International Journal of Advance in Computer Science & its application – IJCSIA 2018
Copyright © Institute of Research Engineers and Doctors , SEEK Digital Library
Volume 7 : Issue 2 [ISSN : 2250-3765] - Publication Date: 25 June , 2018

Machine Learning Framework for Software Aging Forecasting

[I M Umesh, Dr.G N Srinivasan, R B Ravi Varma]

***Abstract*—Innovations in electronics and hardware have led to the scenario of multiple softwares running on the same hardware resulting in multiuser, multitasking and virtualized environments. The reliability of such high performance computing systems depends both on hardware and software. The software systems, during operation accumulate errors or garbage leading to software aging which may lead to system failure and hazardous consequences. Software aging needs to be dealt with specialized techniques. To deal with software aging, a technique called software rejuvenation exists that reboots or re-initiates the software to avoid fault or failure. The detection of software aging is one of the important steps before rejuvenating the system as the rejuvenation process may lead to downtime which has a business impact. Hence, software aging prediction is gaining importance and is one among the trends in the recent research area in the field of software aging and rejuvenation. In this paper, it is explored how the machine learning technology can be used to predict software aging. The predictive power of the built in algorithms of machine learning tools can be used for software aging forecasting. The software aging model has been built using machine learning tools. The developed software aging detection model can be used to schedule the rejuvenation to prevent crash or performance degradation of software systems.**

***Keywords*— software aging, rejuvenation, virtualization, fault tolerant, metric.**

I. Introduction

One of the main issues in cloud services is software aging, be it Software as a Service (SaaS), Platform as a Service (PaaS) or Infrastructure as a Service (IaaS). Software aging is the type of software fault that reduces the performance of the software system. Software aging happens due to the accumulation of aging related bugs that consume resources leading to resource exhaustion. Aging effects are the result of error accumulation that leads to resource exhaustion. This status of softwares makes the system gradually shift from healthy state to failure prone state. The system performance metrics needs to be monitored in order to find the aging patterns while the system is running. The accurate prediction of time of aging need to be forecasted to initiate the necessary actions to counter the aging effects. The software aging consequences can be avoided by using the technique called software rejuvenation.

Cloud based software systems, in general are high performance computing systems with several virtual machines (VM) running simultaneously. They encompass multiple cloud service segments where it communicates with each other through certain interfaces or connections resulting in distributed processing. The services in their long-run operation accumulate numerous internal errors and garbage, thus resulting in software aging and probable failure or performance degradation [1]. Numerous large scale software applications and associated systems do suffer from performance degradation or even failure because of premature resource exhaustion. The huge resource consumption, fragmentation and error accumulation causes software aging. The reason for software aging cannot be determined due to the dynamic nature of operation. Thus, it is to be resolved at run time only as it occurs.

The above issues demand certain optimal and effective fault tolerant technique to avoid software failure [4] at runtime. To deal with the problem of aging, a number of fault tolerant approaches have been proposed such as redundancy oriented software failure and its recovery, rollback scheme based on check points etc., Recently, a novel preventive and proactive approach called ‘software rejuvenation’ has been proposed to deal with these aging issues. Software rejuvenation enables freeing up the resources, storage and deletion of garbage, system reconfiguration etc., that significantly reduces the probability of premature system failure and performance degradation caused due to software aging.

Detection of software aging is an important step as the rejuvenation process includes several activities which may disrupt services that in turn have business impact. In this paper, a model for detection of probable aging of software systems running on virtual machines is proposed. The proposed model examines the recent resource utilization using aging indicators at different level; VM level and application specific. The proposed model forecasts the probable software aging time and pre-emptive rejuvenation can be done avoiding crash or hang.

II. Related work

To deal with software ageing, software rejuvenation has emerged as a key technique that significantly reduces performance degradation, resource depletion and premature system failure [1]. Researchers have employed different techniques for aging detection and rejuvenation. Previous

I M Umesh
Research Scholar, Bharathiar University, Coimbatore, Tamil Nadu India
umesh.mphil@rvce.edu.in

Dr. G N Srinivasan
R V College of Engineering, Bengaluru, Karnataka India
srinivasangn@rvce.edu.in

R B Ravi Varma
R V College of Engineering, Bengaluru, Karnataka India
ravivarma@rvce.edu.in

studies have used measurement-based and model-based rejuvenation approaches. Measurement based approaches directly monitor system variables which are aging indicators and predict software aging patterns by analyzing the collected runtime data statistically. Model-based studies can be distinguished by the type of stochastic process used to model the phenomenon such as Markov-based and Petrinets.

