

Software aging prediction – a new approach

Shruthi Parashivamurthy¹, Nagaraj Girish Cholli²

¹Department of Computer Science and Engineering, Global Academy of Technology, Bengaluru, India

²Department of Information Science and Engineering, RV College of Engineering, Bengaluru, India

Article Info

Article history:

Received Feb 16, 2022

Revised Sep 16, 2022

Accepted Oct 13, 2022

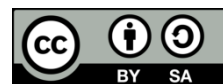
Keywords:

Algorithm
Machine learning
Prediction
Rejuvenation
Software aging

ABSTRACT

To meet the users' requirements which are very diverse in recent days, computing infrastructure has become complex. An example of one such infrastructure is a cloud-based system. These systems suffer from resource exhaustion in the long run which leads to performance degradation. This phenomenon is called software aging. There is a need to predict software aging to carry out pre-emptive rejuvenation that enhances service availability. Software rejuvenation is the technique that refreshes the system and brings it back to a healthy state. Hence, software aging should be predicted in advance to trigger the rejuvenation process to improve service availability. In this work, the k -nearest neighbor (k -NN) algorithm-based new approach has been used to identify the virtual machine's status, and a prediction of resource exhaustion time has been made. The proposed prediction model uses static thresholding and adaptive thresholding methods. The performance of the algorithms is compared, and it is found that for classification, the k -NN performs comparatively better, i.e., k -NN showed an accuracy of 97.6. In contrast, its counterparts performed with an accuracy of 96.0 (naïve Bayes) and 92.8 (decision tree). The comparison of the proposed work with previous similar works has also been discussed.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Shruthi Parashivamurthy

Department of Computer Science and Engineering, Global Academy of Technology

Bengaluru, India

Email: shrutip@gat.ac.in

1. INTRODUCTION

The performance degradation caused by software aging has hit various types of computing systems including virtualized cloud systems [1], web servers [2], [3], clusters [4] and online transaction processing systems [5]. The software aging concept has also impacted spacecraft systems [6] and military systems [7]. The impact may be loss of life in critical applications. Software aging happens because of unreleased file handles, data corruption, memory fragmentation, memory leaks, storage space fragmentation and round-off error accumulation. Software aging reduces the performance of cloud-based systems because of the complexity with which they are built. The system consists of servicing components and an execution environment. The system's boundary separates it from its environment, but its services are towards the surrounding environment [8]. In complex systems like the cloud, various levels like application level or operating system are prone to software aging [9]. Operating system-level effects are non-released memory, file handles, and sockets. Application-level effects include non-terminated threads, round-off errors, or data file corruption. It is very important to estimate the optimal time to trigger the rejuvenation to mitigate the software aging effects in cloud systems. Researchers have attempted to predict the time of software aging which can be seen in the previous works. Researchers have used threshold-based, statistics-based and machine learning approaches to estimate the software aging time. It is possible to predict the failure time of

the system using machine learning algorithms. The various aging indicators used to estimate resource exhaustion include memory and central processing unit (CPU) usage [10].

In this work, software aging prediction has been made using a new approach wherein virtual machine's current resource utilization status is fed to a machine learning model that classifies the virtual machines as healthy, aging-prone and aged using the k -nearest neighbor (k -NN) algorithm based new method. Static thresholding and adaptive thresholding methods have been used for aging prediction. Once the virtual machines are classified, rejuvenation is to be initiated for aging-prone and aged virtual machines. The rejuvenation process cleans up the system's internal state and brings the system back to its original state by removing the accumulated errors. The time when the rejuvenation is initiated is called rejuvenation trigger time. The time to trigger the rejuvenation has been forecasted using the new method in this work.

2. RELATED WORK

Based on the type of algorithm used, machine learning approaches are categorized into two types: classification approaches and regression approaches. In the classification method, the system status is classified as either stable or unstable. Forecasting of system failure can be done using a regression method. The procedure has been explained in [10]. Yan and Guo [11] developed a mechanism that forecasts software aging using a machine learning algorithm. Data was collected from a live commercial web server and the collected data was pre-processed. To identify a subset of the model parameters set, a feature selection algorithm was applied. A time series model was used for the prediction of selected parameters. To predict software aging, the model was built using machine learning algorithms. Sensitivity analysis was done to analyze how heavily outcomes depend on input variables. IIS webserver was used to apply the method. Experiment results were analyzed and found that the proposed method predicts software aging in the early phase of the system development life cycle.

