A Survey Report On Software Aging and Rejuvenation studies in Virtualized Environment

SUBHASHREE BARADA
School of Computer Engineering
KIIT University
Bhubaneswar, India
baradasubhashree@gmail.com

SANTOSH KUMAR SWAIN
School of Computer Engineering
KIIT University
Bhubaneswar, India
sswainfcs@kiit.ac.in

Abstract- Software aging is a phenomenon that occurs in a long running complex software system that tends to decrease in performance or increase in failure rate. This phenomenon lead to performance degradation and crash/hang failure. To counteract software aging a technique named as software rejuvenation has been proposed, which remove aging related failures and its effects. Nowadays, virtualized platform has become the popular choice to deploy complex and long running application. Critical and long running applications are always expected to be available but these applications are more prone to suffer from software aging phenomenon. To counteract this phenomenon software rejuvenation technique has been used. To eliminate downtime outages due to rejuvenation we have combined rejuvenation manager component with two aging indicators application-specific indicator and system-wide indicator. The idea behind our paper is to propose a algorithm which will choose the correct rejuvenation technique for virtual machine(VM) and virtual machine monitor(VMM) according to its aging effect. Through this method we can save both time and cost.

Keywords- Software aging, Software rejuvenation, Virtualized, VM, VMM, Application-specific indicator, System-wide indicator.

I. INTRODUCTION

Increase of failure rate in computer system are more due to software failures than hardware failures. Software often shows degradation in performance level after using it for long time[1]. As we use a software for a longer time period it shows some error like degradation of software, crash or hang failure, memory leaking, memory fragmentation, unreleased file locks, data corruption, storage space fragmentation and round of error. This constitutes a phenomenon called Software aging[1][2]. Software aging means software has aged and it shows sign of aging like crash or hang. Software aging can be compared with human aging. Like human software also exhibits memory leak, decrease in performance level, increase of downtime cost, burden to new generation etc[3]. Due to this reliability and availability of a software have increased tremendously. The consequence of software aging leads to software failures which in turn causes huge economic losses or risk to human life. Software aging has been observed not only in long running system but also in safety and mission critical applications. Aging can be considered at different level like such as OS, application process, application component, middleware, virtual machine, and VM monitor [3][4] .Aging has been experienced in many areas like OS, Real time system, Web servers, Cloud computing systems, SOAP server. To counteract the aging effect a preventive technique has been proposed by Huang et al. called as software rejuvenation[8] .Software rejuvenation is a preventive maintenance action which involves cleaning up the aging effects in a system to prevent from future failure. Software rejuvenation is a cost effective and time effective method. Software rejuvenation is used to delay the failure of a system for some time and make the system to perform its work[8]. Virtualization is the new technology which overcomes the error of physical machines like set up cost, memory leak, memory fragmentation, security, reliability etc[10]. Virtualization is the technology that creates virtually a software abstraction layer between hardware and application and OS running on top of it. Virtualization is the underlying mechanism of cloud computing[10]. As cloud computing is becoming popular technology in today's scenario most of the physical machines are transferred to virtual machines[9]. But continuous operation on VM and VMM lead to software aging issues. As VMM is the abstraction layer between hardware and OS and many of the applications running on top of it is not rebooted frequently. So to mitigate the effects of software aging in VM and VMM we have combined rejuvenation techniques with aging related indicators for better results.

This paper has been organized as follows: In section-II, we have represented review of the literature, in section-III, we have highlighted the comparisons among physical machines and virtual machines of their respective software aging effects, in Section-IV, we have summarizes the discussion. Finally, in section-V, we have given our conclusion and future work.

II. REVIEW OF LITERATURE

Many works have been reported on software aging and rejuvenation. Software aging is not a new topic still it lacks proper research background. Due to which recent safety critical applications also suffer from software aging. But, only few works have been reported on software aging in virtualized environment. In this chapter, we discuss the work done by previous researchers on many more other categories of memory related software aging both in physical machines and virtual machines.

