Indonesian Journal of Electrical Engineering and Computer Science
Vol. 3, No. 3, September 2016, pp. 572 ~ 578
DOI: 10.11591/ijeecs.v3.i2.pp572-578

Optimum Software Aging Prediction and Rejuvenation Model for Virtualized Environment

I. M. Umesh*[1], Dr. G. N. Srinivasan[2]
[1]Research Scholar, Bharathiar University, Coimbatore, Tamil Nadu, India
[2]R.V.College of Engineering, Bengaluru, Karnataka, India
*Corresponding author, e-mail: umesh.mphil@rvce.edu.in

Abstract

Advancement in electronics and hardware has resulted in multiple softwares running on the same hardware. The result is multiuser, multitasking, multithreaded and virtualized environments. However, reliability of such high performance computing system depends both on hardware and software. For hardware, aging can be dealt with replacement. But, software aging needs to be dealt with different techniques. For software aging detection, a new approach using machine learning framework is proposed in this paper. For rejuvenation, the proposed solution uses Adaptive Genetic Algorithm (A-GA) to perform live migration to avoid downtime and SLA violation. The proposed A-GA based rejuvenation controller (A-GARC) has outperformed other heuristic techniques such as Ant Colony Optimization (ACO) and best fit decreasing (BFD) for migration. Results reveal that the proposed aging forecasting method and A-GA based rejuvenation outperforms other approaches to ensure optimal system availability, minimum task migration, performance degradation and SLA violation.

Keywords: *software aging, rejuvenation, virtualization, fault tolerant, metrics*

1. Introduction

Cloud based software systems, in general are high performance computing systems with several virtual machines (VM) running simultaneously. They encompass multiple cloud service segments wherein they communicate 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 exact 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 techniques to avoid software failure at runtime [2]. 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 aging of software systems running on virtual machines and rejuvenation is proposed. The proposed model forecasts the probable software aging effect using machine learning framework and pre-emptive rejuvenation of virtualized environment using adaptive genetic algorithm based live migration technique. The proposed model enhances the service availability of cloud systems by avoiding crash or hang that may occur due to software aging.

Received April 2, 2016; Revised July 25, 2016; Accepted August 10, 2016

2. Related Work

Software rejuvenation has emerged as a key technique to deal with software aging which significantly reduces performance degradation, resource depletion and premature system failure [1]. A number of approaches have been developed to enhance 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. The time based scheme intends to estimate 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) and Markov regenerative process (MRGP) [3, 5]. Lei Cui, et al., conducted studies in virtualized environment for software aging effect and concluded that aging effects are comparatively more in virtual machine than in physical machine [4]. Jing Liu, et al., proposed an adaptive failure detection method depending on the CPU and memory usage of certain VMs. The transmission delay of monitoring packets between service components and the AFD (Aging failure Detector) unit are chosen as the basic runtime metrics for aging detection [6]. Yongquan Yan proves that choosing a proper data set is more important than choosing a method for software resource consumption forecasting [7]. Some researchers have suggested rejuvenation based on the time estimation [8]. Previous studies also have discussed the predictive measurement approach that can be used for aging detection [9, 10]. Xiaozhi Du, et al., proposed software aging prediction model based on extreme learning machine (ELM) for a real VOD system [17].

Payal Kulkarni has proposed a method of shifting of application on virtual machine to avoid workload. Another contribution of the work is the specification of a time-based policy adapting the rejuvenation [11]. Fumio Machida, et al., proposed a combined server rejuvenation technique that performs VM rejuvenation simultaneously with VMM by rearranging the placement of VMs each time when VMM rejuvenation is performed [12]. Kenichi Kourai, et al., claim that only a VMM should be rebooted when only the VMM needs rejuvenation i.e., rebooting operating systems should be independent of rebooting an underlying VMM [13]. Some researchers have proposed a CTMC model to monitor behaviour of the virtualized system, which was later applied for rejuvenation scheduling [14]. Aye Myat Myat Paing, et al., proposed a comprehensive availability model for both VM clustering software rejuvenation model and VM migration based software rejuvenation model [15]. Some of the researchers have also developed an approach, VMSR for software rejuvenation to implement on application server systems [16]. CTMC approach to consolidate multiple VMs on a single host server was employed in some studies. Using virtualization technology, management of server hardware and software becomes more practical and availability can be enhanced [18].