Alonso *et al.* [12] performed a comparison of various regression algorithm families like linear regression, regression trees and hybrids. The researchers compared these algorithms in various scenarios and various aging concepts involved. The outcome of the experimentation indicated that phenomena performed better in the hybrid version i.e., MP5 between linear regression and decision tree. Due to the bugs in the software like unreleased threads or memory leaks, resource exhaustion was caused leading to aging phenomena. The model included linear piecewise models (i.e., a reasonable number of linear patches) capturing various aging slopes and speeds. In one of the previous works, three machine learning algorithms were used along with time series models for the prediction of software aging in web applications [13]. The three machine learning algorithms used are decision trees, naïve Bayes classifier and neural network model. The researchers built the models relating several system variables to aging trends like throughput and number of connections. This was based on the observation that aging phenomena can be approximated by making use of the piecewise linear model. The models in this work were trained using samples of data that were collected in preliminary experiments. The model built was able to predict the time-to-exhaustion (TTE) of system resources under different conditions.

Alonso *et al.* [14] had compared the large set of families like decision tree, linear discriminant analysis/quadratic discriminant analysis (LDA/QDA), random forest, support vector machines, naïve Bayes and k -NN for prediction of state in the context of a three-tier J2EE system. Andrzejak and Silva [15] compared four classification methods: ZeroR, decision tree, naïve Bayes, and support vector machines. These algorithms are compared under constant software aging injection rate by considering one aging indicator metric i.e., memory consumption. The results indicated that all classification methods performed similarly.

Jia *et al.* [16] used multiple linear regression algorithms to do a detailed analysis to predict web server parameters. In the first step, the system was pressurized using a pressure testing tool and collected data was pre-processed. The resource consumption trend was generated using the time series model in the second step. In the third step, the feature selection algorithm was used to select the best subset to be used as input parameters of the algorithm. In the fourth step, analysis was done using multiple linear regression and the aging process prediction. In the final stage, the algorithm feasibility is evaluated using evaluation metrics. The results indicated that this algorithm could predict the aging process in the allowable error range.

Liu and Meng [17] designed a method for predicting software aging which used an artificial bee colony algorithm. This achieves better prediction accuracy as the back propagation neural network optimization is achieved. The experiment results showed that the software aging prediction trend is more accurate than the traditional BP neural network. The proposed method also has a faster convergence speed and more prediction results which are more stable.

From the previous works, it can be observed that the concept of software aging is gaining importance. Researchers are trying to predict the software aging time to trigger the rejuvenation to avoid its impact. Similar works related to software aging prediction have been mentioned in the literature. There is a

scope for improvement or alternative methods to predict software aging. Considering these points, an attempt has been made to find a new approach to predict software aging in the proposed work. The proposed *k*-NN based method performs better compared to similar previous works.

3. MOTIVATION

The motivation to conduct this research originated since software aging is an emerging research area and machine learning is an emerging technology trend. Contributing to the area of software aging and research is satisfying work as this area is gaining momentum in recent years. The power of machine learning algorithms can be applied to achieve the objective. As the usage of cloud-based applications is increasing, it is the responsibility of the service provider to provide uninterrupted services to satisfy users. The inclusion of a module that avoids the impacts of software aging on a platform on which the application is hosted will make the service provider trustworthy. In this regard, the intended research helps cloud service providers also.

4. THE PROPOSED MODEL

Most of the services hosted on the cloud run in a virtualized environment. Virtualized environment includes various layers such as physical hardware, virtual machine, virtual machine monitor and applications running on a virtual machine. Figure 1 shows the typical cloud platform.

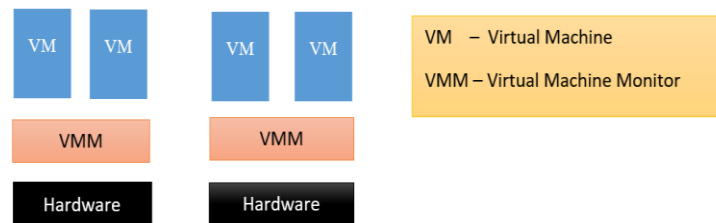


Figure 1. Typical cloud platform

The long-running applications on virtual machines suffer from software aging and hence there are chances of affecting service availability. The resource consumption metrics are collected from various layers of the virtualized environments on which cloud services are hosted to predict software aging. The metrics chosen to collect are aging indicator metrics. Figure 2 provides information regarding aging indicators. The metrics and justification for selecting these metrics are given.

