International Journal of Combined Research & Development (IJCRD)
eISSN:2321-225X;pISSN:2321-2241 Volume: 4; Issue: 1; January -2015

SIP Policy: Theoretical and Conceptual Facts of Software Rejuvenation in Convoluted System

[1]Nagaraj G Cholli, [2]Dr. Srinivasan G N
[1,2]Dept. of Information Science & Engineering, R V College of Engineering.,
Bangalore, Karnataka, India

ABSTRACT— A software aging in convoluted system refers to the situation where software degrades with span of time. This phenomenon, which may eventually lead to system performance degradation or crash/hang failure, is the result of depletion of operating system resources, data deception and numerical error assembly. A technique called software rejuvenation has been incorporated, which essentially involves periodic aborting an application or a system, flushing its intramural state and re-starting it. A main issue in rejuvenation is to discover ideal time to initiate software rejuvenation. Software rejuvenation is a proactive technique that allows preventing the occurrence of software failing. A novel approach called Smart interval and payload (SIP) policy is introduced to overcome all the hurdles in the present scenario based on Software Rejuvenation approaches. SIP policy accepts time from user and optimizes the rejuvenation time whenever workload is variable; otherwise the system is rejuvenated at its rejuvenation point. SIP policy avoids software failure and it helps to achieve high availability of convoluted system.

Keywords— SOFTWARE REJUVENATION, SYSTEM PERFORMANCE, CONVOLUTED SYSTEMS, WORK LOAD BALANCING.

I. INTRODUCTION

Software rejuvenation has become a new horizon for increasing the system reliability and availability in a long run. With time, the system outages tend to increase due to the aging of software which may be caused due to numerous factors like memory leaks, unreleased locks, file descriptor leaking and so on. The rejuvenation of the software based on time factor tends to periodically rollback a uninterrupted running routine to prevent blunder in the future. Thus the better way to avoid software failure and to increase the availability and reliability of the system is to find the failure probable state and rejuvenate the software prior to the failed state. The phenomenon is to investigate about time based rejuvenation policies in maintaining high reliability of software systems. Software rejuvenation is a process of elegantly ceasing a running routine and re-starting it[1].The rejuvenation strategy is primarily intended for servers where the applications are intended to run incessantly for days without any failure. Software aging involves the gradual degradation of application performance over time that may lead to untimely cessation of the program. The main objective of the process is to maintain higher system reliability and availability by cleaning internal system states prior to the failure state of the application.

II TACKLING SOFTWARE REJUVENATION

Software rejuvenation can be split predominantly into two proposition as follows *[2]:*

a) Time based approach:
In this approach, rejuvenation is performed without any feedback from the system. Here rejuvenation can be based just one lapsed time or instantaneous cumulative number of jobs on the system.

b) Time and payload approach:
In this approach, rejuvenation is performed based on information on the system fitness. The system is monitored continuously and data collected from the system, such as resource utilization and system performance. This data is analysed and an estimated time of resource exhaust is calculated. Estimation can be based purely on time or on system payload.

III SOFTWARE REJUVENATION TECHNIQUES REVIEW

Software rejuvenation techniques depending on dynamic payload and time. Broadly these techniques are classified as follows:

a) *Standard Rejuvenation:*
In Standard rejuvenation [2] [3], once triggered interval is reached rejuvenation occurs. This technique does not take payload into consideration. It ignores both peak load and off peak load.

b) *Delayed Rejuvenation:*
In delayed rejuvenation [3], schedule for rejuvenation are set at peak load nodes. Once the time reaches the peak period, the rejuvenation process gets activated from the next off peak node.

c) Mixed Rejuvenation:

The mixed rejuvenation [2] [3] policy is the combination of standard rejuvenation and delayed rejuvenation technique. If the rejuvenation is timed early in peak period, rejuvenation of the application is done immediately or else the rejuvenation is delayed till the next off period starts.

VI. ARCHITECTURE OF SOFTWARE REJUVENATION IN CONVOLUTED SYSTEMS

Rejuvenation time is calculated depending on variable workload which is given by system. System periodically checks the workload and update to rejuvenation manager.

