), responsible for managing the execution of virtual machines and for the interaction with the hardware resources by separating virtual from physical layer. Another advantage of virtualization is a much greater portability of applications (and of entire systems) across various physical environments. Besides cloud computing, a former successful example of software virtualization technology, with a strong focus on portability, is the Java Virtual Machine (JVM).

As many enterprises employ virtualization to run their services, the problem of performance degradation and, more generally of software aging, becomes crucial. Aging has been analyzed by researchers in many modern virtualized environment, wherein long executions leads to errors accumulation and gradual performance decrease. Conventional approaches for aging detection and estimation have been used, which can be roughly divided into model-based and measurement-based approaches. At the same time, besides aging detection and estimation (which allow to determine if and when an aging failure is likely to occur), the problem of how to rejuvenate a virtualized system gained considerable attention by researchers. Indeed, rejuvenation is a key high-availability technique in this context, since the advantages given by VMs as application “containers” has led to novel and lightweight rejuvenation strategies (based on restarts, migration and failover) able of guaranteeing high levels of continuity of service in presence of aging problems.

The next Section introduces the general characteristics of SAR research in virtualized environments; then, the techniques will be surveyed according to the usual categorization of model-based and measurement-based aging analysis (Section 3 and 4). Hybrid approaches combining both types of techniques are presented in Section 5. Section 6 presents the main techniques to perform rejuvenation in virtualized environments. Finally, Section 7 draws the main conclusions.

2. Aging analysis and rejuvenation for virtualized environments

The analysis of software aging deals usually with two main tasks, the detection and the estimation of aging trends through one or more aging indicators. These are related to resource exhaustion (e.g., memory, storage, operating system resources, energy), and/or to user-perceived performance (e.g., response time, latency, served transactions, throughput). They characterize a degradation phenomenon manifesting itself often at system level (i.e., within the execution environment of the application software) and eventually at user level.
In virtualized environments, the system level – namely, the execution environment - includes not only the physical machine with its operating system (OS) and low-level software, but also the virtual environment, such as the Virtual Machine (VM) on which the software is running and/or the whole virtualization technology (including the Virtual Machine Monitor (VMM), the most important layer responsible for creating a VM environment for an OS and its applications). Figure 1 sketches a generic architecture of a virtualized environment. Virtualization can be implemented: i) via native VMMs (also known as bare-metal hypervisors, such as Xen, VmWare ESX Oracle Vm Server), in which there is no “host” OS but the VMM runs on and controls directly the hardware, or ii) via "hosted" solutions (also known as hosted hypervisors, like, for instance, VMware Workstation, VirtualBox, Parallels), in which the access to hardware is managed via the host OS. On top of the VMM, there are VMs: depending on the proactive/reactive fault tolerance strategy (including rejuvenation), there can be standby VMs, which can take over the active VMs in case of failure or upon a rejuvenation request. In a more general scheme (e.g., in virtualized data centres), there may be other physical nodes and a VM can be migrated to a different node if, for instance, the hosting node (or part thereof, such as the VMM) fails or needs to be rejuvenated.

In such a scenario, the execution environment is quite different from a traditional non-virtualized environment. From the SAR perspective, this affects: i) the choice of the most appropriate strategy for aging analysis (i.e., how to predict the most likely time of aging failure occurrence in order to figure out when to schedule a rejuvenation action), and ii) the consequent rejuvenation action to perform (i.e., how rejuvenation should be performed).

The ultimate goal of aging analysis – encompassing detection and estimation – is to determine the most likely time of aging failure occurrence, so as to figure...
out *when* to schedule a rejuvenation action. This is done either through analytical modelling – formulating the problem via stochastic models and finding the best rejuvenation time maximizing the system availability/survivability/performability – or by monitoring system’s health indicators and apply statistical inference techniques on time series to forecast future trends. These approaches, also known as **model-based** and **measurement-based**, respectively, are sometimes combined into a hybrid strategy, wherein models are properly parametrized by field data. In the next Sections, aging analysis and then rejuvenation techniques proposed for virtualized environments are explained.

