A workload-based nonlinear approach for predicting available computing resources

JIA Yunfei1,*, ZHOU Zhiquan2, and WU Renbiao1

1. Tianjin Key Laboratory for Advanced Signal Processing, Civil Aviation University of China, Tianjin 300300, China;
2. School of Computing and Information Technology, University of Wollongong, Wollongong 2522, Australia

Abstract: Performance degradation or system resource exhaustion can be attributed to inadequate computing resources as a result of software aging. In the real world, the workload of a web server varies with time, which will cause a nonlinear aging phenomenon. The nonlinear property often makes analysis and modelling difficult. Workload is one of the important factors influencing the speed of aging. This paper quantitatively analyzes the workload-aging relation and proposes a framework for aging control under varying workloads. In addition, this paper proposes an approach that employs prior information of workloads to accurately forecast incoming system exhaustion. The workload data are used as a threshold to divide the system resource usage data into multiple sections, while in each section the workload data can be treated as a constant. Each section is described by an individual autoregression (AR) model. Compared with other AR models, the proposed approach can forecast the aging process with a higher accuracy.

Keywords: software aging, nonlinear phenomenon, fault forecasting.

DOI: 10.21629/JSEE.2020.01.21

1. Introduction

Software aging refers to a type of performance degradation resulting from inadequate computing resources. Error conditions such as imperceptible memory leaks, uncollected system garbages, unreleased file locks and the like will permanently consume a lot of system resources. It is difficult to detect these error conditions at the testing phase because they merely emerge after longtime execution at the operational phase. A runtime process cannot provide good services unless sufficient CPU cycles, memory, and/or disk I/O bandwidth are provided.

Inadequate computing resources, as a result of software aging, may incur performance degradation or crashes. Hence, software aging is usually indicated by resource leaks. The speed of aging, which is indicated by the slope of resource leaks, is influenced by many factors, among which the system’s workload is wildly believed to have a major effect [1,2].

To study the aging phenomenon, researchers employed a design of experiment (DOE) approach to accelerate the aging process [3], where the speed of aging of web servers has been investigated with respect to page type, page size and connection rate. It was found that the aging process is obviously accelerated by the request rate. Grottke et al. [4] assumed an even workload when studying the aging phenomenon in a web server, and used an autoregression (AR) model to forecast system resource exhaustion. With a constant workload, the system resource exhaustion would only be affected by the internal aging process, and thus the AR model was sufficient for forecasting purposes. Unfortunately, however, the workload of a real-life systems is often uneven. For example, a web server is subject to a much larger connection rate in daytime than midnight. That is to say, a web server should have different levels of aging speed at different time periods. Vaidyanathan and Trivedi [5] noted the effect of workload on the speed of aging, and carried out an empirical study to build a software aging model involving workload. The workload was represented by a set of system parameters related to system activities. Those parameters were clustered into eight groups. Vaidyanathan and Trivedi calculated the exhaustion rate of computing resources with respect to each workload group. They found that the aging speed was faster with higher system activities. Their findings were only for modelling purposes rather than for forecasting the exhaustion of resources. Furthermore, their model was built based on the transition rate from one level of workload to another, and the transition rate was obtained using a statisti-
cal method by observing the real world. It should be noted that a real-world server’s workload transition is case-by-case, and therefore the above model was only used for offline analysis and could hardly be applied to forecasting available resources online.

Under varying workloads, an aging software system could exhibit abrupt or dramatic changes in its behaviour, which can hardly be processed and analyzed by the rich achievements in conventional linear models, although the workload of a system can often be monitored conveniently. A question naturally arises: can we incorporate the effect of workload into an aging forecasting model? This paper attempts to address this question.

The rest of this paper is organized as follows. Section 2 presents the motivations of this study. Section 3 reports an aging phenomenon observed in our datasets, and Section 4 introduces our approach. Section 5 illustrates our strategy where the workload is used as prior information to divide the aging process into various sections in such a way that each section will have a relatively unchanged workload. Also in this section, a threshold AR (TAR) model is employed to forecast the free memory of a software system, in which the workload is used as the threshold. Section 6 concludes this paper.

2. Motivations

During the two decades of software aging research, most studies have focused on aging forecasting and control, almost all researchers tended to explore the root origins and the evolvement of software aging. Some researchers noted the effect of workload on the speed of aging, for which they either built mathematical models or conducted experimental studies.

