A Study on Software Aging and Rejuvenation Techniques

[1]Nagaraj G Cholli, [2]I M Umesh,[3]Dr. Srinivasan G N
[1,3]*Dept. of Information Science &Engg., R V College of Engg., Bangalore, Karnataka, India*
[2]*BharathiarUniversity, Coimbatore, Tamil Nadu, India*

***Abstract:-*Software aging is the phenomenon in which software systems hang/ crash or show decreased performance. Software rejuvenation is the proactive technique proposed to counter software aging. In this paper, the authors discuss several categories of software "bugs" and how they lead to software aging. Numerical error accumulation, unplanned resource allocation policies are the common causes of software aging. Most of these problems are caused by bad software design or faulty code [1]. Bad code affects the software in the long run; bad code smell is the symptom in the source code of a program that possibly indicates a deeper problem.**

Software rejuvenation is the method which involves stopping the running software and restarting it after removing error conditions. This paper presents an overview of software rejuvenation approaches developed for dealing with software aging. Different approaches need to be used for rejuvenating software systems running on virtualized environment, HPC systems etc.,

Virtualization is the concept where multiple operating systems run on single physical hardware. Virtualization is an emerging technology that has great impact on transforming the IT landscape and fundamentally changing the way that people compute[2]. Several researchers have studied the advantages of virtualization technology to rejuvenate the softwares running on virtual machines and address the software aging problem. In this paper we discuss few such studies and discuss how virtualization has helped software rejuvenation strategies.

Key words - Bad code smell, software maintainability, software metrics, virtualization

I. CAUSES OF SOFTWARE REJUVENATION

Software aging is becoming significant as the economic importance of the software is growing. Software aging is the phenomenon in which software systems show reduced performance or crash. There are various causes for software aging like bad codes, memory leaks, memory fragmentation, data corruption etc., Here, we discuss some of them.

A. Bad Codes

Bad codes affect the software in the long run but they are not technically incorrect; bad code smell is the symptom of the problem that may lead to serious problem later. Bad code smells are a new measure of software maintainability. If left unattended, bad smells can consume lots of resources in terms of maintenance costs, testing etc., bad code smell can be identified by set of software metrics. Software metrics reflects code that is decaying and is likely to cause the frequent faults, and associated testing expenditure. Many researchers have done lot of work on detection of bad code smells.

Tiago Pessoa et al.[3] developed a tool which is an Eclipse plug-in that detects code smells and performs assessment in Java source code. The detection algorithm is based on Binary Logistic Regression, was initially calibrated by using a moderately large set of pre classified methods (by human experts) and validated for the Long Method code smell.

Foutse Khomh et el. [4] detected several code smells in Eclipse and Azureus releases. It was observed that classes with code smells are more change prone when compared to others and also specific code smells are more correlated. The negative impact of code smells on class change proneness was proved by empirical evidence.

Dag I.K. Sjøberg et al. [5] conducted study of code smells effect on maintenance. A controlled study was done to quantify the relation between bad code smells and effort needed for maintenance in an industrial setup. The outcome of the study was that the presence of code smells alone did not affect comprehension but that their combination tended to increase the developers' effort on comprehension tasks.

B. Memory Leaks / Fragmentation

In correct use of memory management routines leads to Memory leak [6]. Memory leak occurs when an application process are dynamically allocated memory blocks and does not release them during execution. Roughly 50% of the applications do contain one or the other memory leaks [7]. Autran Macêdo et al. [8] conducted a detailed study on how memory management is done inside application process focusing fragmentation and leakage memory problems. The authors illustrated how fragmentation and leakage occurs and how they accumulate leading to aging-related failures.

G. Carrozza et al [9] observed that memory leaks are the one of the reasons for memory exhaustion problems in software systems. A practical approach was proposed by authors to identify aging caused by memory leaks in the middleware used for critical applications development. Aging trends triggered due to memory leaks in software systems based on a CORBA-compliant OTS middleware were detected by the authors. A real-world middleware for developing mission-critical applications for Air Traffic Control has been studied.

Distributed applications suffer reduced performance due to memory leaks. As more and more applications are migrated to virtual environment and cloud, it brings additional uncertainty of not knowing the configuration of the physical machine on which the application is actually running [10]. It is very difficult to detect and debug such problems in development and test environments.

Another important memory-related aging effect is the memory fragmentation [1]. As memory fragmentation and memory leaks are located inside the application process, the effects of such defects are less at the user level. The impact will be higher when it happens at the kernel level.

Vaidyanathan, K et al., [11] proposed a model to estimate the exhaustion rate of resources of the operating system both as a function of time and the system workload state. The authors constructed a semi-Markov reward model based on workload and resource usage data collected from UNIX operating system. The fault management techniques such as software rejuvenation may be implemented using this model to prevent unexpected outages.

Michael Grottke et al. [12] created aging effect classes studying different types of aging effects based on their common characteristics as shown in Table 1. Examples for Volatile aging effects are OS resource leakage and memory fragmentation as they can be removed from re-initialization of the system. Non-volatile aging effects still exist after reinitializing of the system/process. Some of the examples for non-volatile aging effects include accumulation of numerical errors; file system and meta data fragmentation.