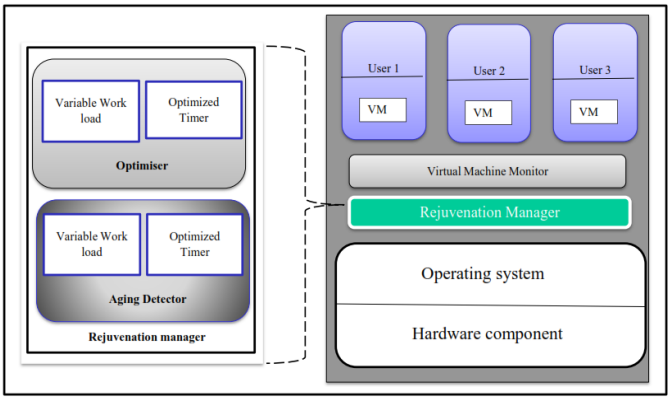

Figure 1: proposed architecture for software rejuvenation in convoluted system involving SIP policy.

SIP policy enabled convoluted system architecture, shown in figure 1, defines a rejuvenation manager consists of aging detector which detects the software aging point and an optimizer to optimize the timer value for a point of rejuvenation. Aging detector and optimizer has two components namely variable workload and timer policy which perform their defined function respectively. Aging detector obtains the value provided by rejuvenation manager periodically and checks for the need for change in rejuvenation time depending on payload and this is updated for rejuvenation manager, if there is need for change in rejuvenation time then rejuvenation manager allows optimizer to change the time, the function of optimizer is to optimize the time depending the values provided by aging detector based on payloads.

V. SIP POLICY FOR REJUVENATING VIRTUAL MACHINE

To focus on the issues related to the aging of the VMM, following assumption is made, neglecting other components failure.

***Assumption*:**

Even though failure not been detected, as per the SIP policy rejuvenation task is triggered, then the rejuvenation Manager will not able to save the VMs states.

In the view of the above assumptions, a finite state machine (FSM) represents the VMM as shown in Fig.2. In the working state, VMM age is increased [4], when a failure is about occur the VMM is not able to accomplish its tasks. Once the failure is detected by the rejuvenation manage VMM enters the failure detected state and soon as the repair is performed, the VMM is rebooted thus restarting from scratch reinitializing VMM age. SIP rejuvenation policy for VMM involve VM reboot, VM suspend and VM migration.

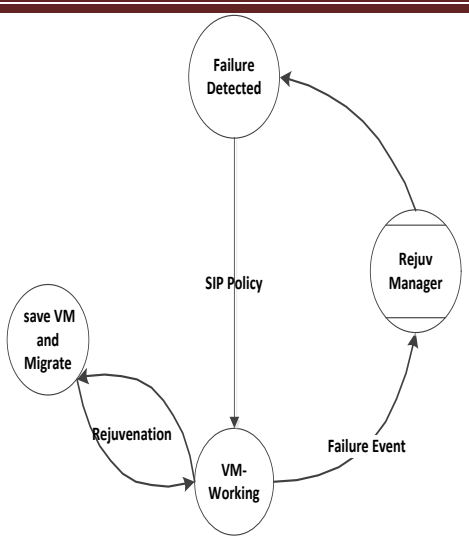

Figure 2: Operating states of Virtual Machine involving SIP policy.

a) **VM shutdown:**
The ideal way to handle 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. Here all the transactions running on VMs are vanished and cleans all the aging states of the VMs in addition to the aging states of the VMM. This approach is also called as Cold-VM rejuvenation.

b) **VM Suspend:**
Here instead of shutting down the hosted VMs, they are suspended prior to rejuvenation is triggered and the executions of the VMs are resumed once the VMM is rejuvenated. Since the execution states are saved prior to rejuvenation, the transactions running on the VMs are not lost due to rejuvenation. Rejuvenating VMM does not control aging in VMs, hence the need of rejuvenating VMs is necessary. This approach is also called as Warm-VM rejuvenation.

c) **VM Migrate:**
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 [5]. 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. The VM continues the execution even while the VMM on the original host is being rejuvenated. However, Rejuvenating VMM does not control aging in VMs 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. This approach is also called as Migrate-VM rejuvenation

VI. SIP POLICY FOR REJUVENATING CONVOLUTED SYSTEMS

SIP policy enabled rejuvenation in Convoluted System has mainly two methods namely Operating System (OS) Cold reboot and OS Warm reboot. Each methods consists of unique working method as follows:

a) **OS Cold Rejuvenation Method:**
In Cold OS reboot [6] process, the system is rebooted immediately at rejuvenation point, i.e., when systems memory consumption reaches a threshold value or predetermined period of time. As per the proposed SIP policy, the free RAM memory left after consumption will be compared to pre-determined threshold value, if the memory left is greater than the threshold value, then the system is allowed to run in normal state OS is restarted immediately.

b) **OS Warm Rejuvenation Method:**

As per the proposed SIP policy, warm OS reboot process, kernel state is saved before rebooting of OS, including all applications running on kernel. Process of saving the kernel state is done by creating a complete image of kernel. OS warm [6] [7] reboot process can be divided in two phases: Suspend and Resume.

