ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 2, Issue 11, November 2013

A Survey on Software Aging and Rejuvenation in Server Virtualized System

Ms.N.M.Hemadevi, Ms. M.Dhivya, Mr.K.Manikandan

Abstract— **Cloud computing an elastic form of grid computing offers better resource availability and satisfies the request of the user in low priced cost. Failure tends to occur in the cloud system due to some type of aging related bugs, as a result of which the resources are exhausted and providing service to the user is under a crisis. Software rejuvenation is a technique that aims at avoids the aging related problems through "Refreshing" and "Rebooting" the virtual machines which is on top of cloud systems. As a result software rejuvenation has its profound implication in improving the system reliability. The paper provides a brief survey on aging related bugs on Virtual machines and various ways in which the aging related problems can be avoided in the virtual machine systems.**

Index Terms—Cloud computing, aging related bugs, cold, warm, migrate rejuvenation.

I. Introduction

Cloud computing a form of ubiquitous computing deals with providing everything as a service .Cloud computing mainly used in business and IT industry it offers heavy outsourcing model computational resource, where service availability, security and quality are essential features. In cloud computing High service availability is the most important requirement increasingly being demanded in commercial computer, and communication systems. In recent years many research efforts have been going to find the optimal infrastructure size and configuration that guarantee the desired availability level. Software fault tolerance is often found to be the bottleneck. A failure in software's is mainly due to certain elusive error conditions which it leads to resource exhaustion. Software systems appear to age as error conditions arise and accumulate with operational time due to certain elusive faults in system software and application software. Software rejuvenation is a proactive fault management technique aimed at cleaning up the internal states in order to prevent the occurrence of severe crash failures in the upcoming years the simplest way to emulate software rejuvenation is to reboot the system or restart the aging application. It is a cost effective technique dealing with software faults that includes protection not only against hard failures and also due to degradation over time of application performance. This paper discusses the software aging related failures among server virtualized system, and the ways in which it can be avoided by making use of a procedure called software rejuvenation.

The section 1deals with "the classification of faults" Section 2 is about "the basic concepts of software aging and rejuvenation", section 3 describes "software aging and rejuvenation in server virtualized systems" and conclusion which concludes the paper.

II. Related work

2.1. Classification of software faults:

Faults, in both hardware and software, can be classified according to their phase of creation or occurrence, system boundaries (internal or external), domain (hardware or software). In this section, we limit ourselves to the classification of software faults based on their phase of creation .some studies have suggested that since software is not a physical entity, it is not focusing to transient physical phenomena (as opposed to hardware), hence software faults are stable in nature [1].some other studies organizes software faults as both permanent and transient.

Gray [2] categorizes software faults into Bohrbugs and Heisenbugs. Bohrbugs is essentially stable design faults and hence, approximately it is deterministic in nature. They can be recognized easily and weeded out during the testing and debugging phase (or early deployment phase) of the software life cycle. A software system with Bohrbugs is related to a faulty deterministic finite state machine. Heisenbugs, on the other hand, fit into the class of temporary internal faults and are intermittent.

They are essentially stable faults whose conditions of creation occur rarely or are not easily recreated. Hence these types of faults result in transient failures i.e., failures which may not occur again if the software is restarted. Heisenbugs are extremely difficult to identify through testing. Hence a piece of software which is developed in the operational phase gets released after its development and testing phase, is more likely to be experienced with failures caused by Heisenbugs than due to Bohrbugs.

***Manuscript received Oct, 2013**.*

***Mr.K.Manikandan**, computer science and Engineering, SriGuru Institute of Technology, Coimbatore, India, 9940780689,*

***Ms.N.M.Hemadevi,** computer science and Engineering, SriGuru Institute of Technology, Coimbatore, India, 7502354899.*

***Ms.M.Dhivya**, computer science and Engineering, SriGuru Institute of Technology, Coimbatore, India, 7708833131.*

Most modern studies on failure data have reported that a large percentage of software failures are transient in nature caused by phenomena such as overloads or timing and exception errors. The revise of failure data from Tandem's fault tolerant computer system indicate that 70% of the failures were transient failures, caused by faults similar to race conditions and timing problems.