Alonso et al. [2] evaluated machine learning algorithms such as decision trees, K-nearest neighbor, Random forest using the R statistical language for aging prediction. The researchers used different sets of values when the ML algorithm had parameters to create different configurations of the same algorithm. The results indicated Random Forest performs better than the rest of the models. Toshiaki Hayashi et al. [3] estimated the performance degradation by passively measuring the traffic exchanged by virtual machines. The authors have justified the selection of traffic characteristics as a performance information source by citing several advantages. This data along with the recorded traffic metrics was tested with the C4.5 machine learning classifier that constructed a decision tree to identify performance. This is a non-intrusive method of metrics collection as the traffic measurement is done on separate machine that is not a part of virtual environment. This facilitates the metrics extraction even under extreme performance degradation.

A number of approaches have been developed for enhancing software rejuvenation by means of appropriate scheduling to reduce downtime and availability maximization. These approaches are classified into two major classes. The first is periodical rejuvenation method that considers time and workload to perform scheduling and the second one incorporates adaptive and proactive rejuvenation where the rate of resource depletion and performance degradation is examined dynamically. Some researchers have suggested rejuvenation based on the time estimation [5-9, 10], and the predictive measurement approach [11, 12]. The time based solution uses various dynamic parameters such as workload, mean time to repair (MTTR), and failure distribution over certain defined period. A number of tools have been developed for scheduling, such as Continuous-Time Markov Chain (CTMC) [5], a Markov regenerative process (MRGP) [7], Stochastic Reward Nets (SRNs) [6,8,9] and others [10,13,14]. On the other hand, predictive paradigm continuously monitors the operational behaviour and performs triggering for rejuvenation in case of any fault occurrence resulting in degradation or downtime.

The virtualization based rejuvenation has emerged as a significant approach to achieve higher resource availability and minimal downtime [14,15,16,17]. The earlier studies [16] proposed a CTMC model to monitor behaviour of the virtualized system, which was later implemented for rejuvenation scheduling. The system availability on the basis of time based rejuvenation was employed and consolidation of virtual machines (VMs) was performed. An approach, VMSR was developed for software rejuvenation for application server systems [18]. CTMC approach to consolidate multiple VMs on a single host server was employed in some studies. The work done in [15,16] employed only the time based rejuvenation policy without taking into consideration of the VMM failure and its rejuvenation problem and since VMM being a very critical point of faults often plays significant role, has not been addressed. Araujo et al. [19] explored aging based fault estimation for private cloud infrastructure because of the accretion of memory leaks in Eucalyptus tool. A software aging detection was proposed in [20] where they explored anomalous behaviour such as CPU and memory utilization of the system. Authors Artur Andrzejak and Luis Silva [21] proposed an approach based on adaptive probing which allows for non-intrusive monitoring and creation of accurate performance models as a function of performed work (served requests). The authors have used machine learning to develop a software aging detection.

III. The proposed software aging forecasting model

This section discusses the proposed software aging forecasting model. The phases include identifying the aging indicators, setting up test bed, evaluating the overhead of used tools, model building and finally forecasting process.

A. Aging indicators identification

For building aging detection model, three metrics have been considered. It is necessary to monitor multiple metrics that reflect broader utilization of the resources. The metrics and justification for choosing these metrics is tabulated below. These metrics are aging indicators at VM level and application level.

Aging indicator	**Description**
Application response time	Application specific aging indicator that update application response time in milliseconds
CPU load	System-wide aging indicator that updates CPU load in percentage
Memory Availability	System-wide aging indicator that updates available memory in percentage

B. Experimental Setup

The experimental setup has physical machine with following configuration.

Host configuration	:	Core i5 3.3 G Hz, 500 GB HDD, 8 GB RAM
Hypervisor	:	VMware ESXi 6.0
VM configuration	:	Processor-Core i5, 2 GB RAM, 40 GB HDD

International Journal of Advance in Computer Science & its application – IJCSIA 2018
Copyright © Institute of Research Engineers and Doctors , SEEK Digital Library
Volume 7 : Issue 2 [ISSN : 2250-3765] - Publication Date: 25 June , 2018

The VM is created on the host with the configuration mentioned above and the operating system is installed. The application was installed which has online shopping functionality. The application was built on XAMPP platform. The next activity was to generate the load using a load generator to mimic the live conditions and capture the metrics from the virtualized environment.

