Indonesian Journal of Electrical Engineering and Computer Science
Vol. 8, No. 3, December 2017, pp. 589 ~ 596
DOI: 10.11591/ijeecs.v8.i3.pp589-596

Software Aging Forecasting Using Time Series Model

I. M. Umesh*[1], Dr. G. N. Srinivasan[2], Matheus Torquato[3]
[1]Research Scholar, Bharathiar University, Coimbatore, Tamil Nadu, India.
[2]Professor**,** R.V.College of Engineering, Bengaluru, Karnataka, India.
[3]Professor, Federal Institute of Alagoas (IFAL), Campus Arapiraca - AL, Brazil
*Corresponding author, e-mail: umesh.mphil@rvce.edu.in

Abstract

With the emergence of virtualization and cloud computing technologies, several services are housed on virtualization platform. Virtualization is the technology that many cloud service providers rely on for efficient management and coordination of the resource pool. As essential services are also housed on cloud platform, it is necessary to ensure continuous availability by implementing all necessary measures. Windows Active Directory is one such service that Microsoft developed for Windows domain networks. It is included in Windows Server operating systems as a set of processes and services for authentication and authorization of users and computers in a Windows domain type network. The service is required to run continuously without downtime. As a result, there are chances of accumulation of errors or garbage leading to software aging which in turn may lead to system failure and associated consequences. This results in software aging. In this work, software aging patterns of Windows active directory service is studied. Software aging of active directory needs to be predicted properly so that rejuvenation can be triggered to ensure continuous service delivery. In order to predict the accurate time, a model that uses time series forecasting technique is built.

Keywords*: Active Directory, Cloud computing, Virtualization, Software aging, Rejuvenation*

1. 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 needs 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. Software rejuvenation is the proactive technique proposed to counter software aging. Software rejuvenation mechanism involves a series of steps such as periodically stopping the application and restarting it after cleaning the internal state. Software rejuvenation executes the actions that include garbage clearance, flushing of buffer queues, reinitializing of the internal kernel tables and cleaning up of the file systems. Intrinsically, it cleans and restores the application operating environment. The rejuvenation methods vary in the approach they employ for carrying out the rejuvenation. Some of the techniques use time-based approach which rejuvenates the system at pre-determined time interval. The other techniques in practice are inspection-based which involves the monitoring of aging indicator metrics and apply rejuvenation mechanism during the forecasted period. As rejuvenation mechanisms employ reboot type of activity, the service may be disrupted which in turn has business impact.

Important research issue is to determine when and how often the software should be rejuvenated and the rejuvenation techniques to be followed to avoid the software aging effects. Usually cloud services have virtualized environment for optimized maintenance. Hence software aging on virtualized environment needs to be addressed. There exists a need to rejuvenate the different layers of the virtualized environment using innovative approaches to enhance the

Received August 15, 2017; Revised October 25, 2017; Accepted November 11, 2017

service availability by reducing the downtime to zero. The virtualized environment consists of different layers such as physical hardware, hypervisor (VMM), Virtual machines (VM), Operating System and Applications running on top of operating system. The Virtual Machine Monitor (VMM) also called as hypervisor is liable to suffer failures or hangs due to software aging [1]. The aging issues in these layers are to be monitored and necessary rejuvenation action to be triggered in order to enhance the service availability and reduce the downtime.

Windows active directory is a necessary service that is used to authenticate the users in the domain network. Software aging of this service may lead to hazardous consequences like service downtime which in turn has business impact. Hence, there exists a need to analyze the software aging patterns in the active directory service for proper estimation of rejuvenation schedule. In this work, analysis is done on live work environment and results are discussed.

The remaining sections, Section 2 presents related work, Section 3 details the software aging study followed by Section 4 that discusses about the results and analysis. The conclusion is covered in Section 5.

2. Related Work

Researchers have employed different techniques for aging detection and rejuvenation. Previous 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 Chains and Petri Nets.

Alonso et al., [2] evaluated machine learning algorithms such as decision trees, K-nearest neighbor, and 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.

Jing Liu et al., [4] proposed an adaptive failure detection method. The parameters chosen for aging detection are CPU and memory usage; the delay in transmission of packet between service components and aging failure detection module. The collected metrics are encapsulated into a packet and sent to the aging detector module. The procedure used achieves two tasks, packet arrival time and the information it carries. Some failure probability is expected if the message does not arrive on time. The CPU usage and free memory available is used to detect the aging severity. The aging severity is divided into four levels i.e., from L1 to L4. Aging Degree Evaluator module inserts it into centralized failure event queue. Based on aging degree number, this queue is ordered and the top events will be rejuvenated inevitably.