We designate faults attributed to software aging as aging related faults. Aging related faults fall under Bohrbugs or Heisenbugs depending on whether the failure is deterministic (repeatable) or transient [3]. Foraging-related bugs, environment diversity can be particularly effective if utilized proactively in the form of software rejuvenation. Rejuvenation operation can be triggered either by time based (on deterministic intervals) or by using measurement and analysis of data of the system condition that undergoes software aging problems in various workstation environments.

2.2. Basic concepts of software aging and software rejuvenation:

Software aging is defined as the state of the software that degrades with time. The primary causes of this degradation are the exhaustion of operating system resources, data corruption, and accumulation of numerical errors, which eventually may lead to performance degradation of the software, crash/hang failure, or both.

A typical example of software aging is progressive increase in memory consumption which conclusively causes a memory leak. Since software aging can be observed only in the software execution, it is difficult to find aging related problems until the software is deployed and executed in a specific environment. This figure describes the threads which lead to aging related failure in the system.

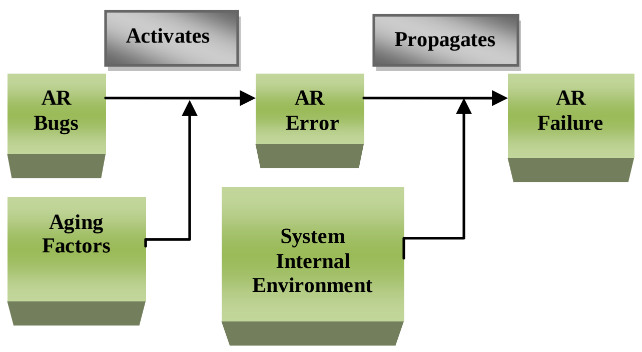

Fig.1 "chain of threats" for an (**AR**) aging-related failure

The accumulation of AR (aging related) errors may tend to AR failure or fault. Aging effects can also be classified into **volatile** and **non-volatile** effects. They are considered volatile if they are isolated by re-initialization of the system or process affected, for example via a system reboot. In contrast, non-volatile aging effects still exist after reinitializing of the system/process. Physical memory division and OS resource outflow are examples for volatile aging effects. File system schema and database metadata fragmentation are examples for non-volatile aging effects [4]. The fault tolerance technique which is used to mitigate the aging effects of system is known as software rejuvenation.

Software rejuvenation is defined as occasionally stopping the running software, cleaning its internal state or its environment and restarting it. Such a technique known as "software rejuvenation" was proposed by "Huang et al", which counteract the aging phenomenon in a proactive manner by removing the accumulated error conditions and freeing up of operating system resources. Garbage collection, flushing operating system kernel tables, and reinitializing internal data structures are some examples by which the internal state or the environment of the software can be cleaned. There are basically two approaches followed for Software rejuvenation and for finding the optimum rejuvenation schedule: first is by analytic model and measurement based rejuvenation. The analytic modelling approach assumes failure and repair time distributions of a system and obtains optimal rejuvenation Schedule to maximize the availability, or minimize the loss probability or downtime of cost. Measurement-based rejuvenation approach is based on monitoring of resource consumption in a computer system and analysis of that data to determine the point of time when a resource will be completely exhausted, thereby causing the system to hang/crash. Measurement based Software Rejuvenation can follow any of the following policies: Purely Time based Software Rejuvenation Policy (PTSRP) or Purely Prediction based Software. The rejuvenation scheduling operation is described in figure.