3. Model-based analysis techniques

**Model-based** aging analysis techniques for virtualized environments share the following characteristics:

- The software applications within a more or less complex virtualized environment (with one or more VMs, physical hosts, and the associated rejuvenation strategies) are most commonly modeled by Markov models, including continuous-time Markov chains (CTMC), homogeneous and non-homogeneous, Markov regenerative processes (MRGM), semi-Markov processes (SMP), stochastic Petri nets (SPN) and stochastic reward nets (SRN), that capture the aging behaviour at different level of abstractions (application or virtualized environment or physical environment level), and, sometimes, the non-aging failing behavior too.¹
- They do not usually refer to specific aging indicators, since the model is assumed to work regardless of which resource is being depleted or of the user-perceived metric. Even when the model refers to a specific indicator, it can easily be applied to another aging indicators. The attribute of interest is related to the final (user-perceived) metric of availability (or related metrics, such as survivability or performability).
- The evaluation of alternatives is one of the objective of model-based analyses: in most of cases, model-based techniques are associated with the evaluation of alternative rejuvenation techniques, at various levels, such as at application, at VM, at VMM/cloud platform, or at node level. Indeed, exploring many such alternatives with a measurement-based approach in a real setting would be too onerous. The adopted models very often encompass the rejuvenation action being performed, for which proper parameters are considered (e.g., time/cost for rejuvenation) for the availability computation on different rejuvenation alternatives. Rejuvenation is addressed by many more studies compared to the SAR literature for generic systems, since, by virtualization, there are more options for rejuvenation, able to drastically reduced the cost, which called for researchers’ investigation, as explained later in this Section.
• Model-based techniques are usually validated by means of numerical illustrations or via simulation, since collection of runtime data for all or part of the cases being modelled is expensive and not enough representative of what is modelled. This regards model-based works in all the SAR research area, and is confirmed in the SAR research for virtualized environment too.

• Models are, by their nature, applicable to many systems; their drawback are in the simplifying assumptions, such as those about the underlying stochastic distributions that characterize the system. This too is not different in the case of virtualized environment SAR research area.

An example of model-based approach, from Chang et al.\(^2\) is shown in Figure 2.

Fig. 2.: A model of aging-related service failure/recovery in a virtualized system (from Chang et al.\(^2\))

In this CTMC model, the initial state is conditioned to be the service breakdown. It is assumed that no other failure occurs during the system recovery. When the service breakdown occurs due to the active VM rejuvenation or the active VM reboot caused by an aging-related bug (VMR), a standby VM on the same host is selected. The failover requires a time \(1/\beta_0\); the VM restart requires \(1/\beta_2\). VMC represents that service breakdown due to VM crash caused by non-aging Mandelbug. The failover takes a time \(1/\beta_1\) (larger than \(1/\beta_0\)); the VM is repaired with time \(1/\beta_3\). After repair, the state is “VM ready to reboot”, and the service is fully recovered after \(1/\beta_2\). Similarly, VMMR is the service breakdown due to the VMM rejuvenation, and VMMC is that service breakdown caused by a VMM crash. Live VM migration is applied in these cases, with different time delays. In this multi-phase CTMC model, transient survivability is considered as measure of interest, and VM/VMM rejuvenation is included as well as bug-caused failures of these components.
Model-based approaches mostly differ from the presented example in the type of model adopted, the configuration/architectural model (e.g., single vs multiple-host), and the attributes of interest analyzed. Several of such works also propose rejuvenation techniques, which will be however detailed in Section 6.

In the following, a brief overview is given about the main model-based techniques proposed in the literature.

Zheng et al.\(^3\) proposes a single-phase version of the approach proposed by Chang et al.\(^2\) to model again survivability. Silva et al.\(^4\) evaluate survivability of cloud computing system by exploiting Petri nets rather than CTMC.