Tian et al. [7] analyzed the reliability of web servers. They employed workload and failure rate calculated from the web logs to evaluate the operational reliability. Bruneo et al. [8] built software aging models for a virtual machine monitor (VMM) in the cloud computing environment, aiming to find the optimal rejuvenation time for the cloud computing system. However, their findings were based on some assumptions about the mechanism and evolvement of aging, which were relatively arbitrary. Machida et al. [9] proposed a semi-Markov model to determine the optimal trigger time for software rejuvenation, and found that software rejuvenation could reduce the job completion time and/or extend the lifetime of the server system. Escheikh et al. [10] investigated a virtualized server system that was subject to software aging. Using stochastic reward nets, they obtained a trade-off between availability, power usage and power performance.

The root origins and influencing factors of software aging are still not well understood. Such understanding can only be obtained through experimental studies. The rationale behind almost all experimental studies is that aging is strongly related to available resources of the computer system [4,5]. Tian et al. [7] carried out a series of controlled experiments to build an aging-workload relationship with a focus on workload parameters. They were able to draw general conclusions based on a series of experimental studies using subject applications, but their conclusions were not for forecasting purposes. Grottke et al. [4] proposed a metric named “estimated time to exhaustion” to predict the approximate time of system resource depletion. Matias et al. [3] reported the aging process of an Apache web server, and found that the aging speed was strongly affected by the workload, file types and other factors. They further proposed an accelerated aging experimental methodology, which was later employed by Zhao et al. [11] to investigate the aging phenomenon of a web server. Vaidyanathan and Trivedi [5] studied the relationship between workload and aging speed, where the workload was defined as a vector representing the activities of the application server. This vector could be clustered to indicate different workload levels, and the aging speed was calculated with respect to each workload level. They found that the aging speed was closely related to the workload.

To sum up, it is clear that workload data can be used as prior information to forecast and/or control software aging. In the existing literature, there is a lack of study in forecasting system resource exhaustion under the real-world scenario of varying workloads. The present paper aims to address this question by proposing a framework that incorporates workload data.

Software aging control has two-fold purposes: (i) evaluate the current degree of aging or forecast resource exhaustion time, and (ii) employ some techniques to rejuvenate the aged software system. Both these two targets can be incorporated into a closed-loop control system. Hence, we propose a framework of aging control with workload information, as illustrated in Fig. 1.

Fig. 1 Framework of software aging control with workload information

In Fig. 1, the workload and system resources of the target system are collected and input into a module to estimate the aging degree. This module can estimate the
present degree of degraded performance and/or the resource exhaustion time. The output of the aging evaluation module is fed back to a comparing unit, where the difference between the estimated aging degree and the required minimum performance of the system (set by the administrator) is calculated to determine whether rejuvenation should be triggered. The rejuvenation trigger process can employ various techniques to clean the aged system, such as resetting it, restarting a module or garbage collection.

3. Datasets

3.1 Data collection

Our datasets come from controlled experiments rather than empirical experiments. We use the datasets provided by a team in Duke University led by Trivedi [4]. The data collection process is described as follows: a workstation is used as a client to generate artificial requests to a web server (Apache httpd 1.3), which is deployed on another workstation. Both workstations are connected via a switch. The system activities of the server are collected periodically. In order to expedite aging, the capacity of the web server under the experimental environment is estimated. The artificial request rate is slightly lower than the capacity of the web server. During the experiment, many parameters are collected via a switch. The system activities of the server are collected periodically. The output of the aging evaluation process is shown in Fig. 2.

In this paper, we adopt a dataset in which varying connection rates are employed. More specifically, `httpperf` sent 350 r/s to the Apache web server for about 400 h, then increases the connection rate to 360, lasting for 400 h, then increases the rate again to 390 and kept this rate for about 500 h. Note that this dataset was not previously analyzed by Trivedi’s team. The free physical memory (memFree) recorded in the dataset is shown in Fig. 2.

In Fig. 2, when the workload changes, the memFree increases to some extent, then decreases gradually, as is contradictory to our intuition. This can be interpreted by the working mechanism of the Linux operating system: when the physical memory is nearly exhausted, Linux will use its swap space (virtual memory) to swap out a lot of pages. Of course, using the swap space will result in performance degradation. Hence, memFree can be used as a leading metric for performance degradation. In addition, Fig. 2 shows that the memory leakage rises when the workload increases. Merely eyeballing is insufficient. We hence use a statistical method to calculate the memory leakage under each workload.