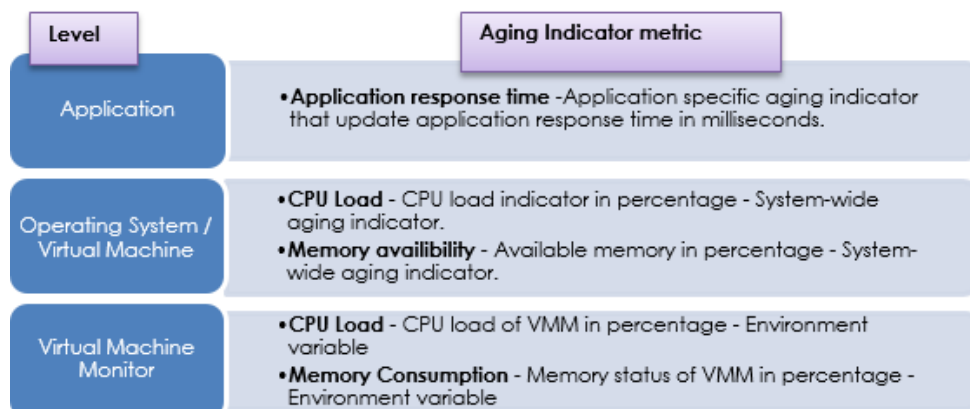


Figure 2. Aging indicator metrics

As the response time of any application indicates the performance of the system, application response time is one of the aging indicators usually considered in studies related to software aging. The most important operating system resources are CPU and memory. The metrics indicating the usage percentage of

these resources can also be considered as aging indicators. In this work, CPU consumption and memory consumption metrics are used to build the prediction model. The prediction model has been built using the following strategy. In this work, for the prototype, metrics collected from a virtual machine (VM) are used and the same technique can also be applied to a virtual machine monitor.

- a. The status of VM identification using the three methods
 - Static threshold: In the live environment, previous data related to resource usage was captured to know when the system was affected by software aging. CPU and memory usage metrics when the system failed were considered as static threshold values. At a certain point in time, various VMs status and resource utilization are captured to build the data set. The scatter graph is plotted using these values. The status of VM is identified by finding the nearest neighbors.
 - Adaptive threshold of CPU usage: The CPU usage history of k -NN is captured. Inter quartile range (IQR) statistical method is applied to find the adaptive threshold. The labeling of nearest neighbors is done based on the adaptive threshold value. The statuses are healthy, aging-prone, and aged.
 - Adaptive threshold of memory usage: The memory usage history of k -NN is captured and IQR is applied to find an adaptive threshold.
- b. Prediction of software aging
 - Once the aging-prone VMs are identified, the nearest aged neighbors are to be found.
 - Resource utilization trend of aged VMs is found and based on this, prediction of time required for aging-prone VMs to reach aged state is made.

Table 1 shows the steps followed for software aging prediction using k -NN based software aging prediction.