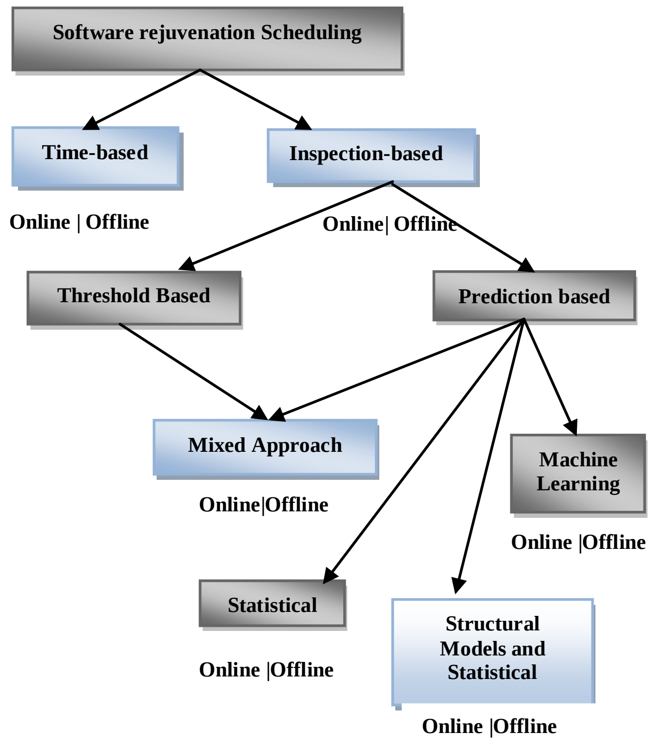

Fig.2 Rejuvenation Scheduling

2.1.1. Approaches of software rejuvenation:

2.1.1.1. Open loop approach:

In this approach the rejuvenation is performed based on system information on elapsed time or instantaneous/ cumulative number of jobs on the system.

2.1.1.2. Closed-loop approach:

In the closed loop approach, rejuvenation is performed based on information on the system "physical condition". The system is monitored constantly at small deterministic intervals and information is collected on the operating system resource usage and system activity. This information is then analyzed to estimate the resource exhaustion time, which may lead to a portion or an entire system degradation/crash. This estimation can be based purely on time or can be based on both time and system workload. Another scheme to estimate the optimal time for rejuvenation could be based on system failure data. The closed-loop approach can also be classified based on data analysis processes is done off-line or on-line. Off-line data analysis is done based on system information collected over a period of time (usually weeks or months). The analysis is done to estimate time for rejuvenation. This off-line analysis approach is best suited for systems whose behavior is fairly deterministic. The on-line closed-loop approach, on the other hand, performs on-line analysis of system information collected at deterministic intervals. The analysis is done after every new set of data is collected to estimate time to rejuvenate. This approach is very general and can work with systems with unpredictable behavior or whose behavior cannot be easily determined.

2.3. Software aging and rejuvenation in server virtualized system:

Software aging problem has been studied in the UNIX operating system and Apache web server. Newly, software aging in the virtual machine monitor (VMMs) as critical as server consolidation using virtual machines (VMs) is widely being carried out. A hypervisor or vmm is the piece of computer software that multiplexes physical resources such as CPU and memory to the VMs running on top of it.

Since several VMs run on one machine consolidating multiple servers, aging problem of the VMM directly affects all the VMs.

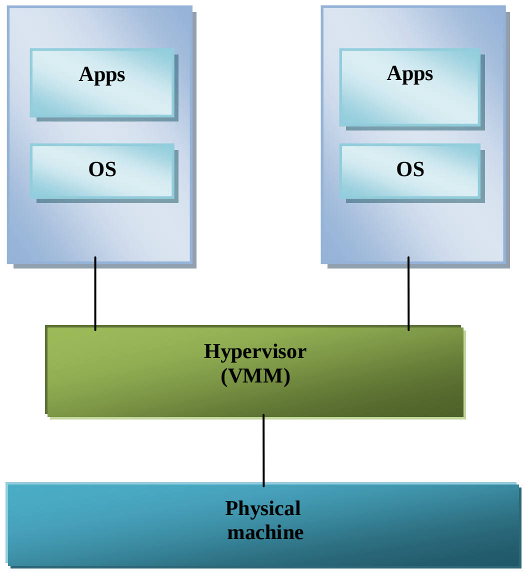

Fig.3 Vmm (virtual machine monitor)

To avoid such software aging, a proactive technique called *software rejuvenation* has been proposed Compared to the traditional software rejuvenation used for common software components, VMM rejuvenation may need additional consideration to the downtime of hosted VMs running on a VMM because they might become unavailable during the VMM rejuvenation. Such downtime needs to be taken into account in determining an appropriate rejuvenation technique while triggers VMM rejuvenation.

2.3.1. Rejuvenation technique:

In this section, we review the three VMM rejuvenation techniques. When VMM rejuvenation needs to be performed on a host, the hosted VMs also need to be controlled because the execution environments of VMs are cleared by the VMM rejuvenation. Prior to VMM rejuvenation, we can perform VM shutdown (i.e., Cold-VM rejuvenation), VMsuspend (i.e., Warm-VMrejuvenation), or VMmigration (i.e., Migrate-VM rejuvenation). These approaches are presented in the next three subsections.