Rivalino Matias et al. [15] on 2012 has done research work on memory related software aging issues which causes aging related failures. They have mainly focused on memory leak problems. They have discussed important drawbacks of using well known system-wide and application-specific aging indicators, as well as propose effectual solutions for both cases. Memory-related aging effects are caused by memory leak and memory fragmentation problems. Memory leak is a software defect that is mainly caused by incorrect use of memory management routines. A memory leak occurs when an application process dynamically allocates memory blocks and, for some reason, does not release them back to the OS during its runtime. Here the authors have tried to find out memory leaks in a system both in user level and kernel level by the use of aging indicators. Aging indicators will detect the error in a system in running condition. System-wide aging indicators provide information related to subsystems components. With the help of system-wide aging indicators free/used physical memory and swap space they have conducted their experiment. But sometimes this indicators give false indication about memory consumption. So they have used aging free baseline to compare all the memory consumption with it for better result. Application-specific aging indicators provide specific information about an individual application process. For detecting memory leaks, proposed the use of the process resident set size (RSS) as aging indicator. Monitoring the RSS in conjunction with the process virtual size is a better strategy than using only the RSS which alone cannot reveal the correct memory leak.

Domenico Cotroneo et al. on 2012[5] have focused on software aging phenomena related to integer overflows. Integer overflows issues are the neglected issues in literature survey. Numerical aging-related bugs can represent an issue for many long-running software applications, such as industrial control systems and signal processing. It is very difficult to avoid and to fix numerical bugs due to lack of importance for computer arithmetic and programming languages by software programmers. The authors have presented some examples of integer overflow that causes software aging. They have tried to give idea about numerical aging related bugs and give research area a new topic for further research. Integer related bugs occur when programmers tries to put the infinite mathematical integers to finite range. They have analysed and discussed about different examples of numerical aging related bugs in MYSQL open source DBMS to give real world problems. Due to lack of interest for integer overflows issues there is no such particular rejuvenation techniques rather than restarting DBMS server. Though software rejuvenation technique can mitigate effect of software aging due to integer overflow still it lack forecasting about software aging. So they have proposed to periodically sample the integer variables to estimate the expected time to overflow for variables at run time and trigger rejuvenation techniques according to estimate value. For future experiment they have include the performance evaluation and optimization of the approach to apply the approach to other kind of systems, and to encompass floating-point errors.

Autran Macêdo et.al on 2011[4] have proposed work on memory related aging effects. The authors have explained how memory management works inside application process, focusing on two memory problems that cause software aging: fragmentation and leakage. Here they have explained the procedure of memory-related software aging focusing on a real and widely adopted memory allocator and presented an experimental study that illustrates how memory fragmentation and leakage occur and how they accumulate over time in order to cause system aging-related failures. For the experiment the test lab is composed of an Intel Pentium Dual Core, 2GHz, 1GB RAM, running Linux (Kernel 2.6.31-19) with glibc 2.10.1.They have created two programs(*Mfrag* and *Mleak*) to implement test cases related to memory fragmentation and memory leak. It is important to highlight that although the total free memory in the heap is larger than the amount requested ,a new memory is requested because the heap is suffering from external memory fragmentation. Requesting a new block to the operating system implies entering in kernel mode, which introduces an extra overhead that penalizes the process performance. Memory leaks inside the operating system kernel affect the entire system and not only a specific process, whose effects remain until the OS restart/reboot. In their future work they have focused on experimenting on memory related effects in virtual environment as it is not susceptible to memory fragmentation .

Lei Cui et al. on 2012[14] have conducted studies in virtualized environment for software aging defect. In this paper aging occurrence is detected by conducting experiments on physical and virtual machines and identify the differences between the two, and propose a feature code-based methodology for failure prediction through system call, then execute a prototype in virtual machine manager layer to predict failure time and rejuvenate transparently. They have conducted four experiments to detect the aging phenomenon in physical machines (PM) and virtual machines, then calculate the aging rate for comparison. For the experiment they have collected three kinds of system resource as metrics that signal software aging Memory resources such as Free Memory, Active Memory; 2) CPU resources including User and System Time; 3) IO resources such as Block Read\Write Count, IO Waiting Time. During stastical analysis they have found decreasing of free memory size. Comparing the aging rate between PM and VM it has been found that aging rate is higher in VM as compared to PM. The authors have proposed a feature code-based method to predict rejuvenation time point and implement a prototype through system calls .

Kehua Su et al. on 2011[6] have proposed a work on software rejuvenation in virtual environment (SRVE) to deal with software aging phenomenon of virtual machine monitor and virtual machine, and to make them improve performance. Software running in the virtual machine system is not rebooted frequently, so software aging issues exist in VMM and VMs. So the authors have proposed some rejuvenation techniques for this. The technique can provide the rejuvenation of VMM and VMs, but it cannot provide zero downtime of the services.