The research work of Yongquan Yan [5] indicates the significance of choosing the proper data set. The researcher proved that choosing the proper data set is more important than the method used to analyze the collected metrics. The work compares the resource utilization of a webserver that is not subjected to artificial load, but a true load which has aging patterns using linear and nonlinear methods. Lei Cui et al. [6] concentrated their work on finding the impact of software aging defect on virtual machine and physical machines. The aging rate in both the forms was calculated and compared. The outcome of this study indicate aging effects are more in virtual machines than physical machines which was caused due to aging effects in code of hypervisor or depletion due additional calls in VMM layer.

The outcome of the study of the existing techniques is that there is a scope for research in finding non-intrusive, platform independent and more accurate aging prediction techniques.

3. The proposed model for software aging forecasting

3.1 Data Collection

In order to find whether the aging patterns exist in long running applications, the study is conducted on long running service Windows Active Directory. For this study, an institute with an approximate user's strength of six thousand is chosen. It is necessary to monitor multiple metrics that reflect broader utilization of the resources. Two metrics are considered for this study i.e., CPU usage and memory availability which are aging indicators. The metrics are collected for a period of six months and analysis is done. The metrics and justification for choosing these aging indicators is given in Table 1.

Table 1. Aging Indicators

Aging indicator	Description
CPU usage	System-wide aging indicator that updates CPU load in percentage
Memory availability	System-wide aging indicator that updates available memory in percentage

To capture the data, a Network Monitoring tool called PRTG (Paessler Router Traffic Grapher) is used. Figure 1 & 2 depict the workload graphs during the data collection.

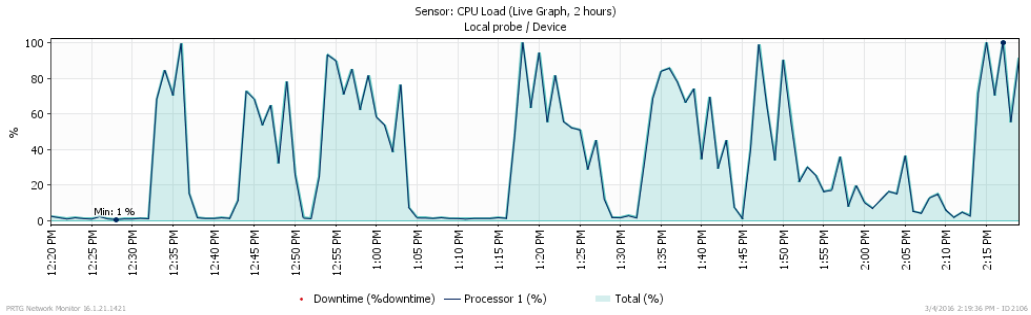

Figure 1. CPU workload graph during the metrics collection

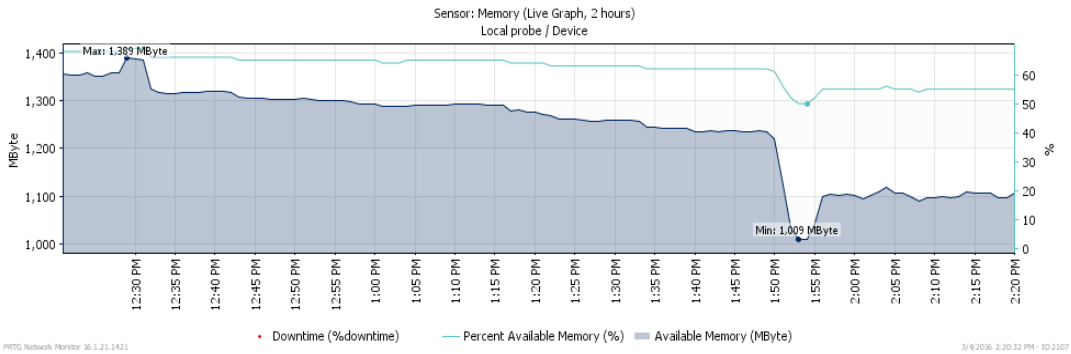

Figure 2. Memory Availability graph during the metrics collection

It is necessary to collect performance metrics without affecting the performance of server at run time. The collected metrics indicate the resource consumption of 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, resource consumption was observed from zero workload machine. The graph indicates the negligible overhead of metrics collector on performance of the application.