2.3.1.1. Cold-VM rejuvenation:

The easiest way to deal with the hosted VMs before triggering rejuvenation of VMM is to shut down all the hosted VMs regardless of the execution states of the VMs. The VMs are then restarted in clean states after the VMM rejuvenation. This approach is called Cold-VM rejuvenation. All the transactions running on VMs are vanished by the Cold-VM rejuvenation [6]. An advantage of the Cold-VM rejuvenation, however, is that the rejuvenation action cleans all the aging states of the VMs in addition to the aging states of the VMM.

2.3.1.2. Warm-VM rejuvenation:

Instead of shutting down the hosted VMs, the hosted VMs are suspended prior to VMM rejuvenation is triggered and the executions of the VMs are resumed at the completion of the VMM rejuvenation. We call this technique Warm-VM rejuvenation [5]. Since the execution states of the hosted VMs are saved prior to VMM rejuvenation, the transactions running on the VMs are not lost due to the VMM rejuvenation. However, Warm-VM rejuvenation retains the aging states of VMs by VM suspend. The aging states in the hosted VMs are not cleared by VMM rejuvenation and hence we need to rely on rejuvenation for VM to clear the aging states of VMs.

2.3.1.3. Migrate-VM rejuvenation:

Live VM migration is a technique to move a running VM to another host incur a short service interruption and is supported in most modern VMM implementations such as Xen and VMware. Although a shared storage system is required to store a VM image, the downtime overhead caused by a VM migration is less. Using live VM migration, hosted VMs are moved to another host prior to VMM rejuvenation and returned back to the original hosting server after the completion of the rejuvenation of the VMM, by a reverse live VM migration. We call this combined method as Migrate-VM rejuvenation [6]. The VM continues the execution even while the VMM on the original host is being rejuvenated. However, the aging states in the hosted VMs are not cleared by the VMM rejuvenation as in the case of Warm-VM rejuvenation. Live VM migration works only when the migration target server is running and it has a capacity to accept the migrated VM.

We further consider the following aspects of Migrate-VM rejuvenation. There are two types of live VM migration "stop and copy" and "pre copy". The Vm makes use of migration back policies namely "return back" and "stay-on" under the return back policy the migrated Vm is moved back to the original host server and in the stay on policy migrated Vm are allowed to run in the other hosting server even after Vmm rejuvenation of original host server.

III. EXPERMENTAL WORK

The three rejuvenation techniques are compared by numerical solution of SRNs (Stochastic Reward Nets) using SPNP (Stochastic Petri Net Package) [7]. The various parameters, its value and their mean time are compared based on the previous research works in order to find the steady state availability of Virtual Machine Monitoring (VMM) rejuvenation technique. The parameters are illustrated in **Table 1** as shown below.

Table 1. Shows the parameters values

Parameter		Values	
	Description	Value	Mean time
λ_{FPV}	VM aging rate	0.005952381	1 week
λ_v	VM Failure rate after aging	0.01388889	3 days
μ_v	VM Failure detection rate	12	5 mins
β_v	VM Failure recovery rate	2	30mins
Υ_v	VM rejuvenation rate	60	1 min
Υ_v	VM restart rate	120	30 secs
σ_v	VM shutdown	120	30 secs
γ_v	VM suspend rate	45000	0.08sec
δ_v	VM resume rate	4500	0.8sec
ω_v	VM migration rate	3600	1 sec
τ_v	VM rejuvenation trigger rate	0.041666	1 day
η_v	VMM aging rate	0.001378	1 week
λ_{fph}	VMM failure detection rate	11	5 mins
δ_h	VMM failure recovery rate	1	1 hour
β_h	VMM rejuvenation rate	30	2 mins
τ_h	VMM rejuvenation trigger rate	0.00595238	1 week

The above parameter which is discussed is based upon on previous work [5].