Table 1. Classes of aging effects

Basic Class	Extension	Examples
Resource Leakage	(1) OS-specific (2) App-specific	Unreleased • Memory (1,2) • File handlers (1) • Sockets (1) Unterminated • Processes (1) • Threads (1,2)
Fragmentation	(1) OS-specific (2) App-specific	Phys. Memory (1) File system (1) Database files (2)
Numerical error accrual	(1) OS-specific (2) App-specific	Round-off (1,2)
Data corruption accrual	(1) OS-Specific (2) App-Specific	File system (1) Database files (2)

Source: Research paper titled "The Fundamentals of Software Aging " authored by Michael Grottke et al.

II. SOFTWARE REJUVENATION

Software Rejuvenation is a proactive fault management technique aimed at cleaning up the system's internal state to prevent occurrence of more severe crash failure in the future [13]. Software rejuvenation mechanism involves the series of steps such as frequently stopping the application and restarting it after cleaning the internal state. Few software rejuvenation techniques have been discussed.

A. Proactive fault recovery

A proactive approach to fault management involves occasionally terminating an application or a system, cleaning its internal state and restarting it. Kalyanaraman Vaidyanathan [14] extended the traditional classification of software faults (deterministic and transient) and included faults attributed to software aging. For each of the fault classes, treatment and recovery strategy was studied and explored methods for evaluating the proactive fault management in operational softwares. The authors consider two-pronged strategy—measurement-based modeling and analytic modeling.

Software aging is diagnosed by collecting and analyzing the collected system data in measurement based approach. Later, proactive methods can be applied to prevent unplanned outages. Potentials problems handling can be automated using measurement based approach where as analytic modeling is aimed to determine optimal times to perform rejuvenation by developing and analyzing stochastic models to maximize availability or minimize downtime cost.

B. Rejuvenation Techniques on Virtualized Environment

Dan Pelleg et al. [15] developed an approach to monitor virtual machines for problems. The approach used hypervisor directly for monitoring of the resource requests. The readings are then analyzed using machine learning. Out-of-band monitoring which uses machine learning and virtualization can be used to accurately identify problems in guest OS.

The disadvantages of existing in-band and out-of band approaches were explained by authors. In-band agents, run on the same agents, In-band agents run on the same system they monitor. This adds workload to the system they are monitoring. An agent may face resource shortage as operating system may divest resources like CPU time or memory.

Out-of-band agents run on different machine other than the system being monitored and hence have reduced visibility into system behavior. The ability to respond is limited to the extent of just informing the administrator about the degrading resource.

The authors successfully identified the hard-to-detect kernel hangs that lead to saturation of CPU resources using their new approach. They also demonstrated that out-of-band monitoring using virtualization and statistical analysis can equal and even surpass the diagnostic accuracy of in-band monitoring, while avoiding the many pitfalls associated with in-band monitoring.

Luis Moura SilvaIn et al [16] proposed automated self-healing techniques which can be applied easily to off-the-

shelf application servers and internet sites. Virtualization can be helpful for software rejuvenation and fail-over in the occurrence of transient application failures and software aging.

The authors used XEN virtualization middleware to create 3 virtual-machines per application server: one VM was used to run software load balancer, the other one to run main application server and the third one was the replication of the application server. The third VM worked as hot stand by.

To detect software aging and trigger a rejuvenation action, they used third VM having load balancer and also a software module. The procedure involved first starting the standby server, then shifting all the new requests and sessions to this second server, the session-state is migrated from the primary to the secondary server. Once on-going requests are finished in the primary server, the server is restarted without losing any in-flight request or session state. This is called "clean" restart.

The authors opined that by using virtualization layer and some software modules, a zero downtime can be achieved without any loss of work. It does not suffer from performance degradation also. The same technique can be applied to single server or clusters also.

Aye MyatMyatPaing and Ni LarThein [17] presented stochastic Petri nets model for time-based rejuvenation policy for VMM to analyze the availability of virtualized server system and resource usage management algorithm to support live VM migration.

Running VM can be migrated from one physical machine to another using live migration technology without outage. The VMM running on the physical machine suffers aging as the time degrades. When the software aging or some potential anomaly happens in one of the physical machine in the resource pool, the rejuvenation manager of management server will trigger a rejuvenation operation. All the new requests and sessions are migrated from the virtual machine of the aging affected physical machine to virtual machine of other physical machine in the resource pool. When the ongoing requests are finished in aging infected PM, these VMM will be rejuvenated. The authors used stochastic Petri nets (SPN) model with time-based rejuvenation policy.

Clustering technology, virtualization and methods for software rejuvenation can provide a very fast recovery to cut down the mean time to recovery to the minimum. It can achieve minimize downtime even in case of service restart.

Jean Araujo [18] presented an approach for software rejuvenation in cloud using Eucalyptus Cloud Computing framework. By using thresholds and time series based forecasting, the authors implemented a resource-usage-aware rejuvenation policy. An accurate forecast of critical points of resource degradation is possible by trend analysis of software aging –related data, matched against multiple time series models.