Xu et al.\(^5\) and Rezaei et al.\(^6\) use Stochastic Reward Nets (SRNs) for availability modelling in a single-server virtualized system with rejuvenation applied at VMM level (by time-based policy), in combination with a measurement-based policy at VM level. SRNs for availability modeling are also used in the technique by Nguyen\(^1\) which considers various failure and recovery modes of multiple VMs and VMMs, and in the one by Han and Xu\(^7\) which considers three different rejuvenation policies (non-rejuvenation, time-based rejuvenation, and time and load-based delay rejuvenation) for single-server virtualization systems with multiple VMs on a single VMM. Similarly, Machida et al.\(^8,9\) use SRNs for cold-VM, warm-VM and VM-migration rejuvenation policies. A further example of multiple-host multiple VMs availability analysis is presented by Myint and Thein,\(^10\) who design a primary-standby servers model, encompassing a load balancer VM in each node responsible for monitoring resources, and a rejuvenation agent installed on each VM. SRNs are also used in the recent works by Escheikh\(^11,12\) where power management performability analysis, evaluating the impact on performance and energy consumption in virtualized systems, is studied.

The work by Thein et al.\(^13\) analyzes system availability with the time-based rejuvenation policy under different cluster configurations, 2 VMs hosted on a single physical server and 2 VMs per a physical server in dual physical servers. The same authors further present a software rejuvenation framework named VMSR to offer high availability for application server systems,\(^14\) proposing again a CTMC to model a single host, multiple VMs in the scheme with hot standby replicas.

Machida et al.\(^15,16\) present the analysis of job completion time under aging and rejuvenation of the VMM using a semi-Markov process. Okamura et al.\(^17\) present the transient analysis of the two main rejuvenation policies, Cold- and Warm-VM rejuvenation, by means of Markov Regenerative Stochastic Petri Nets (MRSPNs). Zhao et al.\(^18\) formulate the problem by a game method, representing the different goals of a service provider (who wants to maximize availability) and a maintainer (who wants to minimize cost). Markov Renewal Processes (MRP) are used to determine the optimal rejuvenation schedule and compute the steady-state availability and maintenance cost. Melo et al.\(^19\) present a SPN model with of cloud availability under two migration-based rejuvenation strategies (with and without a test before migration).
Finally, some researchers proposed slightly different models for aging analysis: examples include the work by Melo et al.,\textsuperscript{20} who formulate an availability model considering live migration for VMM rejuvenation based on extended \textit{Deterministic Stochastic Petri Nets (DSPNs) and Reliability Block Diagrams (RBD)}, and the one by Rahme et al.,\textsuperscript{21,22} who use \textit{Dynamic Fault Trees (DFT)} to model cloud-based software rejuvenation.

4. Measurement-based analysis techniques

On the other side of the spectrum there are \textbf{measurement-based} approaches, which monitor and analyze values of aging indicators through probes at system level – related to system resources exhaustion, e.g., memory, storage - as well as at user level – e.g., response time, latency, throughput, violations of Service Level Agreements (SLAs). In a virtualized setting, measurement-based techniques are characterized by the following features:

- Indicators are measured at any layer of the virtualization technology stack. Specifically, monitoring can be done at application or OS level within the VM layer, at VM layer to probe the state and resource consumption of the VMs (e.g., for load balancing, scaling, VM migration and rejuvenation decisions), as well as at virtualization technology layer, most often referring to the VMM component. Besides system resources (CPU, memory, storage, network), other indicators of interest are related to VMs, for instance, the number of VM new allocations/releases, the time to start/stop VMs or to the time to migration – parameters of interest to assess the cost of possible rejuvenation. These metrics are collected during the runtime execution and are used to forecast future trends based on past observation, making decision accordingly about rejuvenation, as well as about load balancing and static or dynamic resources allocation.

- Data gathered are analyzed by one or more of the following techniques:
  - \textbf{Time Series Analysis}. It is based on trend detection and estimation of a set of aging indicators. Tests for trend detection are used to accept/reject the hypothesis of no trend in data (e.g., Mann-Kendall, t-student, Seasonal Kendall tests). Trend estimation can exploit many models, e.g., multiple linear regression, regression smoothing, Sen’s slope estimate procedure, autoregressive models, non-linear models. In the common case of presence of correlation among multiple aging indicators, data transformation, feature selection, or dimensionality reduction techniques are used; an example is the Principal Component Analysis (PCA) followed by regression.