Kenichi Kourai et al. on 2011[8] have proposed a new technique for fast rejuvenation of VMM known as warm-VM reboot. As VMM is the fundamental software for running VMs, its performance degradation affects all the VMs that depend on it. So the author here proposed a new software rejuvenation technique that saves both downtime cost and time. The warm-VM reboot enables efficiently rebooting only a VMM by suspending and resuming VM . They have developed two mechanisms: on-memory suspend/resume of VMs and quick reload of a VMM. When a VMM is simply rejuvenated by system reboot operating systems running on the VMs built on top of a VMM also have to be rebooted when the VMM is rejuvenated. This increases the downtime of services provided by the operating systems. Here they have compared warm-VM rejuvenation with VM-migration. In this paper they have explained the problem of software rejuvenation technique of VMM. They have implemented their experiment based on Xen and performed several experiments. They have compared different rejuvenation techniques like system reboot, cold-VM, warm-VM and VM migration. Compared with a normal reboot, the warm-VM reboot reduced the downtime by 74 percent at maximum. In a cluster environment, the warm-VM reboot achieved higher total throughput than the system using VM migration and a normal reboot.

Fumio Machida et al. [9] on 2008 have presented the issues of performability management in a virtualized data centre(VDC) that hosts multiple services using virtualization. Performability is a concept of a mixed metric of performance and availability. The users of a VDC generally request a certain level of application performance in a service level agreement (SLA). VDC providers need to decide an optimal server configuration and management operations for guaranteeing application performance and maximizing the availability. They have focused on placement algorithm of VMs and rejuvenation schedules for VMs and VMM in a VDC. The VM placement, which assign VMs to applications, should be decided for satisfying the performance requirements under the limited number of physical servers. In the meantime, the rejuvenation schedule should be decided for VMs and VMMs for maximizing overall system availability in a VDC. During the down time of a VM, the number of available application instances decreases and the performance of application service go down. During the down time of a VMM, VMs and applications running on the same physical server must be down as well. The goal of performability management in a VDC is finding an optimum VM placement with optimum rejuvenation schedules for VMs and VMMs. The optimum VM placement and schedules maximize the overall availability and performance of the VDC under the constraints of performance levels of applications specified in SLAs. They formulized the problem of performability management in a VDC.

Kenichi Kourai et al. [10]on 2007 have proposed a new technique for fast rejuvenation for VMM called as warm-VM reboot. When a VMM is rejuvenated operating system running on the VMs built on top of a VMM also have to be rebooted. This increases the downtime of services provided by the operating systems. It takes long time to reboot many operating systems in parallel when the VMM is rebooted. After the operating systems are rebooted with the VMM, their performance is degraded due to cache misses. The file cache used by the operating systems is lost by the reboot. The warm-VM technique enables only a VMM to be rebooted by using the on-memory suspend/resume mechanism and the quick reload mechanism. The on-memory suspend/resume mechanism performs suspend and resume of VMs without accessing the memory images. The quick reload mechanism preserves the memory images during the reboot of a VMM. The warm-VM reboot can reduce the downtime and prevent the performance degradation just after the reboot. They have experimented this technique based on Xen and performed several experiments to show the efficiency. The warm-VM reboot reduced the downtime by 83% at maximum and kept the same throughput after the reboot. For future work they have decided to empirically evaluate the reduction of performance degradation by using the warm-VM reboot in a cluster environment and to enable privileged VMs to be rebooted without the reboot of the VMM and to be suspended.

Fumio Machida et al. [11] have proposed a work where they combined server virtualization technique that rejuvenates both VMs and VMMs. To maximize the resource utilization they have used live VM migration for shifting of VMs when VMMs are rejuvenated. By the help of simulation technique they have showed the high availability of VMs with maximum resource utilization in virtualized data centre.

Aye Myat Myat Paing et al. [13] on 2012 have focused on rejuvenation of VMMs efficiently without affecting VMs. They have combined rejuvenation techniques with Live VM migration technology for better optimization of resource usage. By the use of stochastic Petri nets they have given a model using time based rejuvenation for VMM. To evaluate the model they have given numerical analysis.

Thandar THEIN et al. [12] on 2007 have presented a Markov model for analyzing availability for long running applications which suffer from software aging. In that model they have shown availability, downtime and downtime costs during rejuvenation.

III. COMPARISION TABLE

From the above literature survey three comparison tables are given to show aging effect difference between physical machine and virtual machine. Table I compare different software aging problem that lies in physical machines, Table II compare different software aging problem that has affected virtualized systems, Table III compare different rejuvenation techniques to mitigate the effects in virtualized systems.