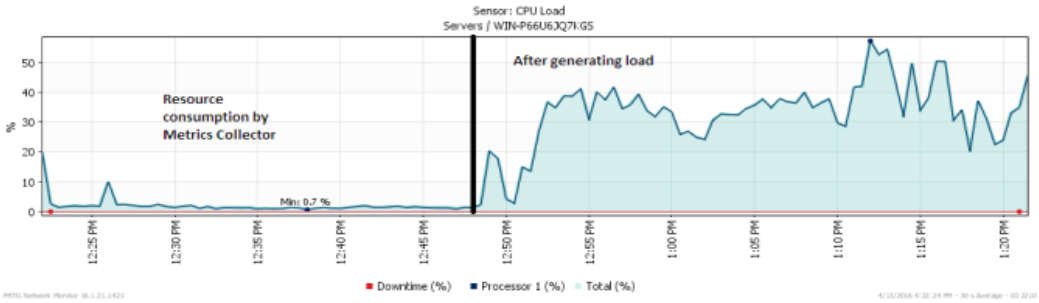

Figure 3. Performance overhead of metrics collector

3.2 Statistical analysis of collected data

Statistical analysis of software aging helps in collecting and scrutinizing collected metrics and finding software aging indicators. The presented analysis considers a period of six months. The monitoring interval was of two hours. Figure 4 has the results of CPU usage percentage of virtual machine that runs active directory. The Figure 5 depicts the results of Memory consumption. The plots also contain an approximated linear function of CPU usage behavior.

Software aging data analysis generally uses Mann-Kendall test to evaluate trend in the data [7-9]. The Mann-Kendall test [10] checks the null hypothesis, *H0*, which shows that there is no trend in the data during the time, against the alternative hypothesis, *H1*, which indicates an upward or a downward monotonic trend in the data. As software aging is a cumulative process, the Mann-Kendall test can be used to reveal trends of software internal state degradation. Among Mann-Kendall tests results we have the Z-value which is used to accept or reject the null hypothesis. Z-value close to zero suggests no trend in the data; a high absolute value indicates the existence of a trend. Figure 4 and Figure 5 indicate the CPU usage trend and memory usage trend respectively.

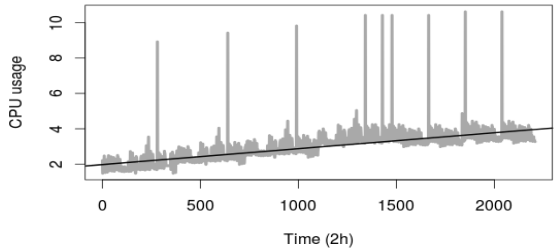

Figure 4. CPU usage trend

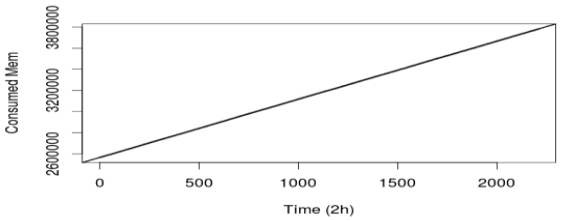

Figure 5. Memory Consumption trend

The Table 2 presents the results of statistical analysis made in the data. The results show Mann-Kendall Z-value is high than zero. Therefore, it is possible to reject the null hypothesis (no trend in the data). The positive value indicates an upward monotonic trend in the data. To calculate the slope of the monotonic trend, we used the Sen Method [11]. The Table 2 also presents the estimated slope and the 95% confidence interval for this slope.

Table 2. Statistical Analysis

Parameter	Memory Consumption	CPU Usage
Mann-Kendall Z-value	70.4	46.2
Estimated slope	550.9424 KB/2h	0.0009 %CPU/2h
95% confidence interval	(550.8521 KB/2h, 551.0322 KB/2h)	(0.0009 %CPU/2h, 0.001 %CPU/2h)

3.3 Prediction using time series forecasting

This section discusses the proposed software aging forecasting model. To prevent the system crash, it is necessary to predict resource exhaustion time. The CPU usage metrics has been tabulated in the Table 3. The data from different time slots of a day that were captured are entered in a table. This is because number of users (load) varies at different time slots. Average values of the CPU consumption percentage are tabulated. Based on this, moving average of four slots is calculated. Moving average (MA) is a calculation to analyze data points by creating series of averages of different subsets of the full data set. In this scenario, MA is calculated on average value of CPU usage metric.