A time-series analysis technique in virtualized systems is the one by Araujo et al.,\textsuperscript{23} who use the \textit{linear, quadratic, exponential growth}, and the \textit{Pearl-Reed logistic} models in order to schedule software rejuvenation.
tion properly. These models have been adopted for predicting memory consumption trends on the Eucalyptus cloud computing framework. Umesh et al.\textsuperscript{,24} also exploit time series models to identify (and forecast) software aging patterns of Windows active directory service. DeCelles et al.\textsuperscript{,25} applies anomaly detection technique based on principal component analysis (PCA) aimed at incipient faults such as software aging. Using case studies involving long-running enterprise benchmark applications, Trade6 and RuBBoS, with injected memory leaks, performance of the PCA-based detector when using just the compressed data is almost equivalent to the case in which the raw data is completely available, but with fewer samples with a compression rate exceeding 75%.

Cotroneo et al.\textsuperscript{,26} define a stress test methodology applied to the HotSpot JVM, based on Design of Experiment (DoE) for a workload-based analysis, PCA for removing first-order correlation among aging indicators at JVM level, clustering to identify workload states, and then multiple regression to relate JVM aging indicators (transformed in the PC space) to the OS memory depletion and user-perceived throughput loss.

Mohan et al.\textsuperscript{,27} study the effect of aging on power usage by using linear regression to estimate the trend. Energy consumption is also considered in\textsuperscript{,28} where an IDS-based self-protection mechanism at the virtual machine level inspired by software rejuvenation concepts is presented. A correlation between IDS accuracy, attack rate, cloud system workload, energy consumption, and response time is identified.

\textit{Machine Learning.} Machine learning adopts algorithms from the field of artificial intelligence (e.g., classifiers and regressors) to identify trends and classify a system state as robust or failure-prone.

One of the first works in this area is by Alonso et al.\textsuperscript{,29} They analyze a three-tiered J2EE system; a machine learning approach is used to automatically build regression trees models that relate several system variables (such as number of connections and throughput) to aging trends, based on the observation that such trends can be approximated using a piecewise linear model. The models are trained using data samples collected in preliminary experiments, and then used to predict the Time To Exhaustion (TTE) of system resources under conditions different than the ones observed during the training phase.

Other techniques use classifiers, like naïve Bayes classifiers, decision trees, and Artificial Neural Networks (ANN). In particular, Sudhakar et al.\textsuperscript{,30} use ANN to capture non-linear relationships between resource usage statistics and the time to failure in cloud systems, to generate a prediction of the time to failure.
Aging in Virtualized Environments

The work by Avresky et al.\textsuperscript{31} and Di Sanzo et al.\textsuperscript{32} present a framework using machine learning models for predicting failures caused by accumulation of anomalies and a proactive system scale up/scale down technique applied at client-server applications in the cloud. It predicts the remaining time to the occurrence of some unexpected event (system failure, service level agreement violation, etc.) of a VM hosting a server instance of the application. Machine learning is also used in the work by Simeonov et al.,\textsuperscript{33} who propose a framework with three VMs, one VM master and two identical slaves, one active and the other in standby; the slaves send health data to the master, which predicts aging based on machine learning algorithm running on such data.

\textit{Threshold-Based Approaches}. These approaches define thresholds for some aging indicators, so as to trigger rejuvenation when the monitored indicators exceed such thresholds. An example of this approach is in the work by Silva et al.,\textsuperscript{34} which adopts thresholds on mean response time and on quality of service indicators. The authors propose a rejuvenation approach based on self-healing techniques, that exploits virtualization to optimize recovery. They propose a rejuvenation framework called \textit{VM-Rejuv}, exploiting virtualization to optimize recovery, with an aging detector module detecting aging conditions based on the mentioned thresholds.

The advantage of measurement-based approach is that software aging forecasting can adapt to the current condition of the system (e.g., the current operational profile, which may not have been foreseen before operation), and can accurately predict the occurrence of aging phenomena. However, this kind of approach may not be easily generalizable, since they exploit some peculiar aspect related to the nature of the considered system (e.g., the fact that some particular resource exhibits seasonal, fractal patterns, or exploit the regularity of the phenomenon). Moreover, measurement-based approaches are not meant to estimate long-term dependability measures such as availability.

