http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 7 (1): 98-104, 2008
ISSN 1812-5638

A Study on Software Rejuvenation Model of Application Server Cluster in Two-Dimension State Space Using Markov Process

[1]Qi Yong, [1]Meng Haining, [1]Hou Di and [2]Chen Ying
[1]School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[2]IBM China Research Laboratory, Beijing 100094, China

Abstract: Software rejuvenation is a main method for counteracting software aging, when it is performed in cluster system, system reliability can be significantly increased. We firstly presented a software rejuvenation policy of application sever cluster. Then we set up a software rejuvenation model in two-dimension state space for a two-node application server cluster and give the quantitative analysis using continuous time Markov process. Finally, the numerical experiment results show that the selected optimal software rejuvenation interval can greatly reduce expected systematic downtime and improve software reliability of application server cluster.

Key words: Software rejuvenation, continuous time Markov process, application server cluster, software reliability

INTRODUCTION

Recent studies have reported the phenomenon of software aging, Parnas and Lorge (1994) and Huang *et al.* (1995), in which the state of system performance degrades with time. The primary symptoms of this degradation include exhaustion of resources, data corruption and instantaneous error accumulation. Software aging has not only been observed in software used on a mass scale but also in specialized software used in high-availability and safety-critical applications. In order to enhance system reliability and prevent systematic degradation or crash, Huang *et al.* (1995) introduced such a preventive fault-tolerant technique called software rejuvenation. It involves occasionally stopping the running software, cleaning its internal state and then restarting.

The fault-tolerant technique enables significantly increased system availability and reduced downtime cost. However, as this reactive action is taken after the failure has occurred, it usually involves considerable system maintenance cost and potential financial losses. Software rejuvenation as a proactive fault management has proved to be an effective complementary approach to traditional reactive recovery. It can take appropriate actions to reduce both system unavailability and downtime cost in a cost-effective way before the system experiences failures. At present, software rejuvenation is studied extensively and applied to many high reliability and availability application in literature (Avritzer and Weyuker, 1997; Garg *et al.*, 1998; Dohi *et al.*, 2000; Vaidyanathan and Trivedi, 2005; Matias and Filho, 2006).

Cluster computing has been widely used in various kinds of software systems. Application server cluster is a collection of independent and self-contained application server. Compared with single application server, application server cluster has characteristic of better extendibility, reliability and fault-tolerant. However, many application server clusters are not tolerant to system failures and lead to loss of services. By combining software rejuvenation with clustering computing, the cluster system availability and performance can be achieved. More recently, two kinds of software rejuvenation policies have been implemented in cluster system to improve performance and availability by taking advantage of the failover feature. In the model-based policy, the cluster nodes are rejuvenated in a rollback way after every determined interval. In the measurement-based policy, the rejuvenation interval is estimated based on the collection and statistical analysis of system data.

Since rejuvenation of operational software systems incurs some overhead, it is important to build software rejuvenation model and determine when and how often software rejuvenation should be initiated. Analytical modeling has been used to address this issue in several research papers. Huang *et al.* (1995) introduced the continuous Markov process to build two-phase software rejuvenation model that includes healthy state, aging probable state, system failure state and rejuvenation state. However, this model aims at generic system and is hardly adapted to research on specific system. By Markov decision process, Pfening *et al.* (1996) proposed a software rejuvenation frame and applied it to AT and T

Corresponding Author: Meng Haining, School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China

communication system. Garg *et al.* (1998) constructed rejuvenation model of transaction processing system based on queuing theory. Dohi *et al.* (2000) set up software rejuvenation model of client/server system and adopted non-parameter statistic analysis to estimate optimal software rejuvenation interval. For cluster system, Garg *et al.* (1998) and Wei *et al.* (2004) presented stochastic Petri net approach to analyze software rejuvenation. Vaidyanathan *et al.* (2001) used stochastic Reward Net to model and analyze cluster system that employed software rejuvenation. Bao *et al.* (2005) and Vaidyanathan and Trivedi (2005) took the system workload into account for building a model to estimate resource exhaustion times.

We extend the model in Garg *et al.* (1998) and set up a software rejuvenation model for two-node application server cluster. From the mathematic modeling point of view, runtime state and operation behavior of application sever cluster are captured and software rejuvenation model is built in two-dimension state by using continuous time Markov process. In order to improve systematic reliability of application server cluster, the systematic availability formula and optimal rejuvenation interval is derived. Finally, the numerical results are given to validate the proposed model.

BASIC CONCEPT

Firstly some fundamental concepts are given and it will be useful for analyzing software rejuvenation model of application server cluster.