Table 1. Steps for software aging prediction using k -NN based method

No	Step
1	Load the dataset which consists of CPU usage and Memory usage percentage.
2	Determine the value of K, which indicates chosen number of neighbors.
3	Calculate the Euclidian distance between the query example and the current point for each point in the dataset. Add this attribute to the dataset.
4	Sort the dataset in ascending order of Euclidian distance (smallest to largest).
5	Pick the k number of rows from the sorted dataset.
6	Get the labels from selected k entries.
7	Return the mode of k labels.
8	Sort the CPU and memory utilization history of k points in the ascending order
9	Find the Median for CPU entries.
10	Identify Quartiles. Before median it is Q1 and after median it is Q3.
11	Find Q1 and Q3
12	Subtract Q1 from Q3 to obtain the Interquartile range $IQR = Q3 - Q1$
13	Calculate $MaxCPUThreshold = IQR3 + s.IQR$ ($s=1.5$)
14	Calculate $CPU\ Utilization \geq MaxCPUThreshold$ (status is aged) $CPU\ Utilization \leq maxCPUThreshold$ and $\geq maxCPUThreshold - 10\%$ (status is aging-prone) $CPU\ Utilization \leq MaxCPUThreshold - 10\%$ (status is healthy)
15	Classify VMs as per status calculated in step 14 comparing with current CPU utilization
16	Calculate $MaxMemThreshold = IQR3 + s.IQR$ ($s=1.5$)
17	Classify VMs as per status calculated in step 14 comparing with current Memory utilization
18	For VMs with status = <i>Aging-prone</i> Find nearest k 'aged' VMs End for
19	For each aged VM Identify the resource utilization trend. Find out the time taken for aged VM to reach the current status from aging-prone status. End for
20	Find out the average time taken by k aged VMs to reach aged status from aging-prone status.
21	On the basis of obtained average time taken, forecast the status of aging-prone

In step 13, the value of s is taken as 1.5 for the following reason. When John Tukey was inventing the box-and-whisker plot in 1977 to display the values, he picked $1.5 \times IQR$ as the demarcation line for outliers [18]. This has worked well, so researchers have continued using that value ever since.

The concept has been implemented using Python scripting language. Python is being used by researchers nowadays because of the various libraries it has that can support any type of research. Python includes libraries and frameworks related to machine learning. It is platform-independent and has a wide user community which makes it the first choice of research.

4.1. k-nearest neighbor algorithm

The *k*-NN algorithm is a supervised machine learning algorithm. It is applied to solve classification problems. The usefulness of the *k*-NN algorithm has been proved by the number of applications built based on this machine learning algorithm. In this research work, the *k*-NN algorithm has been used to classify the entity of virtualized environment as aged, aging-prone, or healthy.

4.2. Cluster creation

Figure 3 shows the sample dataset used for plotting the scatter graph. The scatter graph is plotted using the dataset to form the cluster. The rows used also included outliers. Outliers, in this work, are the aging indicator metrics that are usually not in the range of other points. It happens because of an unexpected spike in resource usage which is actually not a result of software aging. Outliers have been handled. Missing values are filled with suitable values. Clusters are formed based on the *x* and *y* values, in this case, CPU and memory consumption status. If the resource consumption reaches 80%, it is considered aged because service delivery will be hit. If one of the CPU or memory values reaches 70%, it is considered aging prone. The static threshold defined is based on our observation. This is also found in previous works [19].

The scatter graph has been plotted to visualize the clustered formed. Each of the clusters indicates a group of VMs with similar status. Figure 4 shows the scatter graph. The scatter graph has been plotted to visualize the clustered formed. Each of the clusters indicates a group of VMs with similar status. The status can be healthy, aging-prone, or aged. The different clusters in the scatter graph shown here indicate groups of entities belonging to various statuses: aged, aging-prone, and healthy.

	CPU	Memory	VM Status
10	2.000	4.000	Healthy
25	2.100	4.120	Healthy
32	2.100	4.150	Healthy
39	2.100	4.500	Healthy
86	2.150	3.520	Healthy
..
35	9.200	4.500	Aging-prone
69	9.256	6.115	Aging-prone
22	9.300	4.150	Aging-prone
113	9.650	4.445	Aging-prone
64	9.780	8.920	Aged

Figure 3. Sample dataset 1

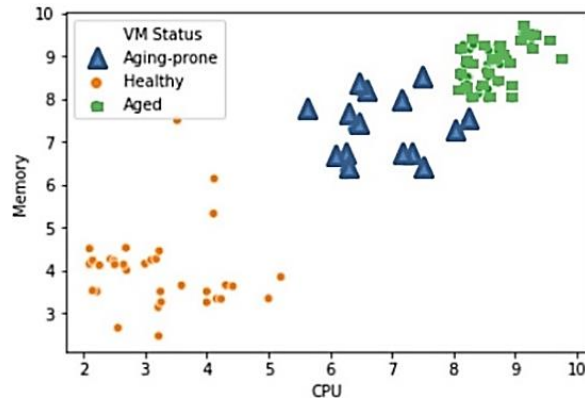


Figure 4. Scatter graph-clusters

4.3. Query point

Once the model is built, the status of any VM can be obtained by providing CPU and memory utilization percentages. This input is called query point. The nearest neighbors are found by calculating Euclidian distance. The formula for calculating the Euclidian distance is given in (1):