The outcomes of the review of related work indicate the necessity of non-intrusive, platform independent software aging detection technique and need for improvement in rejuvenation strategies applied to virtualized environment that increases the trust-worthiness of the software system.

3. The Proposed Aging Forecasting and Rejuvenation Model

In this section, the proposed dynamic aging forecasting and rejuvenation model has been discussed.

3.1. System Model

The proposed system for aging detection and rejuvenation can be characterized in terms of its novelties for machine learning framework based aging forecasting, enhanced selection and VM migration policies based rejuvenation. The live migration doesn’t have any significant impact on the internal states of VM. However, live migration of the aged VM can take it to the fresh state host. Hence, live migration can significantly control aging. In our proposed model, hypervisor (VMM) software aging is neutralized that can affect VM straightforwardly. Live migration is performed so as to accommodate aged VMs to the VMM without any bug related accumulations.

In this paper, a dynamic software aging prediction and rejuvenation model has been proposed for applications running on virtualized environment. The proposed model encompasses multiple interconnected components or services with numerous functional software components. A metrics collector tool has been used to monitor VMM functions for aging forecasting and scheduling rejuvenation. Figure 1 represents the overall architecture of the proposed controller based software aging and rejuvenation model. The aging detector retrieves key performance indices about VMMs like CPU utilization and memory availability. The run time metrics collected from running VMMs are fed to aging forecasting model built using machine learning framework. Once a VMM is detected with highest aging probability or proneness, the proposed mechanism performs migration of the VM to the new VMM (host node) as per resource availability and scheduling. It significantly reduces the computational cost and probability of downtime.

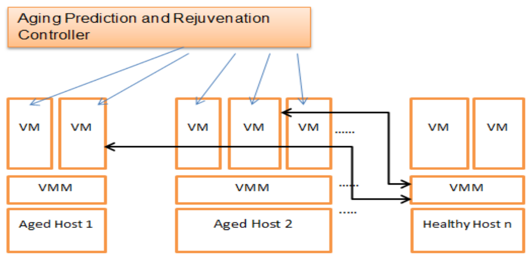

Figure 1. Architecture

3.2. Aging Forecasting

This section discusses the proposed software aging forecasting model. In cloud environments, the VMs run on a Virtual Machine Monitor (VMM), also called hypervisor. This component is liable to suffer failures or hangs due to software aging. In the proposed aging detection model, the metrics collected from VMM and VM are used which reflects the status of virtual environment. The metrics collected here are CPU usage and Memory usage as the aging of these resources directly impact the performance.

The application running on a VM is given load using a load generator tool to mimic the live environment and the metrics which are aging indicators are captured using a network monitoring tool. Several runs are conducted and the data set obtained is used to build aging forecasting model using machine learning framework. Figure 2 depicts the CPU workload graph.

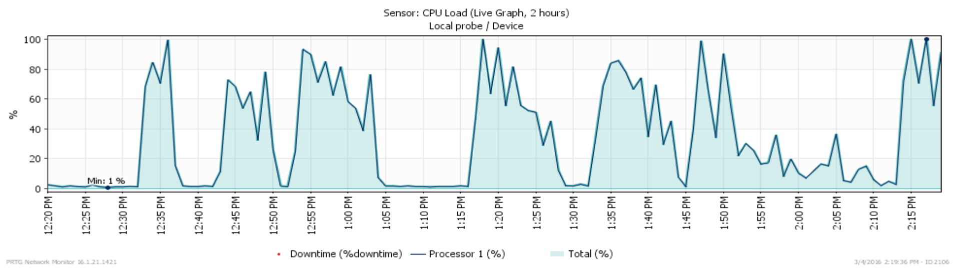

Figure 2. CPU Workload Graph during the Test Run.

The aging forecasting process includes the steps as mentioned below.

1. The metrics are input to aging forecasting model that is already built to forecast the values of aging indicators of VMM and VM. The metrics include latest data, previous days data collected during the same time interval under similar workload. The obtained results are analyzed.

2. Metrics of previous two hours is extracted and fed to time series forecasting algorithm of a machine learning framework; the basic learners used are Linear Regression and Gaussian Processes. The forecasted values are analyzed to find aging patterns.

3. 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.