- Differently from model-based techniques, the analysis and validation of these techniques consider measurements from real systems, with the goal of identifying whether the system is in a failure-prone state due to software aging, to forecast the time-to-aging-failure, and to plan software rejuvenation accordingly. The advantage of a measurement-based solution is the possibility to gather accurate and detailed information about the aging state of the system, and there is no need to make assumptions on model parameters, since real data are available; on the other hand, findings are very system-specific, and hard to generalize, even though the large base of systems analyzed with these approaches is constantly increasing, thus improving the generalization ability of this type of solutions.
In the context virtualized environments, systems range from cloud applications (more often web/application servers running on the cloud) to entire platforms, the most studied one being Eucalyptus. Specifically, several authors\textsuperscript{23,35–38} analyze aging and propose rejuvenation in the Eucalyptus cloud computing infrastructure by measurement-based approaches, e.g., time series forecasting. These are applied to common indicators about resources consumption, especially: RAM memory and swap space exhaustion and CPU utilization by the VMs. Based on time series analyses, prediction-based rejuvenation is scheduled, so as to reduce the downtime by predicting the proper time to perform the rejuvenation.

A quite different aging study on Eucalyptus is presented by Langner \textit{et al.}\textsuperscript{39} The authors propose a technique to detect aging problems shortly after their introduction by runtime comparisons of different development versions of the same software. In this case – as in few other cases in the SAR literature\textsuperscript{40,41} - the detection aims at revealing aging problems before their long-term runtime manifestation.

- Besides studies on cloud computing, a relevant environment on which measurement-based methods have been applied is represented by the \textbf{Java Virtual Machine (JVM)} and related applications (e.g., based on J2EE). In this case, parameters of interest at VM level are about the state of the JVM (captured by parameters like heap usage, instantiated classes, number of threads, objects size, number of allocations, number of garbage collections, number of JIT compilations, etc.) Several examples of measurement-based approaches in the JVM present techniques for \textbf{leak detection}. Haining \textit{et al.}\textsuperscript{42} and then Meng \textit{et al.}\textsuperscript{43} study the performance of the JVM running a J2EE Application Server, looking for memory leaks associated with the garbage collection mechanism in JVM. They use the Virtual Machine Profiler Interface (JVMPI), to collect resource usage from the application server, including the JVM heap memory usage and CPU utilization, as well as the response time and throughput at client level.

In the studies by Šor \textit{et al.}\textsuperscript{44,45} memory leaks detection in Java applications is performed through a statistical approach based on the “age” of objects, measured as number of garbage collection cycles (or generations) they survive. The idea is to analyze how live instances of the class are distributed over different generations, and evaluate if objects follow the “generational hypothesis”, i.e., most of them become unreachable very soon after creation (in other words, they “die” young). The growth of the number of generations where class instances are present is a symptom that the application allocates some objects without then freeing them (memory leak).

Memory leaks are also analyzed by Xu \textit{et al.}\textsuperscript{46} who present an assertion-based technique for leak detection within the Jikes RVM, a high-performance Java-in-Java virtual machine, and then applied it to real leaks in
large-scale applications like Eclipse and MySQL. Xu et al.\textsuperscript{47} also proposed the Self-organizing Maps (SOM) to capture VMM behaviors from runtime measurement data, and takes a neighborhood area density of a winning neuron as an aging quantification metric. Results of two experiments injecting different resource leaks on the Xen platform show that the algorithm has a high true positive rate and a low false positive rate. Sor et al.\textsuperscript{48} use machine learning for memory leak detection implemented in a commercial tool called Plumbr.