Jmeter tool has been used to generate load of several users simultaneously. The tool simulates heavy load on server that can be used to analyze performance under varying conditions. During our study, several test runs were conducted varying the number of users from 5 to 300. The metrics values which are aging indicators are collected for building aging forecasting model using machine learning framework. The summary of load generated to mimic the live environment is tabulated below.

Concurrent users	Min 5 users, Max 300 users
Load type	Application access
Duration	One month

Once the application is given load, metrics collection is done using some metrics collector tool. For this purpose, PRTG, a network monitoring tool has been used. Figure 1 & 2 depict the workload graphs.

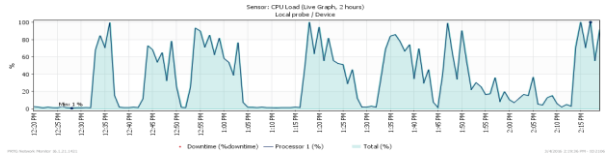

Figure 1: CPU workload graph during the test run.

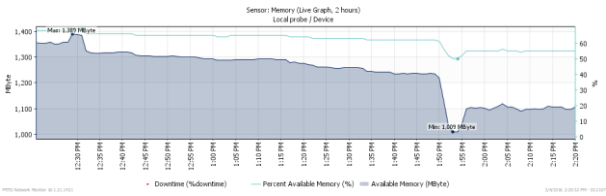

Figure 2: Memory Availability graph during the test run.

C. Overhead of using metrics collector

It is necessary to collect performance metrics without affecting the performance of physical and virtual machines at run time. The collected metrics indicate the resource consumption of various performance related parameters and hence non –intrusive approach has been used as the measurement program does not affect the hardware or software functionality and load.

The PRTG tool sensors use the programming interfaces of each device wherever possible. This means the administrator does not have to install additional client applications or agents on each device, thus simplifying and accelerating the setup and keeping the devices free of additional performance overhead.

To test the overhead of running PRTG, the application running on VM was stopped and the resource consumption was observed. The graph indicates the negligible overhead on performance of the application.

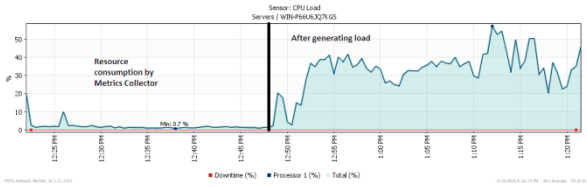

Figure 3: Performance overhead of metrics collector

IV. Model building using machine learning framework

The metrics which are aging indicators viz., CPU usage percent, Memory availability percentage, HTTP loading time are collected for a specific period of time. The collected metrics have been used to build aging detection model using WEKA machine learning framework. The metrics are given as input in ARFF format to WEKA decision tree REPTree which is a fast decision tree learner that builds a decision using information gain/variance and prunes it using reduced-error pruning (with backfitting).

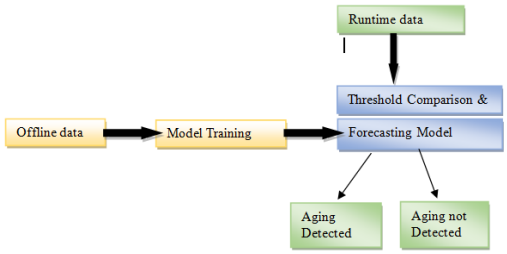

Figure 4: Aging forecasting model

V. The aging forecasting process

The process includes the steps as mentioned below.

Step 1: Just-in-Time metrics are moved to a database using a small module embedded in the aging detector application. Latest row is read from the database and compared with the threshold values for each aging indicator.

Step 2: Using Time Series prediction, future values are predicted. The basic learner used is Linear Regression. The forecasted values are analyzed to find aging patterns.

Step 3: The forecasting is done using model built in weka, the output is the values of aging indicators for next few hours.

Weightage is given for the results of static threshold and forecasting by two methods. The rejuvenation factor can be expressed as

$$R_F = P_{o_n}^{w_n} + M_{o_n}^{w_n} + R_{o_n}^{w_n} \quad (1)$$

where P , M and R represent the aging indicators CPU load, memory availability and http loading time respectively. o_n (n=1, 2, 3) represent aging status in three different type of methods i.e., threshold comparison, forecasting using time series and forecasting using model. w_n indicates the weightage given to the evaluated results in three methods.