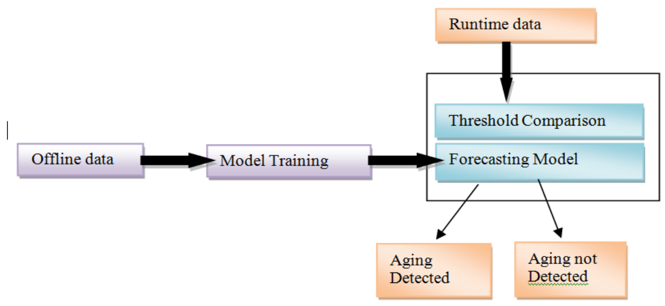

Figure 3. Aging Forecasting Model

Weightage is given for the results of static threshold comparison and forecasting. The rejuvenation factor can be expressed as:

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

Where P and M represent the aging indicators CPU load and Memory consumption 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 consumption weightage 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'.

3.3. Genetic Algorithm based VM migration

As VM relies on VMM to interact with hardware, VMM aging affects VM directly. VM Migration takes VM to a fresh state Physical Machine. In live migration, the focus is on nullifying the aging effects of hypervisor (VMM). After migration, VM leverages the OS software without software aging accumulation. After a VM migration, VM moves to a VMM without aging-related bugs accumulation.

The proposed rejuvenation model is based on live migration concept and hence avoids shutting down problem or rebooting. This approach significantly reduces the downtime and enhances the steady state system availability. In this paper, an evolutionary computing algorithm called Adaptive Genetic Algorithm based Rejuvenation Controller (A-GARC) has been proposed for allocating or placing aged VM onto the healthy host. A-GARC schedules live migration based on the outcome of the aging forecasting. Once the aged VMM is detected, A-GARC triggers the rejuvenation scheduler automatically to start migrating aged VM to the healthy host. The targeted host accommodates VM and thus ageing accumulation on that VM is avoided to ensure undisrupted function, maximum steady state system availability.

Genetic Algorithm based A-GARC scheme intends to reduce cloud (web server) related SLA violation and performance degradation. A-GARC algorithm takes feature mapped values of healthy hosts and aged VMs as the population. Here the individual are the part of a tree, where A-GARC (rejuvenation controller) functions as the root, and aged VMs are child nodes. A-GARC algorithm executes crossover function to select the best host for accommodating aged VMs. The algorithm used for genetic algorithm (GA) based task placement or relocation is given as follows:

Step-1 Create VM and install applications on certain host node
Step-2 Forecast software ageing probability based on aging indicator metrics
Step-3 Perform Initial mapping of VMs and connected hosts and then use it as initial population for GA
Step-4 Identify the aged hosts using Step-2 so as to avoid any probable SLA violation and conflict during rejuvenation
Step-5 Identify the list of VMs allied with each host node, which are not under migration, so that SLA performance violation could be avoided
Step-6 Perform aged VMs selection and retrieve the list of all VMs ready for migration
Step-7 Sort all VMs on the basis of its aging status
Step-8 Retrieve the complete active hosts possessing minimal workload and resource under-consumption
Step-9 Allocate VMs on the healthy active hosts
Step-10 Perform mapping for VMs and hosts and obtain proposed partner chromosome and add it to the population
Step-11 Retrieve the initial child by executing initial crossover between the roots (Step-3) and proposed partner (Step-10) based on candidate’s fitness value, where fitness value is obtained by:

$$\text{Fitness value} = \frac{\text{MIPS Utilized by all VMss on a host}}{\text{MIPS allocated to the host}}$$

Step-12 Mutate the first child chromosome and add to population by allocating some VMs randomly to the hosts

Step-13 Estimate the Roulette wheel (Rwheel) value using probability fitness (PF), where:

$$\text{PF} = \frac{\text{Candidate Fitnes Value (Candidate, rootVMList)}}{\text{Cumulative fitness value}}$$

$$\text{Rwheel} = \frac{\text{PF}}{\text{Number of hosts}}$$

Step-14 Select parent nodes using Rwheel probability value to perform Crossover
Step-15 Perform mutation for the Chromosome using Step-14 and add it to the population
Step-16 Perform Step 13-15 till stopping Criteria. (We have defined 100 generations as Stopping Criteria)
Step-17 From population, obtain the host having higher candidate fitness value for Placement
Step-18 Allocate VMs to the host using Step-17
Step-19 Feed the mapping input to Step-2 for next scheduling purpose

For algorithm development, Java-Eclipse has been used and CloudSim has been used as the simulation platform. To evaluate the robustness of the proposed system with realistic environment, host servers of different configurations have been used.