A further study on the JVM is the mentioned work by Cotroneo et al.\textsuperscript{26}, which investigates throughput loss and memory consumption in the Hotspot JVM, via a James’ mail server workload, with the DoE-based experimental methodology described above. Many parameters of the JVM are monitored through JVMMon, a monitoring tool developed for the aging analysis purpose, and aging-suffering components within the JVM are identified through correlation analysis. An interesting finding is about the key component designed to limit the aging trends in Java applications, namely the garbage collector, which itself suffers from aging.\textsuperscript{26}

5. Hybrid analysis techniques

An important generalization of the previous two methods for aging analysis is what are called hybrid approaches, proposed by some researchers as a combination of model-based and measurement-based solutions. Hybrid solutions usually adopt a stochastic model to describe the phenomenon, and determine the model parameters through measurement, that is, via observed data. Solutions of this type, although not much common, are indeed able to put together the advantages of both approaches. A typical example is the one proposed by Trivedi and Vaidyanathan\textsuperscript{49} in which: i) a measurement-based semi-Markovian model for system workload is built; ii) TTE for each considered resource and state (using reward functions) is estimated, and finally iii) a semi-Markov availability model is provided, based on field data rather than on assumptions about system behavior.

In the context of virtualized environments, a hybrid solution is adopted by Liu et al.\textsuperscript{50} which measures the trends of various resources in a cloud-based streaming system with ATM endpoints, including CPU, storage, network, at several layers (the stream software, VM, VMM, network) and use them to parametrize a model to schedule service rejuvenation, implemented by means of VM failover to replicas. The end result is an improvement in terms of served transactions per second.

A further hybrid strategy is proposed by Machida et al.\textsuperscript{51} who present an original countermeasure to software aging different from rejuvenation, called software life extension – a sort of workaround in which, upon software aging detection, additional memory is allocated to the VM executing the aging-affected software. In this case, experimental data on memcached are used to derive a more general model
by a semi-Markov process formalism.

The solution proposed by Kadirvel et al.\textsuperscript{52} is applied to a system manager for a batch-based job submission system on a virtualized platform, aimed at managing and controlling virtualized resources to support remediation approaches (such as elastic increase and decrease of resource capacity, VM migrations, dynamic resource configuration changes). The technique combines a Petri Net-based approach to model a system manager module, suffering from health deterioration due to resource exhaustion, with the usage of feedback control theory to control resource consumption and delay/prevent resource. Three different rejuvenation strategies are implemented and tested – process rejuvenation, VM migration and dynamic increase of resource allocation, chosen based on the planning module.

In all the mentioned classes of studies, an additional analysis factor is workload dependency. Since aging has been shown to be correlated with workload variation,\textsuperscript{53} several studies accounted for its impact; two examples are the following.

Brueno et al.\textsuperscript{54} present a workload-based analysis of VMM aging and rejuvenation under different policies for availability maximization, under under \textit{variable} workload conditions. They exploit dynamic reliability theory and symbolic algebraic techniques, representing the CDFs associated with the VMM events, under different workload conditions, by continuous phase type (CPH) distributions, and using the Kronecker algebra to implement the conservation of reliability principle and the variable timer policy.

Ficco et al.\textsuperscript{55} perform a workload-dependent analysis of performance degradation and memory indicators in Apache Storm, an event stream processing (ESP) application that can deploy tasks over a cloud architecture, by means of workload-dependent time series analysis. The Mann-Kendall test and Sen’s procedure, often used for aging trend detection and estimation, are used on sliced windows where the workload and the performance/memory trends are judged to be in contrast (e.g., their trends increase despite the workload decreases).

6. Rejuvenation techniques

Figure 3 outlines the common rejuvenation techniques at VMM and at VM level. Software rejuvenation can act on a virtual machine infrastructure by rejuvenating

![Fig. 3.: Rejuvenation Techniques](image)

the VMM and/or its VMs. About VMM rejuvenation, it can be performed by
restoring the VMM or part thereof, and alternative strategies depend on whether rejuvenation affects only the VMM, or also VMs running on top of the VMM\textsuperscript{8,9}. About the VM rejuvenation, the strategies differ in how the rejuvenation process is managed. They are explained in detail hereafter.

VMM rejuvenation can be performed by the following techniques:

- **Cold-VM rejuvenation**, in which the VMs are also restarted when the VMM is rejuvenated. Cold-VM rejuvenation simply shuts down the hosted VMs before triggering the VMM rejuvenation and restarts the VMs after the completion of VMM rejuvenation.