3.2 Preprocessing

In this research, the aging speed is defined as the memory leak speed. To improve the calculation accuracy of the aging speed, we exploit a non-parametric method developed by Sen [12] to estimate the slope of variables. This method is not affected by outliers, and is robust to missing data. This approach focuses on all pairs of data points \(y_k\) and \(y_l\) where \(k < l\). For each pair, the slope \(s_{ql} = (y_l - y_k)/(l-k)\) is calculated. Sen’s slope estimate is defined as the median of \(n(n-1)/2\) obtained \(s_{ql}\)’s. This gives an aging speed of 2 013 kB/s under 390 r/s, 1 924 kB/s under 360 r/s and 1 803 kB/s under 350 r/s. The results tell us that the aging speed increases with a higher workload.

Obviously, larger workloads will speed up the aging process. There is a mode changing at the 2 501th point. Thus, we cannot apply a uniform time series model all through this process. The data are obviously unstable. Before using our approach, the data should be first differentiated to obtain a stable new time series. The differentiation process is expressed as follows:

\[ y_t = \nabla z_t = z_t - z_{t-1} \quad (1) \]

where \(z_t\) refers to the time series shown in Fig. 2, and \(y_t\) refers to differentiated time series, which is shown in Fig. 3.

Fig. 3 shows that the differentiated time series roughly fluctuates around a constant value. It is detrended and thus becomes stable. Next, we should determine whether an AR model is appropriate for all sections of the time series. This can be determined by autocorrelation and paracorrelation of each section. More specifically, if the calculated autocorrelation decreases exponentially with \(t\), and paracorrelation cuts off after \(k\) steps, the time series can be treated as stable and can be described by the AR model [13]. The autocorrelation of a time series \(y\) can be calculated as follows:

\[ r_k = \frac{c_k}{c_0} \]
\[ c_k = \frac{1}{N} \sum_{t=1}^{N-k} (y_t - \bar{y})(y_{t+k} - \bar{y}) \]  
\[ \phi_{kk} = \text{corr}[\hat{z}_t - \hat{\hat{z}}_t, z_{t-k} - \hat{\hat{z}}_{t-k}] \]

where \( \bar{y} \) refers to the mean of \( y \). The paracorrelation can be calculated as follows:

\[ y_t = \phi_0 + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t \]

where \( \varepsilon_t \) is a sample from white noises with the mean zero, and \( \phi_0, \phi_1, \phi_2, \ldots, \phi_p \) can be estimated by least-squares or similar error control criteria. The AR model is used to interpret the stable and linear time series. The steps of using the AR model to formulate a time series are described as follows.

(i) Determine if the AR model is appropriate for the target time series.

(ii) Determine the order \( p \).

(iii) Estimate \( \phi_0, \phi_1, \phi_2, \ldots, \phi_p \) via an error control criterion.

(iv) Make \( n \)-step ahead forecasting with the obtained recurrent formula.

\[ y_t = \phi_0 + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t \]

where \( \varepsilon_t \) is interpreted by \( p \) previous observations of that time series \( \varepsilon_t \) refers to the order of AR model) An AR model constructs a recurrent relation between the current observation and previous values, and can be used for forecasting purposes.

4.2 TAR

The philosophy behind TAR is that the time series \( y_t \) can be divided into \( l \) sections by \( l-1 \) thresholds. Then, \( y_{t-d} \) can be assigned to various sections. Finally, \( x_t \) in each section will be modelled by AR. It can be formulated as

\[ \begin{aligned}
\left\{ y_t &= \phi_0^j + \sum_{i=1}^{p_j} \phi_i^j y_{t-i} + \varepsilon_i^j, j = 1, 2, \ldots, l-1 \\
r_{j-1} &< w_{t-d} \leq r_j
\end{aligned} \]

where \( \varepsilon_i^j (j = 1, 2, \ldots, l-1) \) refers to \( l \) independent normal white noise time serieses; \( d \) refers to the delay (a positive integer); \( r_j (j = 1, 2, \ldots, l-1) \) refers to the thresholds; \( l \) refers to the number of thresholds; \( \phi_i^j \) refers to the coefficients of the \( j \)th AR model; \( p_j \) is the order of the \( j \)th AR model. \( w_{t-d} \) refers to the threshold, which can be \( y_t \) itself or other time series. In our case, \( w \) refers to the workload of Apache. In this way, the memFree \( y_t \) can be interpreted by different forms of the AR model under different workloads.