4. Results and Discussion

The proposed software aging forecasting and rejuvenation model incorporates controller based aging forecasting which uses forecasting model built using machine learning framework. A-GARC based rejuvenation approach executes live VM migration. In order to evaluate the performance of the proposed aging detection and rejuvenation model, other heuristics have also been applied. We have applied other heuristics such as Ant Colony Optimization (ACO) and best fit decreasing (BFD) based placement schemes. To evaluate the efficiency of the proposed system, the key performance parameters such as SLAV downtime, migration instances, SLA caused performance degradation and steady state system availability have been estimated.

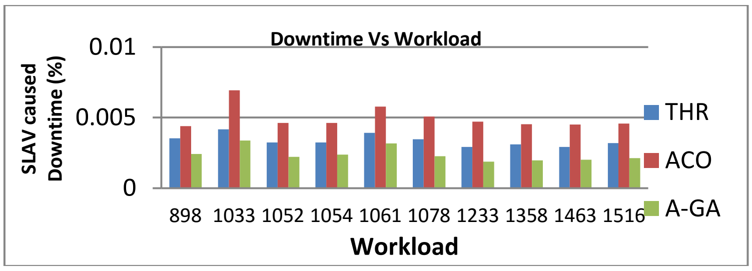

Figure 4. Downtime Due To Virtualization Based Software Rejuvenation

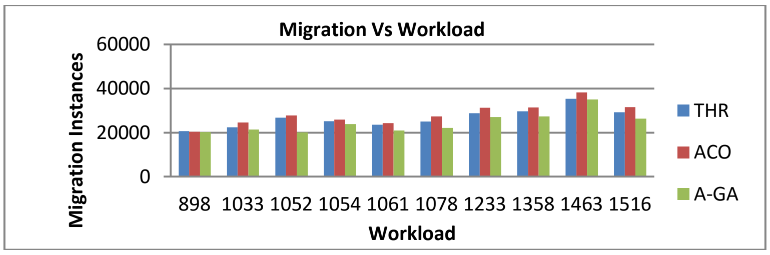

Figure 5. Total Task Migration Instances Due To Rejuvenation

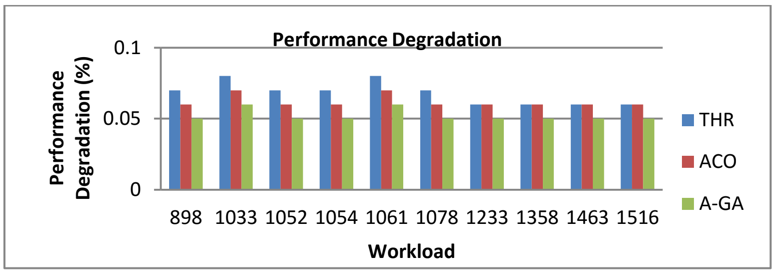

Figure 6. Performance Degradation Due To Virtualization Based Software Rejuvenation

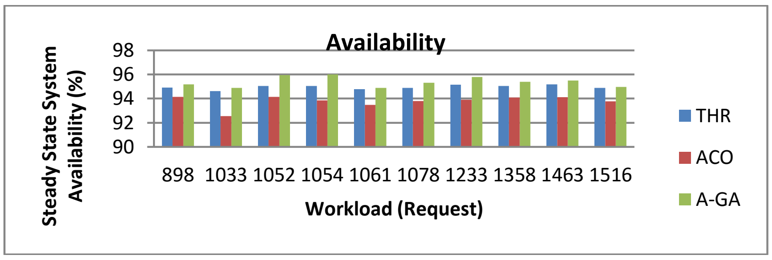

Figure 7. Steady State System Availability during Rejuvenation

Figure 4 represents the downtime, where it can be found that the proposed model outperforms other heuristics. Figure 5 represents the total task migration instances to exhibit rejuvenation. The fact that number migration directly impacts the downtime probability has been examined for the proposed system, where it has been found that the proposed A-GARC based scheme exhibits minimal migration and hence reduces the probability of downtime. The performance degradation is also reduced significantly in our proposed rejuvenation mechanism. For dynamic system like cloud web servers, it is very important to ensure maximum steady state system availability. The experimental evaluation reveals that the proposed system enables maximum steady state availability than other approaches.