TABLE I. COMPARISONS AMONG DIFFERENT TYPE OF PHYSICAL MACHINES

Paper Name	**The Mechanics of memory related software aging**	**Monitoring memory related software aging: An exploratory study**	**Monitoring of Aging Software Systems affected by Integer Overflows**
Author	Autran Maceda, Tais B. Ferreira, Rivalino Matias Jr	Rivalino Matias Jr., Bruno Evangelista Costa, Autran Macedo	Domenico Cotroneo and Roberto Natella
Year	2011(IEEE)	2012(IEEE)	2012(IEEE)
Proposed work	They have explained the effect of memory related aging effects by dissecting a real and widely adopted memory allocator.	They have discuss important drawbacks of using system-wide and application-specific aging indicators, as well as propose effective solutions for both cases.	They have focused on software aging phenomenon related to integer overflow.
Advantages	They found out the problem in both user level and kernel level. It is a real problem with real consequences.	Monitoring combined RSS and virtual sizes offer a much more accurate aging indicator for memory leak detection.	Online monitoring approach will help to detect symptoms of potential overflow.
Disadvantages	User level allocator does not release the new acquired blocks which increases the process size. Impact on process performance.	System-wide aging indicators present a high risk of making false diagnostic of aging. RSS aging indicators is not able to detect all leak.	They have not analyzed about numerical aging related bugs that involve floating point arithmetic.
Future Work	To work on whether virtual memory based computing are affected by memory.	For future work combined approach using user level and kernel level indicators can be used for aging.	Performance evaluation and optimization of the approach, to apply the approach to other kind of systems ,and to encompass floating-point errors.

TABLE II. COMPARISONS AMONG DIFFERENT TYPE OF VIRTUALIZED SYSTEMS

Paper Name	**Software Aging in Virtualized Environments: Detection and Prediction**	**Software Rejuvenation in Virtualization Environment**
Author	Lei Cui, Bo Li, Jianxin Li , James Hardy;Lu Liu	Kehua Su, Hongbo Fu, Jie Li, Dengyi Zhang
Year	2012(IEEE)	2011(IEEE)
Proposed Work	In this paper proposed a feature code-based methodology for failure prediction through system call, then implement a prototype in virtual machine manager layer to predict failure time and rejuvenate transparently**.**	In this paper they have analyzes different software rejuvenation techniques used in VMM and VMs.
Advantages	Feature code based technology helps to predict the failure rate and calculate aging rate.	The rejuvenation techniques described in the paper are able to rejuvenate either VMM or VMs.
Disadvantages	The prediction deviation against reality is less than 10% than predicted.	No rejuvenation technique can clear all the aging effects from VM and VMM.
Future Work	Development of a new technology to predict near about accurate software aging effects.	To study some technique which can provide zero downtime of services and software rejuvenation of VMM and VMs using VM migration between servers.

TABLE III. COMAPRISONS AMONG DIFFERENT SOFTWARE REJUVENATION TECHNIQUES

Paper Name	**Fast Software Rejuvenation of Virtual Machine Monitors**	**Toward Optimal Virtual Machine Placement and Rejuvenation Scheduling in a Virtualized Data Center**	**A Fast Rejuvenation Technique for Server Consolidation with Virtual Machines**
Author	Kenichi Kourai, Shigeru Chiba	Fumio Machida, Dong Seong Kim, Jong Sou Park, Kishor S. Trivedi	Kenichi Kourai, Shigeru Chiba
Year	2011(IEEE)	2008(IEEE)	2007(IEEE)
Proposed Work	In this paper, they have proposed a new technique for fast rejuvenation of VMMs called the warm-VM reboot.	Presents the issues of performability management in a virtualized data center that hosts multiple services using virtualization.	They proposed a new technique for fast rejuvenation of VMMs called the warm-VM reboot.
Advantage	It prevented the performance degradation due to cache misses after the reboot.	The optimum VM placement and schedules maximize the overall availability and performance of the VDC.	The warm-VM reboot reduces the downtime and prevents the performance degradation due to cache misses after the reboot.
Disadvantage	Cannot clear all the aging effects from VMM.	VMM are neglected in this case.	All the aging effects are not cleared from applications and VMM.
Future Work	Enable privileged VMs to be rebooted without the reboot of the VMM and to be suspended.	Development of a hierarchical stochastic model which allow dynamic VM migration across physical servers.	To empirically evaluate the reduction of performance degradation by using the warm-VM reboot in a cluster environment.