Optimal rejuvenation schedule

From this parameter values, first we computing the Steady-state availability by varying the rejuvenation trigger intervals of VM and VMM. We define the reward function for computing steady-state availability. By using certain combination of the rejuvenation trigger intervals, the steady-state availability of the VM is maximized. This is caused by the fact that more repeated rejuvenation increases the down time due to the rejuvenation and less frequent rejuvenation also increases the down time caused by software failure. By identifying the point which maximizes the steady-state availability, we can solve the optimum combination of rejuvenation trigger intervals. The maximum values of steady-state availability for the three rejuvenation techniques are shown in **Table 2**. To find the optimum combination, we used a gradient search method [8]. As a result, Migrate-VM rejuvenation outperforms the other two rejuvenation techniques in terms of the maximum value of steady-state availability of the VM.

Table 2.optimal rejuvenation schedule

METHODS	$1/\tau_v$[hour]	$1/\tau_h$[hour]	STEADY-STATE AVAILABILITY
Cold-VM rejuvenation	34.72	121.95	0.998080
Warm-VM rejuvenation	31.95	112.36	0.998115
Migrate-VM rejuvenation	31.90	10.90	0.998714

IV. CONCLUSION

The rejuvenation in a server virtualized system using the warm, cold and a migration method in the Migrate-VM rejuvenation achieves higher steady-state availability compared to the other two due to the ability to preserve the VM execution, during VMM rejuvenation. Depending on the VMM rejuvenation techniques chosen, determining an appropriate combination of rejuvenation triggers the intervals of VM and VMM that is essential for achieving high-availability.

ACKNOWLEDGEMENT

We would like to thank the principal and staff members of SriGuru Institute of technology, friends and family members for their support and guidance in bringing this article. We also like to thank them for their valuable reviews.

REFERENCES

[1] Y. Huang, P. Jalote, and C. Kintala, "Two Techniques for Transient Software Error Recovery," Lecture Notes in Computer Science, vol. 774, pp. 159-170, and 1994.

[2] J. Gray, "Why Do Computers Stop and What Can Be Done About It?" Proc. Fifth Symp. Reliability in Distributed Software and Database Systems, pp. 3-12, Jan. 1986.

[3] K. Vaidyanathan *and K.* S. Trivedi, *"*A comprehensive model for software rejuvenation", IEEE Trans. on Dependable and Secure Computing, *Apr.* 2005 *(in press).*

[4] Michael Grottke, Rivalino Matias Jr., and Kishor S. Trivedi,"Fundamentals of software aging", In *Proc.* 1st International Workshop on Software Aging and Rejuvenation/ © IEEE 19th International Symposium on Software Reliability Engineering, 2008.

[5] K. Kourai, S. Chiba,"A fast rejuvenation technique for server consolidation with virtual machines", in: Proc. Int'l Conf. on Dependable Systems and Networks, DSN 2007, pp. 245–255, 2007.

[6] Dong Seong Kim ; "Modeling and Analysis of Software Rejuvenation in a Server Virtualized System" Software Aging and Rejuvenation (WoSAR), 2010 IEEE Second International Workshop on Date 2-2Nov.2010.

[7] G. Ciardo, J. K. Muppala, and K. S. Trivedi, SPNP: Stochastic Petri Net Package, In Proc. Int'l Workshop on Petri Nets and Performance Models, pp. 142-151, 1989.

[8]M.S.Bazaraa, H.D.sherali, C.M.Shetty,"Nonlinear Programming Theory and Algorithm" John Wiley, New York,1990.

N.M. Hemadevi was born in Erode on 26th February 1991. She received her B.E. degree from Avinashilingam University, Coimbatore, TamilNadu in 2012. She is currently pursuing her M.E. (CSE) in SriGuru Institute of Technology, Varathayangarpalayam, Coimbatore, and Tamil Nadu. Her current research interests in cloud computing and distributed system.

M. Dhivya was born in Theni on 17th November 1990. She received her B.E. degree from Avinashilingam University, Coimbatore, TamilNadu in 2012. She is currently pursuing her M.E. (CSE) in SriGuru Institute of Technology, Varathayangarpalayam, Coimbatore, and Tamil Nadu. Her current research interests in cloud computing, soft computing and distributed system.

Mr.K.Manikandan was born in Dindigul on 27th may 1984.He received his B.E.degree from RVS college of Engineering and Technology, Dindigul, TamilNadu in 2008 and he completed his M.E.in Oxford Engineering College, Chennai in 2012.He is currently working as an Assisant Professor in SriGuru Institute of Technology. Coimbatore. His current research works are in cloud computing and distributed computing

.