$P_{o_n}^{w_n}$ is the sum of CPU load weightage, $M_{o_n}^{w_n}$ is the sum of Memory availability weightage and $R_{o_n}^{w_n}$ is the sum of http load time in different methods.

Rejuvenation factor R_F becomes 'Y' if overall weightage crosses the value X. The rejuvenation will be triggered once the value of rejuvenation R_F is found to be 'Y'. The overall process flow is shown in the figure 5.

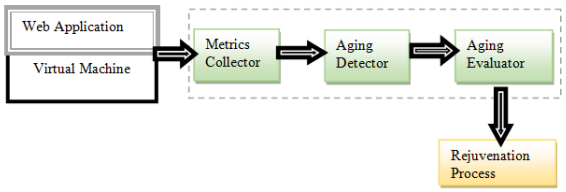

Figure 5: Aging forecasting process flow

The downtime of critical service is expensive in the current business scenario. The implementation of software aging detection model is very much useful to schedule proper rejuvenation schedule in order to increase the availability of services which the business value. Figure 6 indicate the importance of software aging detection model in the live environments.

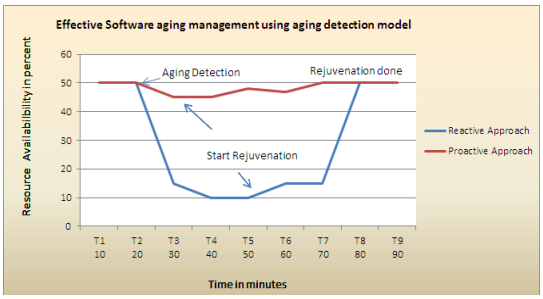

Figure 6: Effective Software Aging management using aging detection tool

V. Rejuvenation

Once the software aging is detected by using the proposed model, rejuvenation of aged VM is to be carried out. For rejuvenation of aged VM, numerous techniques exists which are discussed briefly in this section.

Live migration of virtual machines is one of the widely adapted techniques that migrates VM from one physical machine to another physical machine without causing outages. The mechanism includes shifting of new requests from affected virtual machine to other virtual machine in the resource pool, rejuvenating the aged VMM and shifting back the services to original VM. Some other techniques in practice include performing rejuvenation of VM and VMM simultaneously that reduces downtime and heals the software aging impacts on both VM and VMM. The rejuvenation process includes various steps like detection of VMM aging, preparation for VMM rejuvenation and restarting VMM and VM.

VI. Conclusion

The proposed fault tolerant paradigm successfully incorporates multi parameter (CPU Utilization, memory etc.,) based aging detection, which then can be used for scheduling rejuvenation. Thus, the proposed system can accomplish realistic and better aging forecasting and dynamic scheduling. In addition, the implementation of multiple parameters for aging detection enables the proposed system to be more realistic and responsive. The overall proposed system can be a potential candidate for fault tolerant software aging detection for cloud infrastructures as well as online/offline software applications.

References

[1] Domenico Cotroneo, Roberto Natella, Roberto Pietrantuono, Stefano Russo, "Software aging and rejuvenation: where we are and where we are going", 3rd International Workshop on Software Aging and Rejuvenation, pp 1–6, 2011

[2] Alonso, J. Belanche, L. Avresky, .R. "Predicting software anomalies using machine learning techniques", In Proc 10th IEEE International Symposium on Network Computing and Applications (NCA), pp.163-170, 2011

[3] T. Hayashi and S. Ohta. "Performance Degradation of Virtual Machines via Passive Measurement and Machine Learning", International Journal of Adaptive, Resilient and automates systems, vol 5, pp 40-56, 2014

[4] Ravi Jhawar and Vincenzo Piuri, "Fault tolerance and resilience in cloud computing environments", Cyber Security and IT Infrastructure Protection, J. R. Vacca, Eds. Elesvier: USA, pp 1–28, 2014

[5] Yennun Huang, Chandra Kintala, Nick Kolettis, N. Dudley Fulton, "Software rejuvenation analysis module and applications", Intl Symposium on Fault Tolerant Computing, Pasadena, pp 381-390,CA, 1995

[6] K Vaidynathan, Richard E Harper, Steven W Hunter, Kishor S Trivedi, "Analysis and implementation of software rejuvenation in cluster systems" , Joint International Conference on Measurement and Modeling of Computing Systems, pp 62-7, 2001

[7] Dazhi Wang, Wei Xie, Kishor S. Trivedi, " Performability analysis of clustered systems with rejuvenation under varying workload ", Performance Evaluation, Vol. 64, pp 247-265, 2007