IV. DISCUSSION

From literature survey about software aging issues focusing on memory leak and memory fragmentation both in physical machines and virtual machines it has been confirmed that software aging can affect any type of safety and critical applications. Software aging is a inevitable process, it only can be prevented. In some of the papers authors have pointed out the main aging effect memory leak and memory fragmentation that has affected many systems. Many of the authors have mainly focused on OS and database which are most affected. The problems that arises in physical machines can be avoided in virtual machines. Some of the authors have pointed out that aging indicator will help to find out the proper aging effect in a system. Some rejuvenation techniques are proposed to mitigate effect of aging in virtualized environment. But this rejuvenation techniques can not remove the aging effects properly from VM and VMM. So our motivation is to propose some rejuvenation technique which will handle the software aging issues in virtualized environment with proper use of aging indicators.

V. CONCLUSION

Virtualization is the new technology that has been adopted to avoid the failures that we find out in physical machines. Software aging is the phenomenon that detect the error probable state in a system which in turn helps us to save both time and cost that used to built a system. Software rejuvenation technique helps to prevent the failure in a system and maintains the system performance into its starting phase. Aging indicators are the main key which will help to detect the error according to its different level of phase. Hence this paper summarizes software aging effects in different system and what are its effects. To mitigate the software aging effect different software rejuvenation technique are suggested. To increase the reliability, security and performance of physical system virtual system are used. From this paper it can be summarized that for better performance in virtualized environment some better rejuvenation technique can be proposed which will be the future work of our paper.

REFERENCES

[1] Michael Grottke, Rivalino Matias Jr., and Kishor S. Trivedi, " The Fundamentals of Software Aging," IEEE, 2008

[2] Domenico Cotroneo, Roberto Natella, , Roberto Pietrantuono, and Stefano Russo, " A Survey of Software Aging and Rejuvenation Studies," ACM, Vol 5

[3] "Software Aging and Rejuvenation," Wiley & Sons, 2008

[4] Autran Macêdo, Taís B. Ferreira, and Rivalino Matias Jr, "The Mechanics of Memory-Related Software Aging," IEEE ,2011

[5] Domenico Cotroneo, and Roberto Natella, " Monitoring of Aging Software Systems affected by Integer Overflows," IEEE, 2012

[6] Kehua Su, Hongbo Fu, Jie Li, and Dengyi Zhang, "Software Rejuvenation in Virtualization Environment," IEEE, 2011

[7] Jyotiprakash Sahoo, Subasish Mohapatra and, Radha Lath, "Virtualization: A Survey On Concepts, Taxonomy And Associated Security Issues," IEEE, 2010

[8] Kenichi Kourai, and Shigeru Chiba, "Fast Software Rejuvenation of Virtual Machine Monitors," IEEE, Vol 8, No 6, 2011

[9] Fumio Machida, Dong Seong Kim, Jong Sou Park, and Kishor S. Trivedi, " Toward Optimal Virtual Machine Placement and Rejuvenation Scheduling in a Virtualized Data Center," IEEE, 2008

[10] Kenichi Kourai, and Shigeru Chiba, "A Fast Rejuvenation Technique for Server Consolidation with Virtual Machines, " IEEE, 2007

[11] Fumio Machida, Jianwen Xiang, Kumiko Tadano, and Yoshiharu Maeno, "Combined Server Rejuvenation in a Virtualized Data Center "

[12] Thandar THEIN, Sung-Do CHI, and Jong Sou PARK," Availability Analysis and Improvement of Software Rejuvenation Using Virtualization," Economics and Applied Informatics, Years XIII, 2007

[13] Aye Myat Myat Paing, and Ni Lar Thein, " High Availability Solution: Resource Usage Management In Virtualized Software Aging, " IJCSIT, Vol 4 , No 3, June 2012

[14] Lei Cui, Bo Li, Jianxin Li, James Hardy, and Lu Liu, "Software Aging in Virtualized Environments: Detection and Prediction," IEEE, 2012

[15] Rivalino Matias Jr., Bruno Evangelista Costa, and Autran Macedo, "Monitoring Memory-Related Software Aging: An Exploratory Study" ,IEEE, 2012

[16] Yennun Huang, and Chandra Kintala, "Software Rejuvenation: Analysis, Module and Applications," IEEE, 1995