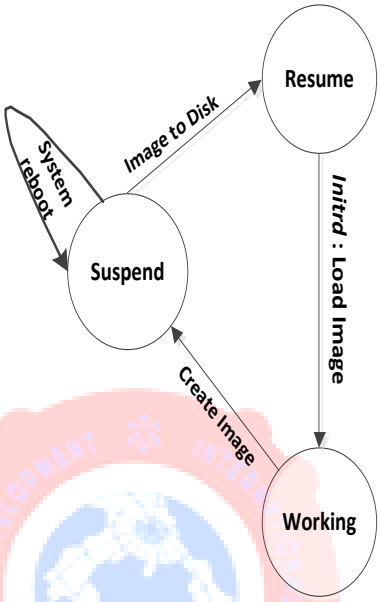

Figure 3: Phases of OS warm reboot process involving SIP policy.

In Suspend phase kernel creates an image of current system state which then writes the data to the disk allowing the system to reboot as shown in Fig.3. In Resume stage, i.e., when the system is turned on, grub loader invokes the ***initrd*** function (before mounting any partitions), later the image is read from disk and loaded to kernel. Thus system runs from same state where it was suspended.

VII. SIP POLICY BASED CONCEPTUAL FACTS OF SOFTWARE REJUVENATION

In this paper, SIP based rejuvenation policy performing dynamic adaptation of the rejuvenation period according to the convoluted system payload condition, is proposed. Fig. 4 shows how the proposed SIP policy works

Considering the aging condition, we propose a modernized interval firing time with respect to the convoluted system payload. The rejuvenation task will be brought into action if the system has been overloaded or deferred if the system has not been intensively used. To take into account the different aging phenomena corresponding to the dynamic payload conditions, the timer is modernized according to a set of payload-dependent period intervals that modulate the way the firing time is adjusted when a payload variation occurs.

Assuming the set of intervals is known, the SIP policy behaves as follows:

i. When the convoluted system is rebooted the timer firing time is set to **Vi** and the clock is set to 0;
ii. Each time the payload varies and reaches a generic payload condition the interval firing time **Vi** is set accordingly.
iii. When the clock reaches the latest interval firing time, the rejuvenation is performed.

The SIP policy behaviour can also be viewed in an equivalent way by reversing the perspective. At a given payload varying instance , **Dp to D_{p+1}**, instead of keeping track of the clock value and then updating the interval firing time, the interval firing time can be fixed according to the payload condition.

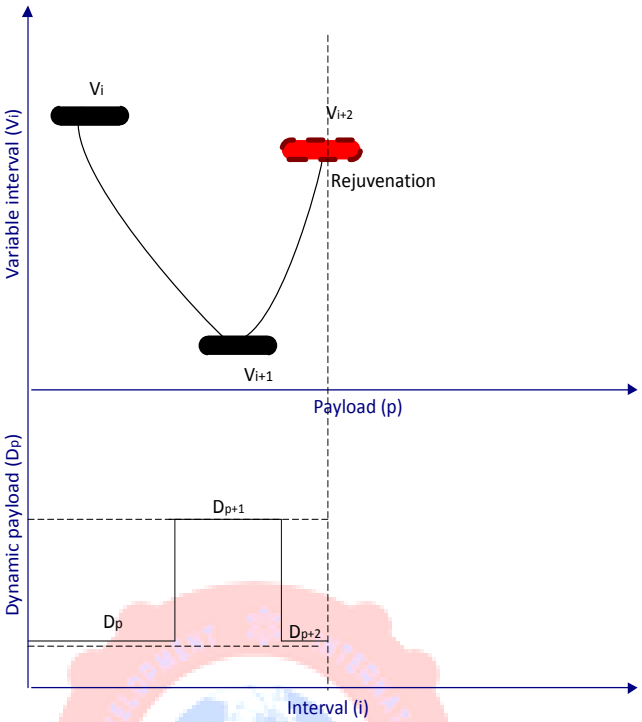

Figure 4: Graphical representation of the SIP policy behaviour

The experimental setup for the proposed architecture as shown in Fig.1, we adopt CentOS operating system because it is open source, highly compatible, stable, and easy to configure KVM (Kernel-based Virtual Machine). For platform virtualization an open source API, libvirt is used, which supports to manage KVM. We make use C programming language, which actually binds libvirt with the centOS. libvirt uses most popular command line interface virsh, which managing virsh guest domains. This program used to create, pause, and shutdown domains. The entire control management of centOS is shown in Fig.5.