$$d(p, q) = \sqrt{(q1 - p1)^2 + (q2 - p2)^2} \tag{1}$$

where *p1* and *p2* are cartesian coordinates of the point *p* and *q1* and *q2* are the Cartesian coordinates of the point *q*, *d* is the distance between *p* and *q*. (*p* and *q* are points for which the Euclidian distance is calculated). After calculating the Euclidian distance, the nearest neighbors are found. The status of the majority of the neighbors indicates the status of the requested VM. The model which is built based on the *k*-NN algorithm returns the status as one of the three options: healthy, aging-prone, or aged. The value of *k* is to be provided which means the number of neighbors to be considered.

4.4. Adaptive threshold method

In this method, the resource usage history of *k*-nearest neighbors is captured. An IQR statistical method is applied to find the adaptive threshold. The labeling of nearest neighbors is done based on the adaptive threshold value. The statuses are healthy, aging-prone, and aged. Figure 5 shows the sample dataset. For example, there are seven data points. These points are the resource consumption percentages of previous days of a virtual machine (represented as T1 to T7 in the program). Table 2 shows the values.

$$IQR = Q3 - Q1 = 6.8 - 6.0 = 0.8 \tag{2}$$

$$Upper\ Threshold = Q3 + s.IQR \tag{3}$$

$$Upper\ Threshold = 6.8 + (1.5 \times 0.8) \tag{4}$$

$$Upper\ Threshold = 6.8 + 1.2 = 8 \tag{5}$$

The obtained status is tabulated, and the query point is labeled accordingly as shown in Table 3. The status of VM is calculated using three methods: static threshold, adaptive threshold using CPU metric, and adaptive threshold using memory usage metric.

Depending on the mode of k points in three evaluations, query point label is done. If three statuses are different, then static threshold status is considered. The screenshot of the program execution has been given in Figure 6.

	K-Value	T1	T2	T3	T4	T5	T6	T7
0	K1	70	53	70	47	63	56	69
1	K2	71	57	78	57	70	60	67
2	K3	71	55	71	53	65	62	74
3	K4	71	60	68	49	64	58	71
4	K5	69	59	72	47	63	57	70
..
115	K116	59	56	67	49	61	57	63
116	K117	70	58	72	54	65	59	61
117	K118	63	55	65	51	65	55	64
118	K119	67	54	64	45	63	55	64
119	K120	65	51	70	42	59	54	65

Figure 5. Sample dataset 2

Table 2. Resource consumption values

Data Points	5.2, 6.0, 6.2, 6.4, 6.7, 6.8, 7
Q2	6.4
Q1	6.0
Q3	6.8

Range: 0 (min)-10(max) indicate percentage of consumption

Table 3. Status of query point from three methods

Nearest neighbors	Status as per static threshold	Status as per current CPU utilization (adaptive threshold)	Status as per current memory utilization (adaptive threshold)
K1	Aged	Aged	Aged
K2	Aging prone	Healthy	Aging prone
K3	Aged	Aging prone	Aged
K4	Aging prone	Aged	Aging prone
K5	Aging prone	Aged	Aging prone

	Nearest Neigh	Static Thhold	CPU Utlz	Dyn Thhold	Mem Utlz	Dyn Thhold
0	0	Aging-prone	Aging Prone	Aging Prone	Aging Prone	Aging Prone
1	15	Aging-prone	Healthy	Healthy	Healthy	Healthy
2	4	Aging-prone	Aging Prone	Aging Prone	Healthy	Healthy

Figure 6. Output indicating the status

4.5. Prediction of software aging

As mentioned in the algorithm, the nearest k -aged VMs are identified. Here the procedure for one aged VM is given. Resource usage of aged VM is found which is previous days' data before it gets aged. Resource usage data is tabulated in Table 4.