The above description means that TAR is a type of AR model which has a different form in each section. When modelling a time series using TAR, we should seek optimal parameter settings in our model, i.e., \( d, l, r_j, p_j, \phi_i^j \). The steps of using TAR to model a nonlinear time series are described as follows.

(i) Calculate the delay \( d \) in the time series.

(ii) Divide the time series into \( l \) sections, and assign each \( y_t \) into each section.
(iii) Determine the order $p$ in each AR model in the $l$ sections.

(iv) Estimate $\phi_0, \phi_1, \phi_2, \ldots, \phi_p$ for each of the $l$ AR models via an error control criterion.

(v) Make $n$-step ahead forecasting with the obtained recurrent formula.

5. Experimental results

5.1 Offline analysis

In this section, we follow the steps described in Section 4 to build a TAR model for the time series shown in Fig. 2. Recall that there are three levels of workload employed in the experiments, i.e., 350 r/s, 360 r/s and 390 r/s. The workload transition point can divide the time series into three parts. Obviously, each part should be described by a different AR model. This can be solved by TAR with the prior information of the workload.

In (5), the time series is divided by $w$ which refers to the workload in our case. The parameter $d$ refers to the delay which reflects the inertia of that system generating this time series. More specifically, it reflects how fast memFree is affected by the workload. In Fig. 2, we can see that aging speed will rise once the workload rises. Since the inertia of a computer system is very little, $d$ can be set to 1 ($d$ must be a positive integer in the TAR mathematical model). This can be validated by the observations shown in Fig. 2. The observation interval is 10 min, and the httpperf can increase the connection rate from 350 to 360 within 1 s. Since the workload changes twice, the time series memFree can be divided into three sections, i.e., it can be described by three AR models.

Next, we should determine the order $p$ of each AR model. Usually, the error between real observations and the output of the AR model will decrease with $p$. If we choose the model that fits the data with the lowest estimated error variance at least squares estimation, an overfitting problem may occur, which will prejudice the generalization of our model. Thus the order $p$ will be determined by a tradeoff between error and generalizability. There are some selection criteria, such as Akaike information criterion (AIC) or Bayesian information criterion (BIC), which can determine the optimal order $p$. These criteria can penalize the number of freely chosen parameters, i.e., the order $p$, in the AR model. In this paper, the order $p$ is determined by the AIC criterion, which takes a form shown as follows:

$$AIC(p) = \ln \hat{\sigma}^2_a + \frac{2p}{n}$$

where $\hat{\sigma}^2_a$ refers to the error variance calculated in least squares estimation, $p$ refers to the order of the AR model, and $n$ refers to the number of points in the time series. In our case, we get the order value 2 for each AR model in the three sections.

In the following, the least-squares method is employed to estimate the parameters $\phi_0, \phi_1$ and $\phi_2$. To validate the effectiveness of our approach, we employ the first 6 400 observations to estimate our model and forecast the remaining 3 200 observations. The data used for estimation include observations under each workload. The resulting models for physical memory (denoted by memFree) and the used swap (denoted by usedSwap) are shown in Figs. 4 and 5.

Figs. 4 and 5 show that our model can fit the observations quite accurately. To quantitatively evaluate the accuracy, we calculate the error between the output of our model and observations. In this paper, the following well-known accuracy indicators are used:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

where $y_i$ and $\hat{y}_i$ are the real and predicted observations, respectively, and $n$ is the number of observations.
JIA Yunfei et al.: A workload-based nonlinear approach for predicting available computing resources

5.2 Online predicting

In a real-word server system, the workload is varying with time. Hence, our approach should dynamically construct the TAR model to accurately predict the exhaustion time. The steps of applying our approach to predict resource exhaustion are described as follows:

Step 1 Construct an AR model for the current workload section.

Step 2 Make n-step ahead forecasting, and estimate the exhaustion time.

Step 3 If the workload changes, construct another AR model for the current workload and go to Step 2.

Following the above steps, we can obtain a series of exhaustion times $T_j$, in which $j$ refers to the number of workload changes during the lifetime of the server system. We follow the above steps to estimate the exhaustion time shown in Figs. 4 and 5. The server system has a 16 GB physical memory and a 32 GB swap space. The experimental results are shown in Table 2.