- **Warm-VM rejuvenation**, in which the execution state of each VM (including the OS and applications running in the VM) are stored to persistent memory, and resumed after the restart of the VMM, in order to reduce the downtime of restarting VMs and their services (although the software running in the VMs is not rejuvenated). This operation can be quickly performed by adopting an on-memory suspend/resume mechanism, in which the memory images of VMs is preserved in main memory during the VMM restart rather than on persistent storage, in order to avoid slow read/write operations to persistent storage. Kourai et al.\textsuperscript{56,57} propose such a variant, and show that compared with Cold-VM rejuvenation, Warm-VM reboot improves the availability of the application hosted on VMs by introducing the on-memory suspend technique and the quick reload mechanism.

- **Migrate-VM rejuvenation**, in which the downtime is further reduced by migrating a VM to another host while the VMM is being rejuvenated, in order to make them available during rejuvenation. This latter scheme does not rejuvenate VMs, and is limited by the capacity of other hosts to accept migrated VMs. Migration can further be divided based on the type of live VM migration (\textit{stop-and-copy} or \textit{pre-copy}), and the policies to return back to the original host after VMM rejuvenation (\textit{return-back} or \textit{stay-on}).\textsuperscript{9} Specifically, the \textit{stop-and-copy} migration foresees to stop the VM operation and copy all the memory contents to the destination server. In the \textit{pre-copy} variant, the VM’s memory is first copied to a destination server without stopping its operation, causing dirty pages in the copied memory contents; then, a stop-and-copy phase is performed, which however updates only the dirty pages, so considerably reducing the downtime overhead compared to a complete stop-and-copy of the entire VM’s memory. As for the restore of the VM on the original host, a \textit{return-back} policy foresees the migration to the original host soon after the VMM rejuvenation is completed; a \textit{stay-on} policy allows the migrated VM to run on the other hosting server even after the completion of VMM rejuvenation on the original host.

The work by Machida et al.\textsuperscript{9} uses SRNs for cold-VM rejuvenation, warm-VM rejuvenation and migration. They studied the steady-state availabili-
ity of a VMs and the expected number of transactions lost, finding that Migrate-VM rejuvenation generally achieves higher steady-state availability compared to Cold- and Warm-VM rejuvenation, because of the ability to preserve the VM execution even during VMM rejuvenation. Moreover, they found that using pre-copy migration is generally better than the pure stop-and-copy migration, and using the return-back policy for migrating back is generally more effective than the stay-on policy.

Machida et al.\textsuperscript{58,59} propose also the contemporary rejuvenation of aged VMs and the VMM in virtualized data centres; differently from usual Cold-VM rejuvenation, the technique forces shutdown only to aged VMs which are to be rejuvenated in the near future, while the robust VMs are moved out from the host server by live VM migration before VMM rejuvenation. Kourai et al.\textsuperscript{60} present VMBeam, a technique that enables lightweight software rejuvenation of virtualized systems using zero-copy migration. The technique starts a new virtualized system at the same host by using nested virtualization, and then migrates all the VMs from the aged virtualized system to the clean one. VMBeam relocates the memory of the VMs on the aged virtualized system directly to the clean system, without any copy. Melo et al.\textsuperscript{19} use an SPN model under two migration-based rejuvenation strategies (with and without a test before migration).

The approach proposed by Torquato et al.\textsuperscript{61} discussed in the previous Section foresees a rejuvenation phase performed by means of VM Live Migration, applied on Cloud Computing testbed which uses OpenNebula and KVM as VMM.

- **VI Micro reboot.** A more complex technique is to perform a very fine-grained reboot of software modules (micro reboot), tailored for the system-level virtualization software (i.e., for the virtualization infrastructure – VI). The technique has been proposed by Le et al.\textsuperscript{62} who applied micro reboot to all modules of the Xen virtualization software; this consists of three main components: the privileged virtual machine, the device driver virtual machine, and the virtual machine monitor.

The best technique (or the best combination of them) depends on the speed of storing/migrating the state of VMs and on the capacity of hosts, as well as on the aging rate of VMs and VMMs, therefore the rejuvenation policy should be determined according to these factors.