Table 4. Resource usage of one virtual machine

Time	CPU Usage	Average	Status
Day 1	6.2	5.7	Healthy
Day 3	5.2		
Day 5	6.0	6.35	Healthy
Day 7	6.7		
Day 9	7	6.9	Healthy
Day 11	6.8		
Day 13	6.4	6.5	Healthy
Day 15	6.6		
Day 17	7.0	7.1	Aging-prone
Day 19	7.2		
Day 21	7.6	7.8	Aging-prone
Day 23	8		
Day 25	7.4	7.65	Aging-prone
Day 27	7.9		
Day 29	8.0	8.15	Aged
Day 31	8.3		

Range: 0 (min)-10(max) indicate percentage of consumption

Hence, current aging-prone VMs take 6 days to become aged VM. Figure 7 shows the screenshot of the program execution. As per the trend observed in the nearest 3 aged VMs, identify the time required for aged VM to become aged from aging-prone. Table 5 shows the resource usage of 3 virtual machines.

```

Number of days taken for transition from aging prone to aged status for VM1: 6
Number of days taken for transition from aging prone to aged status for VM2: 8
Number of days taken for transition from aging prone to aged status for VM3: 4
Average days taken for transition from aging prone to aged status for VMs: 6.0

```

Figure 7. Prediction part of random execution

Table 5. Resource usage of 3 virtual machines

Aged VM	No of days taken	Average days
VM-1	5	
VM-2	7	6
VM-3	6	

4.6. Rejuvenation

Software rejuvenation is the technique that refreshes the system and brings it back to a healthy state. The rejuvenation process is triggered for aging-prone VMs to improve service availability. Actions are triggered based on classification as depicted in Table 6.

Table 6. Aging status and actions

VM Status	Action taken	Remarks
Healthy	Rejuvenation not required	Observation continues.
Aging-prone	Forecasting is done to identify resource exhaustion time.	Rejuvenation is triggered before resource exhaustion happens.
Aged	Rejuvenation is triggered immediately.	The system returns to a healthy state after rejuvenation.

4.7. Evaluation of the proposed method

As the work is k -NN based new approach, the performance of the k -NN classifier has been compared with similar classifiers decision tree and naïve Bayes for the same data set. The result indicates that the k -NN algorithm performs better than the decision tree. The execution result is tabulated in Table 7. Details of previous research works for software aging prediction have been tabulated in Table 8. It can be observed that the proposed model of software aging prediction addresses the drawbacks in the previous works.

Table 7. Performance comparison

Algorithm	Data set size	Accuracy
k-nearest neighbor	115	97.6
Naïve Bayes	115	96.0
Decision tree	115	92.8

Table 8. Comparison of similar research works

Researchers	The highlight of the work	The proposed model
Fang <i>et al.</i> [20]	Instead of fixed thresholds, the method used in this work regularly regulates the thresholds by taking feedback information in the running process into account. Recommended adaptive threshold for aging detection	Adaptive thresholding is a part of the overall software aging prediction strategy in this research work.
Ahamad [21]	Found reasons for aging, effects of software aging. Concluded that it is impossible to stop software aging, but it is possible to reduce its speed and progress.	The software aging prediction method employed in this work enables rejuvenation to reduce the speed and progress of aging accumulation.
Liu <i>et al.</i> [22]	A monitoring agent in every VM collects metrics; CPU usage and free memory available to detect the aging severity. It is an intrusive method.	The tool used for collecting the metrics related to software aging is non-intrusive in this work. NMS tool captures metrics without adding overhead.
Yan [23]	Operating system parameters and database parameters in the running phase are collected using a built-in windows counter without disturbing the running system. Used IIS Webserver which is platform specific.	The model used in this work can be deployed on any platform like Windows or Linux. It is not platform-specific.
Cui <i>et al.</i> [24]	The rate of aging is more in virtual machines.	The proposed model is built for a cloud platform which is a virtualized environment. It justifies the chosen platform.
Umesh <i>et al.</i> [25]	Software aging forecasting using time series model. Not recommended method if there is a spike in resource usage.	The proposed model uses static and adaptive techniques which eliminates this concern.
Umesh and Seinivasan [26]	Used different methods for forecasting. Weightage given to different techniques is not acceptable in all scenarios.	The model proposed in this work improves the prediction accuracy.