Utilization of software and hardware resources was analyzed in scenarios in which some cloud operations were performed continuously. A rejuvenation method was applied to prevent unavailability of the system by scheduled process restart.

C. Rejuvenation in High performance computing

The failure rate of super computers also increases with increase in the scale of high performance computing (HPC). The failures in HPC can be minimized by resetting or repairing the system components which is rejuvenation technique being followed. Checkpoint/restart is the most pervasive fault-tolerance (FT) technique used in HPC, yet it suffers from severe performance degradation, enormous requirement of storage, and significant overhead of writing to disks. Furthermore, this reactive fault tolerance approach cannot avoid long failure downtime, which is intolerable for performance-demanding applications.

CONCLUSION

In this paper, we have presented some of the causes for software aging and discussed the software rejuvenation techniques. We conclude that Software rejuvenation can be used to prolong the availability of services.

Software rejuvenation on virtualized environment is the emerging research area. As the availability of reliability of servers running on virtual environment needs to have zero down time in the current business scenarios, more optimized, effective software rejuvenation techniques for virtualized environment are the need of the hour. As many firms are moving from physical hardware usage to cloud services, development of very effective rejuvenation techniques for cloud environment will contribute a lot to service provider as well for the users.

REFERENCES

[1] J. Alonso, et al., A comparative experimental study of software rejuvenation overhead, Performance evaluation (2012), doi: 10.106 /j.peva.2012.09.002
[2] Thandar THEIN et al., "Availability Analysis and Improvement of Software Rejuvenation Using Virtualization", Economics and Applied Informatics, 2007, Issue 1, Pages 5-14.
[3] Tiago Pessoa, "An Eclipse Plugin to Support Code Smells Detection", INFORUM'2011 conference proceedings, Luis Caires e Raul Barbosa (eds.), 8-9 September, Coimbra, Portugal, 2011
[4] FoutseKhomh, An Exploratory Study of the Impact of Code Smells on Software Change-proneness , Proceeding WCRE '09 Proceedings of the 2009 16th Working Conference on Reverse Engineering Pages 75-84 IEEE Computer Society Washington, DC, USA 2009
[5] Dag I.K. Sjøberg, "Quantifying the Effect of Code Smells on Maintenance Effort", IEEE Transactions on Software Engineering, Volume 3, Issue 36, 2012
[6] Q. Ni, W. Sun, and S. Ma, "Memory leak detection in Sun Solaris OS", Proc. Int'l Symp. on Computer Science and Computational Technology, 2008.
[7] Nikita Salnikov-Tarnovski, "Why is your software aging?",http://plumbr.eu/blog/why-is-your-software-aging, August 15, 2013.

[8] AutranMacêdo, Taís B. Ferreira, RivalinoMatiasJr, "The mechanics of memory-related software aging", IEEE International Workshop on Software Aging and Rejuvenation - WoSAR , 2010

[9] G. Carrozza, D. Cotroneo, R. Natella, A. Pecchia, S. Russo, "Memory Leak Analysis of Mission-Critical Middleware", Journal of Systems and Software, 2010

[10] Vladimir˘Sor and SatishNarayanaSrirama, "A statistical approach for identifying memory leaks in cloud applications", CLOSER 2011 - International Conference on Cloud Computing and Services Science.

[11] Vaidyanathan K., , "A measurement-based model for estimation of resource exhaustion in operational software systems", Proceedings 10th International Symposium on Software Reliability Engineering,1999

[12] Michael Grottke, RivalinoMatias Jr. and Kishor S. Trivedi, "The Fundamentals of Software Aging", 19thInternational Symposium on Software Reliability Engineering, 2008.

[13] A.T. Tai, L. Alkalaj, and S.N. Chau, "On-Board Preventive Maintenance: A Design-Oriented Analytic Study for Long-life Applications", Performance Evaluation, vol 35, no.3-4, pp. 215 -232, 1999.

[14] KalyanaramanVaidyanathan, "Proactive management of software systems: analysis and implementation", Doctoral Dissertation, Duke University Durham, NC, USA ©2002

[15] Dan Pelleg et al. , "Vigilant—Out-of-band Detection of Failures in Virtual Machines", ACM SIGOPS Operating Systems Review, Volume 42, Issue 1, January 2008, Pages 26-31

[16] Luis Moura Silva et al.,"Using Virtualization to Improve Software Rejuvenation", Sixth IEEE International Symposium on Network Computing and Applications (NCA 2007)

[17] Aye MyatMyatPaing and Ni LarThein, "High availability Solution: Resource Usage Management in Virtualized Software Aging", International Journal of Computer Science & Information Technology (IJCSIT) Vol 4, No 3, June 2012

[18] Jean Araujo et al," Software Rejuvenation in Eucalyptus Cloud Computing Infrastructure: a Method Based on Time Series Forecasting and Multiple Thresholds", Third International Workshop on Software Aging and Rejuvenation,2011.