In order to rejuvenate VMs themselves, besides the mentioned Cold-VM rejuvenation that indirectly rejuvenates also the VM, a conventional replication approach can be applied:

- **VM Failover.** VM Failover techniques are based on conventional active/passive replication. The idea is to redirect, upon detection of aging in a VM, all the incoming requests to a another VM, and then rejuvenate
the aged VM (e.g., by restart). A VM can be rejuvenated while the other replicas are active and the workload is redirected to them, or by activating a standby (i.e., idle) replica when rejuvenation is triggered.

For example, Silva et al.\textsuperscript{34} propose a cluster failover framework for web applications based on virtualization, namely VM-Rejuv. The framework consists of a Load Balancer, an Active Server, and a Standby Server, each running in a dedicated VM. The Load Balancer redirects requests to the Active Server while it is correctly working, and monitors the Active Server for aging symptoms (e.g., performance falls below a threshold). When rejuvenation is triggered, new requests are redirected to the Standby Server; the Active Server is rejuvenated only after all pending requests have been processed and session data have been migrated to the Standby Server, in order to assure a clean restart (i.e., rejuvenation does not cause the loss of session data and the failure of user requests). This framework can be implemented in a cost-effective way by using off-the-shelf application servers, monitoring, and load balancing software, at the cost of a moderate overhead.

In a virtualized environment, the management of the state transfer can also be done by exploiting the hypervisor. For instance, the work by Distler et al.\textsuperscript{63} addresses the problem of stateful replica management proposing an architecture where recovery is allowed in parallel with service execution, and uses copy-on-write techniques for state transfer between virtual replicas of a host. Similarly, Reiser et al.\textsuperscript{64} initially proposed to use hypervisor to initialize a new replica in parallel to normal system execution, and thus minimizes the time in which a proactive reboot interferes with system operation.

It is finally worth noting that, although the advantages of using VMs can be counterbalanced by a higher memory fragmentation induced by the virtualized environment. Alonso et al.\textsuperscript{65} study experimentally the main software rejuvenation techniques, considering (i) physical node reboot, (ii) virtual machine reboot, (iii) OS reboot, (iv) fast OS reboot, (v) standalone application restart, and (vi) application rejuvenation by a hot standby server, and blamed virtualization for the drastic increase in memory fragmentation, which can cause aging-related failures in long-running systems. This stresses the importance of choosing the right set of techniques for applying rejuvenation in a virtualized environment.

7. Conclusions

SAR techniques in virtualized environments are increasingly emerging in the last years. More than 55% of research works in this area are published in the last 5 years, hence it is a relatively young research area. They have appeared mainly after 2007, after the widespread adoption of this kind of systems. In such studies, researchers often analyze several rejuvenation policies based on virtual machine and/or virtual machine monitor reboot/rejuvenation. Such strategies are then evaluated both
by model-based approaches and by measurements. The availability of cloud computing software to experiment such strategies without excessive costs is favoring aging analysis on cloud. A relevant example is represented by the studies on Eucalyptus cloud-computing framework. From the model-based perspective, models differ from each other in the formalism adopted (e.g., CTMC, SPN, SRN, MRGP, MRP), in the configuration/architectural model (e.g., single vs multiple-host), in the attributes of interest analyzed (survivability, availability, performability), and in the rejuvenation schemes (e.g., Cold-VM, Warm-VM, Migrate-VM, VM failover). Measurements-based approaches differ from traditional SAR approach just in the indicators being adopted: as mentioned, the execution environment concept is quite different from non-virtualized systems, but the techniques used (mostly, time series analysis and machine learning) are unchanged with respect to conventional SAR literature. A relevant difference is that in most of works on SAR in virtualized environment, researchers associate one or more rejuvenation actions to the aging evaluation task, resulting in a literature slice with much more works including rejuvenation strategies with respect to the general SAR literature. Looking at the increasing trends of virtualization and cloud computing, it is easy to envision a proportional increasing trend in the studies dealing with aging and rejuvenation in virtualized environments in the next years. Moreover, because of the increased economical impact of availability/performability/survivability and thanks to the possibilities offered by emerging cloud-based architectures, we will likely witness to an explosion of rejuvenation solutions at several layers of virtualized environments.

References

8. F. Machida, D. S. Kim, and K. S. Trivedi. Modeling and analysis of software rejuve-
Aging in Virtualized Environments