Table 3. Predictions using time series

Time	Day	Time Slot	CPU Usage (%)	Moving Average	Center Moving Average	S_t, I_t	S_t	Deseas-onalized data	T_t	Predic tions
1	Day 1	1	12				0.30	40.39	45.25	13.44
2		2	70				1.44	48.58	46.38	66.84
3		3	63	48.25	48.625	1.30	1.22	51.76	47.52	57.83
4		4	48	49	50	0.96	1.03	46.77	48.65	49.93
5	Day 2	1	15	51	51.25	0.29	0.30	50.49	49.78	14.79
6		2	78	51.5	52	1.50	1.44	54.13	50.92	73.37
7		3	65	52.5	52.625	1.24	1.22	53.40	52.05	63.35
8		4	52	52.75	53	0.98	1.03	50.67	53.18	54.58
9	Day 3	1	16	53.25	53.5	0.30	0.30	53.85	54.32	16.14
10		2	80	53.75	54.75	1.46	1.44	55.52	55.45	79.90
11		3	67	55.75	55.875	1.20	1.22	55.05	56.58	68.87
12		4	60	56	56	1.07	1.03	58.46	57.72	59.23
13	Day 4	1	17	56	56.75	0.30	0.30	57.22	58.85	17.48
14		2	80	57.5	58.75	1.36	1.44	55.52	59.98	86.43
15		3	73	60			1.22	59.98	61.12	74.39
16		4	70				1.03	68.21	62.25	63.89
17	Day 5	1					0.30		63.38	18.83
18		2					1.44		64.52	92.97
19		3					1.22		65.65	79.91
20		4					1.03		66.78	68.54

The moving average is taken from an equal number of data on either side of a central value. This ensures that variations in the mean are aligned with the variations in the data rather than being shifted in time. Once the values are smoothed using moving average method, it can be further smoothed by center moving average method, a special procedure applied when the number of seasons is even. The relevant graph is shown in figure 6.

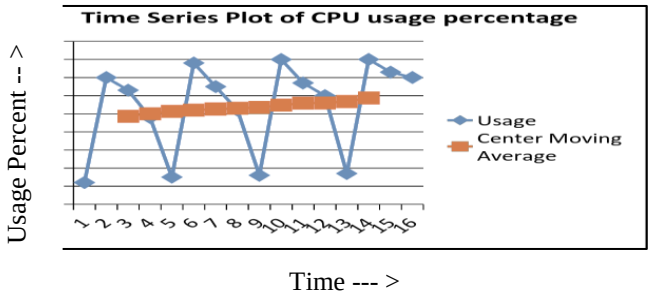

Figure 6. Graph indicating CPU usage and center moving average

The seasonal component S_t and irregular component I_t are obtained by dividing actual data by Centre Moving Average. By doing this we obtained the seasonality and irregularity of the data. The value 1.30 means, in the day 1, third time slot, seasonality and irregularity component is 30% above the base line data. The value 0.29 mentioned in fourth time slot of day 1 indicate that the seasonality and irregularity component is 71% below the baseline data. The next step is to get rid of irregularity component, I_t. This is done by averaging time slots of each day. The values generated are 0.30, 1.44, 1.22 and 1.03 for our data. This means during third time slot of day 1, the seasonal data is 22% higher than the baseline data. The irregularity has been removed by using this method.

The next step is to deseasonalize the data. This can be achieved by dividing the time series data with seasonal component. So far, we have removed the irregularities and deseasonalized the data. The next step is to find the trend component. In order to get trend component, Simple Linear Regression is to be performed using deseasonlized data as Y variable and time as X variable. The Simple Linear Regression and Co-efficients, Y Intercept and Slope obtained are 44.11 and 1.33 respectively. Now, the trend component can be obtained by using the formula,

$$T_t = \text{Y variable} + \text{X variable} * \text{time component} \quad (1)$$

The final values, predictions can be obtained by multiplying seasonal component with trend component. The summary data is presented in the Table 4.

Table 4. Summary Data

Regression Statistics	
Multiple R	0.876714
R Square	0.768627
Adjusted R Square	0.7521
Standard Error	3.064644
Observations	16

4. Results and Discussions

In order to evaluate the accuracy of the proposed technique, forecasting of the values is done for already collected metrics. The results indicate that the predicted values are accurate. The Figure 7 indicate graph that depicts the comparison of actual values and predicted values.