Unlike conventional rebooting based approaches, our proposed scheme performs live migration and hence can be effective for real time utilities where system availability and SLA are vital to be accomplished. In addition, it can be cost effective than the other approaches.

5. Conclusion

In this paper, a new approach has been developed for aging forecasting which used resource consumption metrics as aging indicators. The Genetic algorithm based VM migration model has shown better results than the other heuristic approaches such as Ant Colony Optimization and best fit decreasing based VM allocation policies. The proposed scheme has outperformed other approaches in terms of SLAV, downtime, performance degradation and steady state availability. In future, the proposed mechanism can be enhanced with different thresholding schemes and task priority based migration scheduling.

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. 2011: 1-6.

[2] Ravi Jhawar, Vincenzo Piuri. Fault Tolerance and Resilience in Cloud Computing Environments. Cyber Security and IT Infrastructure Protection. 2nd Edition. Elsevier. 2014: 1-28.

[3] Yennun Huang, Chandra Kintala, Nick Kolettis, N. Dudley Fulton. *Software Rejuvenation Analysis Module and Applications.* Intl Symposium on Fault Tolerant Computing. CA. 1995: 381-390.

[4] Lei Cui Bo Li, Jianxin, Li, James Hardy, Lu Liu, *Software Aging in Virtualized Environments: Detection and Predictio*n. IEEE 18th International Conference on Parallel and Distributed Systems (ICPADS). 2012: 718-719.

[5] Dazhi Wang, Wei Xie, Kishor S Trivedi. Performability analysis of clustered systems with rejuvenation under varying workload. *Performance Evaluation.* 2007; 64: 247-265.

[6] Jing Liu, Jiantao Zhou, Rajkumar Buyya. *Software Rejuvenation based Fault Tolerance Scheme for Cloud Applications*. IEEE 8th International Conference on Cloud Computing. 2015: 1115-1118.

[7] Yongquan Yan. A Practice Guide of Predicting Resource Consumption in a Web Server. *Review of Computer Engineering Studies*. 2015; 2(3): 1-8.

[8] F Salfner, K Wolter. Analysis of service availability for time-triggered rejuvenation policies. *Journal of Systems and Software.* 2010; 83(9): 1579-1590.

[9] Lei Li, K Vaidyanathan, KS Trivedi. *An Approach to Estimation of Software Aging in a Web Server*. International Symposium on Empirical Software Engineering-(ISESE). 2002: 45-52.

[10] Kenichi Kourai. *Fast and Correct Performance Recovery of Operating Systems Using a Virtual Machine Monitor*. International conference on Virtual execution environments. 2011; 46: 99-110.

[11] Payal Kulkarni. Software Rejuvenation and Workload Distribution in Virtualized System. *International Journal of Innovative Research in Computer and Communication Engineering*. 2015; 3(6): 5966-5973.

[12] Fumio Machida, Jianwen Xiang, Kumiko Tadano, Yoshiharu Maeno. *Combined Server Rejuvenation in a Virtualized Data Center*. 9th International Conference on Autonomic & Trusted Computing (UIC/ATC). 2012: 486-493.

[13] Kenichi Kourai, Shigeru Chiba. Fast Software Rejuvenation of Virtual Machine monitors. *IEEE Transactions on Dependable and Secure Computing*. 2012; 8(6): 839-851.

[14] Thandar Thein, Jong Sou Park. Availability Analysis of Application Servers Using Software Rejuvenation and Virtualization. *Journal of Computer Science and Technology.* 2009; 24(2): 339-346.

[15] Aye Myat Myat Paing, Ni Lar Thein. Stochastic Reward Nets Model for Time based software Rejuvenation in Virtualized Environment. *International Journal of Computer Science and Telecommunications.* 2012; 3 (1): 1-10.

[16] Kenichi Kourai. *Fast and Correct Performance Recovery of Operating Systems Using a Virtual Machine Monitor*. International conference on Virtual execution environments. 2011; 46(7): 99-110.

[17] Xiaozhi Du, Huimin Lu, Gang Liu. Software Aging Prediction based on Extreme Learning Machine. *TELKOMNIKA*. 2013; 11(11): 6547-6555.

[18] Ferdy Nirwansyah, Suharjito. Hybrid Disk Drive Configuration on Database Server Virtualization, *Indonesian Journal of Electrical Engineering and Computer Science.* 2016; 2(3): 720-728.