[8] Yun Liu, Yue Ma, James J. Han Haim Levendel, Kishor S. Trivedi, "Modeling and analysis of software rejuvenation in cable modem termination systems", 13th International Symposium on Software Reliability Engineering (ISSRE), pp 159-172, 2002

[9] Yun Liu, Yue Ma , James J. Han , Haim Levendel , Kishor S. Trivedi, "A proactive approach towards always-on availability in broadband cable networks", Computer Communications, Vol. 28. , pp 51-64. 2005

[10] F. Salfner, K. Wolter, "Analysis of service availability for time-triggered rejuvenation policies", Journal of Systems and Software, Vol. 83, No. 9, pp 1579-1590, 2010

[11] Lei Li , K. Vaidyanathan, K. S. Trivedi, " An Approach to Estimation of Software Aging in a Web Server", Int'l Symp. Empirical Software Eng. (ISESE), pp 45-52, 2002

[12] Kenichi Kourai, Fast and Correct Performance Recovery of Operating Systems Using a Virtual Machine Monitor, International conference on Virtual execution environments, Vol.46, pp 99-110, 2011

[13] Hiroyuki Okamura, Tadashi Dohi, "Comprehensive evaluation of a periodic checkpointing and rejuvenation schemes in operational software system", Journal of Systems and Software, Vol. 83, pp 1591-1604, 2010

[14] K Vaidynathan, Kishor S. Trivedi, "A Comprehensive model for software Rejuvenation", IEEE Transaction on Dependable and Secure Computing, Vol.2. No 2, pp 124-137, 2005

[15] Fumio Machida, Dong Seong Kim, Kishor S. Trivedi, "Modeling and Analysis of Software Rejuvenation in a Server Virtualized System", 2nd Workshop on Software Aging and Rejuvenation, San Jose, CA , pp 1-6, 2010

[16] Thandar Thein, Jong Sou Park, "Availability Analysis of Application Servers Using Software Rejuvenation and Virtualization", Journal of Computer Science and Technology, Vol. 24, No. 2, pp 339-346, 2009

[17] Thandar Thein, Sung-Do Chi, Jong Sou Park, "Availability Modeling and Analysis on Virtualized Clustering with Rejuvenation", Int'l Journal of Computer Science and Network Security, Vol.8. No. 9, pp 72-80, 2008.

[18] Kenichi Kourai, "Fast and Correct Performance Recovery of Operating Systems Using a Virtual Machine Monitor" , International conference on Virtual execution environments, Vol.46. No. 7, pp 99-110, 2011

[19] Matheus Melo, Jean Araujo , Rubens Matos , Julian Menezes , " Comparative Analysis of Migration-Based Rejuvenation Schedules on Cloud Availability", IEEE International conference on Systems, Man, and Cybernetics, Manchester, pp 4110–4115, 2013

[20] Jean Araujo, Rubens Matos, Paulo Maciel, Rivalino Matias, "Software Aging Issues on the Eucalyptus Cloud Computing Infrastructure", IEEE International Conference on Systems, Man, and Cybernetics, Anchorage, AK , pp 1411–1416, 2011

[21] Artur Andrzejak and Luis Silva, "Using Machine Learning for Non-Intrusive Modeling and Prediction of Software Aging", Network Operations and Management Symposium,. NOMS 2008. IEEE, pp 25-32, 2008

[22] Toshiaki Hayashi and Satoru Ohta , "Performance Degradation Detection of Virtual Machines via Passive Measurement and Machine Learning", International Journal of Adaptive, Resilient and Autonomic Systems, Vol 5, No. 2, pp 40-56, 2014

About Authors:

I M Umesh received his M. Phil Degree in Computer Science from Bharathidasan University in 2008. He is currently working as System Analyst at R V College of Engineering, Bengaluru and pursuing his Ph D from Bharahtiar University, Tamil Nadu, India. His research interests include Software

Dr G N Srinivasan received his Ph D Degree in Computer Science from Avinashilingam University, Tamil Nadu in 2010. He is currently working as Professor at R V College of Engineering, Bengaluru. His research areas include Software Engineering, Information Retrieval and Data Mining

R B Ravi Varma received the Master degree from S.V. University, Tirupathi and M.S Degree in Information Technology from Mangalore University. He is currently working as System Analyst at R V College of Engineering, Bengaluru. His Research interests include Cloud Computing and Machine Learning.