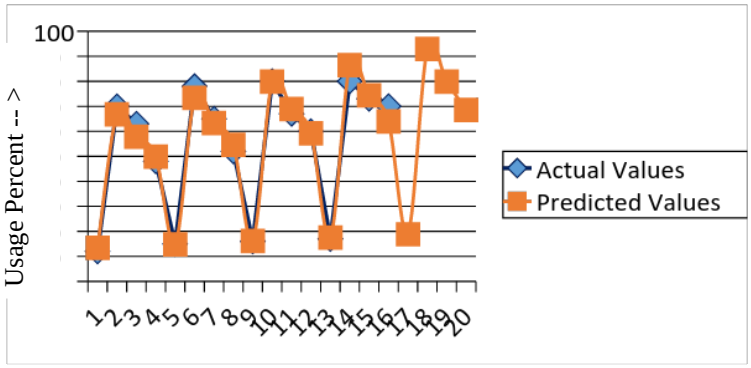

Figure 7. Comparison of actual values with predicted values

The predictions done here are for one day. The same technique can be used for forecasting the values of aging indicators for upcoming week or a month. The aging indicators taken here are CPU usage percentage and memory availability percentage. The other significant parameter to be used here is the threshold value of aging indicator. i,e, maximum load tolerable by the system. The threshold value can be obtained by studying the history of the service like when the system was crashed, the reason for failure and what were the value of parameters. Once the threshold values are identified, the weights are assigned to these parameters depending on whether the system is processor intensive application or memory intensive application. By varying the weights and the threshold, we can get different models of decision-making. The inputs to the model are x1 and x2 that indicate CPU usage and memory availability which are aging indicators. The weights w1 and w2 are assigned depending on the type of application. Depending on whether the weighted sum $\sum_j w_j x_j$ is less than or greater than some predetermined value, rejuvenation can be triggered.

5. Conclusion

The proposed forecasting technique can be a potential candidate for software aging prediction. As more and more educational institutes are using private cloud services as a part of their IT infrastructure, ensuring the continuous service delivery is important. This work helps in having better rejuvenation schedules thus avoiding downtime.

References

[1] Melo Matheus, Paulo Maciel, Jean Araujo, Rubens Matos, and Carlos Araujo. *Availability study on cloud computing environments: Live migration as a rejuvenation mechanism*. Proc. 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks. 2013; 1-6.

[2] Alonso, J. Belanche, L. Avresky, R. *Predicting Software Anomalies Using Machine Learning Techniques.* Proc 10th IEEE International Symposium on Network Computing and Applications (NCA). 2011; 163-170.

[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*. 2014; 40-56.

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

[5] Yongquan Yan. A Practice Guide of Predicting Resource Consumption in a Web Server. *Review of Computer Engineering Studies*. 2015; 1-8.

[6] Lei Cui, Bo Li, Jianxin Li, James Hardy, and Lu Liu. *Software Aging in Virtualized Environments: Detection and Prediction*. Proc. International Conference on Parallel and Distributed Systems, IEEE. 2012; 718-719.

[7] Grottke, M., Li, L., Vaidyanathan, K., & Trivedi, K. S. Analysis of software aging in a web server. *IEEE Transactions on reliability.* 2006; 55(3): 411-420.

[8] Garg, S., Van Moorsel, A., Vaidyanathan, K., & Trivedi, K. S. *A methodology for detection and estimation of software aging*. Proc. Ninth International Symposium on Software Reliability Engineering. IEEE. 1998: 283-292.
[9] Machida, F., Andrzejak, A., Matias, R., & Vicente, E. *On the effectiveness of Mann-Kendall test for detection of software aging*. Proc. IEEE International Symposium on Software Reliability Engineering Workshops. 2013: 269-274.
[10] Mann, H. B. Nonparametric tests against trend. *Econometrica: Journal of the Econometric Society*. 1945: 245-259.
[11] Sen, P. K. Estimates of the regression coefficient based on Kendall's tau. *Journal of the American Statistical Association*. 1968; 63(324):1379-1389.
[12] Xiaozhi Du, Huimin Lu, Gang Liu. Software Aging Prediction based on Extreme Learning Machine, TELKOMNIKA. 2013; 11(11): 6547-6555.
[13] Ferdy Nirwansyah, Suharjito. Hybrid Disk Drive Configuration on Database Server Virtualization. Indonesian *Journal of Electrical Engineering and Computer Science*. 2016; 2(3): 720 – 728.