5. CONCLUSION

In this work, an attempt has been made to forecast software aging. During the testing phase of software development, the application can be tested for all types of probable issues, but a problem like software aging must be dealt with during runtime only. It cannot be avoided; it can only be managed. As the accumulation of errors, lock contention, and data corruption, lead to this problem, the impact can be seen as the owner's loss as the service provider will lose the customers. Also, reduced performance and decreased reliability are other negative impacts of software aging. Even if all proactive measures are taken, the problem of software aging cannot be prevented, but it can only be managed. The only available solution is to predict the future status and pre-emptively rejuvenate the system. The aging forecasting is done using the new method. This research work can be one of the considerable contributions to the area of software aging and rejuvenation





REFERENCES

- [1] D. J. Dean, H. Nguyen, and X. Gu, "UBL: unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems," in *Proceedings of the 9th international conference on Autonomic computing - ICAC '12*, 2012, Art. no. 191. doi: 10.1145/2371536.2371572.
- [2] J. Zhao, K. S. Trivedi, M. Grottke, J. Alonso, and Y. Wang, "Ensuring the performance of Apache HTTP server affected by aging," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 2, pp. 130–141, Mar. 2014, doi: 10.1109/TDSC.2013.38.
- [3] H. Meng, X. Hei, J. Zhang, J. Liu, and L. Sui, "Software aging and rejuvenation in a J2EE application server," *Quality and Reliability Engineering International*, vol. 32, no. 1, pp. 89–97, Feb. 2016, doi: 10.1002/qre.1729.
- [4] V. Castelli *et al.*, "Proactive management of software aging," *IBM Journal of Research and Development*, vol. 45, no. 2, pp. 311–332, Mar. 2001, doi: 10.1147/rd.452.0311.
- [5] K. J. Cassidy, K. C. Gross, and A. Malekpour, "Advanced pattern recognition for detection of complex software aging phenomena in online transaction processing servers," in *Proceedings International Conference on Dependable Systems and Networks*, 2002, pp. 478–482. doi: 10.1109/DSN.2002.1028933.
- [6] E. Marshall, "Fatal error: How patriot overlooked a scud," *Science*, vol. 255, no. 5050, Art. no. 1347, Mar. 1992, doi: 10.1126/science.255.5050.1347.
- [7] A. Avritzer, R. G. Cole, and E. J. Weyuker, "Methods and opportunities for rejuvenation in aging distributed software systems," in *2008 IEEE International Conference on Software Reliability Engineering Workshops (ISSRE Wksp)*, Nov. 2008, pp. 1–6, doi: 10.1109/ISSREW.2008.5355518.
- [8] M. Grottke, R. Matias, and K. S. Trivedi, "The fundamentals of software aging," in *2008 IEEE International Conference on Software Reliability Engineering Workshops (ISSRE Wksp)*, Nov. 2008, pp. 1–6. doi: 10.1109/ISSREW.2008.5355512.
- [9] A. Gupta, B. R. Mohan, S. Sharma, R. Agarwal, and K. K., "Prediction of software anomalies using time series analysis – a recent study," *International Journal on Advanced Computer Theory and Engineering*, vol. 2, no. 3, pp. 101–108, 2013.