Table 2 Exhaustion time under each workload

| Workload/(req/s) | Exhaustion time for memory/min | Exhaustion time for swap/min |
|------------------|--------------------------------|------------------------------|
| 350              | 16 500                         | 120 000                      |
| 360              | 8 900                          | 65 000                       |
| 390              | 3 700                          | 45 000                       |

6. Conclusions

In this paper, the nonlinear aging phenomenon observed from a web server with varying workloads, has been analysed. There is an obvious relationship between the aging speed and the system’s workload. The aging speed is calculated by a robust algorithm to validate the trend of resource leak. To address the forecasting problems caused by the unstable and nonlinear time series, we have employed the prior information of workload, which has allowed us to accurately estimate the aging slope and to further forecast the system resource exhaustion time.

The workload is used as a threshold to tell us which model can be used when forecasting system resource exhaustion. Then, the evolvement of the system resources is described by a TAR model. The accuracy of our approach is compared with that of the AR model to show the advantage of our approach. Our approach can be easily implemented and can be applied to any service-oriented applications.

Acknowledgements

We would like to thank Kishor S. Trivedi and Michael Grottke for providing the data sets for this research.

References

[1] BRUNEO D, DISTEFANO S, LONGO F, et al. Workload-based software rejuvenation in cloud systems. IEEE Trans. on Computers, 2013, 62(6): 1072–1085.
[2] YAN Y Q, GUO P, CHENG B, et al. An experimental case study on the relationship between workload and resource consumption in a commercial web server. Journal of Computational Science, 2018, 25: 183–192.
[3] MATIAS R, BARBETTA P A, TRIVEDI K S, et al. Accelerated degradation tests applied to software aging experiments. IEEE Trans. on Reliability, 2009, 59(1): 102–114.
[4] GROTTKE M, LI L, VAIDYANATHAN K, et al. Analysis of software aging in a web server. IEEE Trans. on Reliability, 2006, 55(3): 411–420.
[5] VAIDYANATHAN A, TRIVEDI K S. A comprehensive model for software rejuvenation. IEEE Trans. on Dependable and Secure Computing, 2005, 2(2): 124–137.
[6] CONTRONEO D, NATELLA R, PIETRANTUONO R, et al. A survey of software aging and rejuvenation studies. ACM Journal on Emerging Technologies in Computing Systems, 2014, 10(1): 1–34.
[7] TIANT J, RUDRARA JU S, LI Z. Evaluating web software reliability based on workload and failure data extracted from server logs. IEEE Trans. on Software Engineering, 2004, 30(11): 754–769.
[8] FUMIO M, XIANG J W, TADANO K, et al. Lifetime extension of software execution subject to aging. IEEE Trans. on
Reliability, 2017, 66(1): 123 – 134.

[9] MOHAMED E, ZAYNEB T, KAMEL B. Workload-dependent software aging impact on performance and energy consumption in server virtualized systems. Proc. of the IEEE International Symposium on Software Reliability Engineering, 2016: 111 – 118.

[10] ZHAO J, WANG Y B, NING G R, et al. A comprehensive approach to optimal software rejuvenation. Performance Evaluation, 2013, 70(11): 917 – 933.

[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] BOX G E P, JENKINS G M, REINSHEL G C. Time series analysis: forecasting and control. Hoboken: Prentice-Hall Inc., 1994.

[13] DAVIDSON J. Establishing conditions for the functional central limit theorem in nonlinear and semiparametric time series processes. Journal of Econometrics, 2002, 106(2): 243 – 269.

Biographies

**JIA Yunfei** was born in 1979. He is currently an associate professor at Civil Aviation University of China. He received his B.E. (2001) and M.S. (2004) degrees from Hebei University of Technology, and completed a Ph.D. in software testing at Beihang University in 2010. His research interests include software testing and software reliability modelling.
E-mail: yfjia@cauc.edu.cn

**ZHOU Zhiquan** was born in 1976. He received his B.S. degree in computer science from Peking University, China, and the Ph.D. degree in software engineering from The University of Hong Kong. He is currently an associate professor at the University of Wollongong, Australia. His current research interest is in software testing and analysis.
E-mail: zhiquan@uow.edu.au

**WU Renbiao** was born in 1966. He received his M.S. degree from Northwestern Polytechnical University and completed his Ph.D. degree in signal processing at Xidian University. Currently, he is a professor at Civil Aviation University of China. His interests are signal processing and image processing.
E-mail: rbwu@cauc.edu.cn