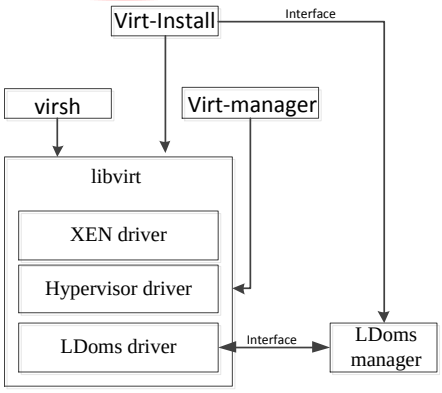

Figure 5: Control management (centOS)

IX CONCLUSION

Smart interval and payload (SIP) policy accepts time from user and optimize the rejuvenation time whenever payload is dynamic, otherwise the system is rejuvenated at its rejuvenation point. SIP policy avoids software failure and it helps to achieve high availability of convoluted system. SIP policy based Architecture of Software Rejuvenation in Convoluted Systems has

been proposed, involving a rejuvenation manager which intern consists of aging detector to check the need for change in rejuvenation time depending on workload and an optimizer to change the time. SIP Policy of rejuvenation can be enforced on modules in both Convoluted Systems and virtual machines. SIP policy considers Physical memory as a primary factor for rejuvenation; hence it is better way to avoid performance degradation and to increase the availability of system.

REFERENCES

[1] Paulo J F, Kishor S Trivedi, Pedro Barbetta.A, "Accelerated degradation test applied to software aging experiments" IEEE Computer, Vol 59,Issue 1, 2010, pp102-114.
[2] Letian Jiang, Guozhi Xu, Xiangyu Peng, "Time and Prediction based software rejuvenation policy", Information Technology and Computer Science, Kiev, 2010, pp114-117.
[3] Stochastic Reward Nets Model for Time based Software Rejuvenation in Virtualized Environment Aye Myat Paing and Ni Lar Thein University of Computer Studies, Yangon Proc. IEEE 26th Pacific International Conference. pp. 125-132, 2011.
[4] Virtual Machine Migration Comparison. "VMware vSphere vs Microsoft HyperV". A principled technologies test report, commissioned by VMware Inc. Principled Technologies Inc.2011.
[5] Pearson and Benameur, "Privacy, Security and Trust Issues Arising from Cloud Computing," Proc. IEEE second International Conference. Cloud Computing Technology and Science, pp. 693-702, 2010.
[6] Javier Alonso Rivalino Matias et al. "A Comparative Evaluation of Software Rejuvenation Strategies", 3rd Int. Workshop on Software Aging and Rejuvenation, Tech report, 2011.
[7] Kishor S.Trivedi et al. "Job Completion Time on a Virtualized Server Subject to Software Aging and Rejuvenation", 3rd International Conference, Workshop on Software Aging and Rejuvenation, Tech report, 2011.
[8] Kishor S. Trivedi et al. "Optimizing Software Rejuvenation Policies under Interval Reliability Criteria", 9th International conference, on Ubiquitous Intelligence and Computing, 9th International Conference on Autonomic and Trusted Computing, 2012.
[9] Lei Cui "Software Aging in Virtualized Environments: Detection and Prediction" IEEE 18th International conference on Parallel and Distributed Systems, 10.1109/ICPADS.2012.111, Singapore, Dec 2012
[10] Arash Rezaei, Mohsen Sharifi, "Rejuvenating High Available Virtualized Systems", International Conference on Availability, Reliability and Security School of Computer Engineering, Iran University of Science and Technology Tehran, Iran, 2010
[11] Yennun Huang and Chandra Kintala, "Software Rejuvenation: Analysis, Module and Applications", IEEE 12th International Symposium on High Assurance Systems Engineering, AT&T Bell Laboratories Murray Hill, NJ 07974 Nick Kolettis and N. Dudley F'ulton, AT&T Bell Laboratories Middletown, NJ 07748, 2010
[12] Domenico Cotroneo, Roberto Natella, Roberto Pietrantuono, "A Survey of Software Aging and Rejuvenation Studies", International paper of Computer Science Università degli Studi di Napoli Federico II, 2011
[13] Ms.N.M.Hemadevi, Ms. M.Dhivya, Mr.K.Manikandan, "A Survey on Software Aging and Rejuvenation in Server Virtualized System", ISSN: 2278 – 7798, International Journal of Science, Engineering and Technology Research (IJSETR) Volume 2, Issue 11, November 2013.