- [10] R. Pietrantuono, J. Alonso, and K. Vaidyanathan, "Measurements for software aging," *Handbook of Software Aging and Rejuvenation*, pp. 73–90, 2020.
- [11] Y. Yan and P. Guo, "A practice guide of software aging prediction in a web server based on machine learning," *China Communications*, vol. 13, no. 6, pp. 225–235, Jun. 2016, doi: 10.1109/CC.2016.7513217.
- [12] J. Alonso, J. Torres, J. L. Berral, and R. Gavaldà, "Adaptive on-line software aging prediction based on machine learning," in *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, Jun. 2010, pp. 507–516, doi: 10.1109/DSN.2010.5544275.
- [13] J. P. Magalhaes and L. M. Silva, "Prediction of performance anomalies in web-applications based-on software aging scenarios," in *2010 IEEE Second International Workshop on Software Aging and Rejuvenation*, Nov. 2010, pp. 1–7, doi: 10.1109/WOSAR.2010.5722095.
- [14] J. Alonso, L. Belanche, and D. R. Avresky, "Predicting software anomalies using machine learning techniques," in *2011 IEEE 10th International Symposium on Network Computing and Applications*, Aug. 2011, pp. 163–170, doi: 10.1109/NCA.2011.29.
- [15] A. Andrzejak and L. Silva, "Using machine learning for non-intrusive modeling and prediction of software aging," in *NOMS 2008 - 2008 IEEE Network Operations and Management Symposium*, 2008, pp. 25–32, doi: 10.1109/NOMS.2008.4575113.
- [16] S. Jia, C. Hou, and J. Wang, "Software aging analysis and prediction in a web server based on multiple linear regression algorithm," in *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, May 2017, pp. 1452–1456, doi: 10.1109/ICCSN.2017.8230349.
- [17] J. Liu and L. Meng, "Integrating artificial bee colony algorithm and BP neural network for software aging prediction in IoT environment," *IEEE Access*, vol. 7, pp. 32941–32948, 2019, doi: 10.1109/ACCESS.2019.2903081.
- [18] Purplemath, "Interquartile Ranges & Outliers," *Purplemath.com*. <https://www.purplemath.com/modules/boxwhisk3.htm> (accessed Nov. 08, 2022).
- [19] P. Kumar, *Forecasting cloud resource utilization using time series methods*. Degree Project in Computer Science and Engineering, Stockholm, Sweden, 2018.
- [20] Y. Fang, B.-B. Yin, G. Ning, Z. Zheng, and K.-Y. Cai, "A rejuvenation strategy of two-granularity software based on adaptive control," in *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*, Jan. 2017, pp. 104–109, doi: 10.1109/PRDC.2017.23.
- [21] S. Ahamad, "Study of software aging issues and prevention solutions," *International Journal of Computer Science and Information Security*, vol. 14, no. 8, pp. 307–313, 2016.
- [22] J. Liu, J. Zhou, and R. Buyya, "Software rejuvenation based fault tolerance scheme for cloud applications," in *2015 IEEE 8th International Conference on Cloud Computing*, Jun. 2015, pp. 1115–1118, doi: 10.1109/CLOUD.2015.164.
- [23] Y. Yan, "A practice guide of predicting resource consumption in a web server," *Review of Computer Engineer Studies*, vol. 2, no. 3, pp. 1–8, Sep. 2015, doi: 10.18280/rces.020301.
- [24] L. Cui, B. Li, J. Li, J. Hardy, and L. Liu, "Software aging in virtualized environments: Detection and prediction," in *2012 IEEE 18th International Conference on Parallel and Distributed Systems*, Dec. 2012, pp. 718–719, doi: 10.1109/ICPADS.2012.111.
- [25] I. M. Umesh, G. N. Srinivasan, and M. Torquato, "Software aging forecasting using time series model," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 8, no. 3, pp. 589–596, Dec. 2017, doi: 10.11591/ijeecs.v8.i3.pp589-596.
- [26] I. M. Umesh and G. N. Srinivasan, "Optimum software aging prediction and rejuvenation model for virtualized environment," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 3, no. 3, pp. 572–578, Sep. 2016, doi: 10.11591/ijeecs.v3.i3.pp572-578.

BIOGRAPHIES OF AUTHORS



Shruthi Parashivamurthy     is an Information Science and Engineering graduate. She also holds the M.Tech. degree in Computer Science and Engineering from VTU. She is pursuing a Ph.D. under VTU in the area of software aging and rejuvenation. She presently works as an assistant professor in the Department of Computer Science and Engineering, Global Academy of Technology, Bengaluru, Karnataka. She has a total of 7 years of experience in teaching and industry. She has published several papers in international journals. She is active in research and a life member of the CSI society. She can be contacted at shrutip@gat.ac.in.



Nagaraj Girish Cholli     is a Computer Science and Engineering graduate. He also holds the M.Tech degree in Computer Science & Engineering from IIT-Roorkee. He obtained Ph.D. from VTU in the year 2016 in the area of software aging and rejuvenation. He presently works as associate professor at the Department of Information Science and Engineering, R.V College of Engineering, Bengaluru, Karnataka, India. He has a total of 14 years of experience in teaching, research, and industry. He has published several research articles in international journals and presented papers at conferences. He is active in research, has filed patents and guiding several Ph.D. scholars. He is also a life member of ISTE and CSI society. He can be contacted at nagaraj.cholli@rvce.edu.in.