Definition 1: Let p_{ij} (t) be transition probability function of continuous-time Markov process and q_{ij} be transition rate. Kolmogorov forward equation is defined as follows:

$$\frac{dp_{ij}(t)}{dt} = \sum_{k=0}^{N} p_{ik}(t)q_{kj}, i, j = 0, 1, 2 \quad (1)$$

Let P (t) be the matrix of transition probability function p_{ij} (t) (i, j = 0, 1, 2, ...) and Q be the matrix of transition rate function q_{ij} (i, j = 0, 1, 2, ...), formula (1) can be expressed in matrix format as follows:

$$P'(t) = P(t)Q \quad (2)$$

Definition 2: Random variables X, Y, which denote the time to failure of application server A and B, respectively, have the following negative-exponential distribution with parameter λ_1, λ_2, respectively:

$$F_X(x) = 1 - e^{-\lambda_1 x}, F_Y(y) = 1 - e^{-\lambda_2 y}, x, y, \lambda_1, \lambda_2 > 0 \quad (3)$$

If the failure occurrences for application servers A and B are independent to each other, the joint probability distribution function of failure occurrence for two application servers is as follows:

$$\begin{aligned} F_{XY}(x,y) &= P\{X \le x, Y \le y\} \\ &= 1 - \{1 - F_X(x)\}\{1 - F_Y(y)\} \\ &= 1 - e^{-\lambda_1 x - \lambda_2 y} \end{aligned} \quad (4)$$

SOFTWARE REJUVENATION POLICY FOR APPLICATION SERVER CLUSTER

Application server cluster is a collection of independent and self-contained application server to provide a reliable and powerful system. Figure 1 shows the architecture of a J2EE application server cluster. The load dispatcher is responsible for receiving client requests

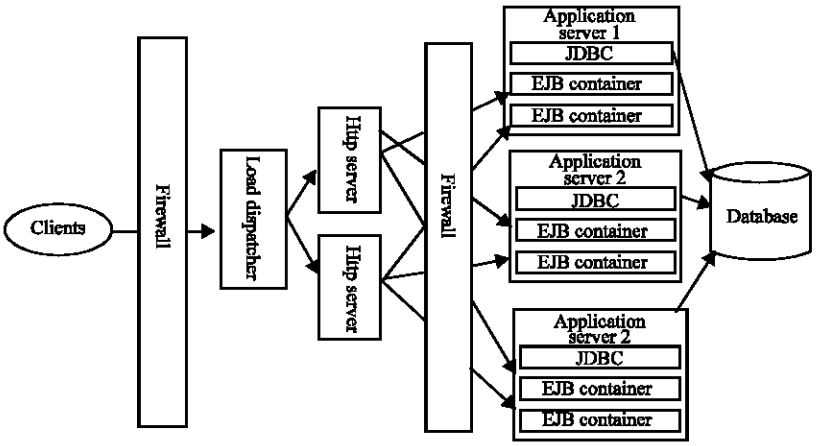

Fig. 1: Architecture of application server cluster

and assigning tasks to the lighter load of http server. Service requests are allocated to different application servers by the load balance theory. The application server connects and queries database server, then returns results to clients.

We introduce below the software rejuvenation policy for application server cluster. Firstly, the analytical rejuvenation model of application server cluster is set up and the probability distribution of state transition is evaluated. Then optimal software rejuvenation interval is derived from the analytical rejuvenation model. When the time to perform software rejuvenation of an application server node is triggered, the application server node backups its runtime states and the load dispatcher migrates the task to another node. Subsequently, the application server node releases its resources, cleaning its internal state and recovery to initialized healthy state. In addition, it should be noticed that when the total load of the application server node is heavy, software rejuvenation might incur overhead and further deteriorate the cluster's performance. Thus, the rejuvenation should be carried out while application server node with lightest load.

SOFTWARE REJUVENATION MODEL OF APPLICATION SERVER CLUSTER

Software rejuvenation model of single application server: For generic software system, Garg *et al.* (1998) defined the two-stage software rejuvenation model, based on which we firstly set up the software rejuvenation model for single application server, as is shown in Fig. 2. The system has three states: working state 0 (denoted as u), failure state 1 (denoted as d) and rejuvenation state 2 (denoted as r). In the beginning, the application server stays in the working state 0. With system performance degrades over time, a failure may occur. If system failure occurs before triggering software rejuvenation, the application server changes from the working state 0 to system failure state 1 and then the system recovery operation is started immediately. Otherwise, the application server changes from the working state 0 to the software rejuvenation state 2 and later the software rejuvenation is carried out. After completing the system repair or rejuvenation, the application server becomes as good as new and changes to the beginning working state 0 again. We define the time interval from the beginning of the system working to the next one as one cycle.

According to the model described above, at any time t the application server can be in any one of three states: up and available for service (working state 0), recovering from a failure (failure state 1), or undergoing software

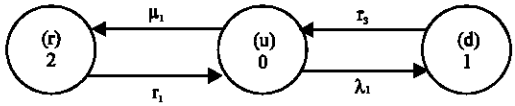

Fig. 2: Software rejuvenation model of single application server

rejuvenation (rejuvenation state 2). To formally describe the software rejuvenation model of single application server, continuous time Markov process denoted as $Z = (Z_t; t \geq 0)$ is used, where Z_t represents the state of application server at time t. The transition probability function of Z is expressed as follows:

$$p_{ij}(t) = P(Z_t = j \mid Z_0 = i)(\forall i, j \in \Omega, t \geq 0) \quad (5)$$

Where, $\Omega = \{0, 1, 2\}$ is the state space set.

For the software rejuvenation model in Fig. 2, λ_1, i_1, r_1 and r_3 represents the failure rates from system working state to failure state, the transition rate to trigger software rejuvenation, the rejuvenation rate from software rejuvenation state to system working state and the recovery rate from system failure state to system working state, respectively. Let Q be the matrix of the transition rate function. According to the state transition relationship of single application server, the transition rate matrix for the continuous time Markov process Z can be easily derived as:

$$Q = \begin{pmatrix} -(\mu_1 + \lambda_1) & \lambda_1 & \mu_1 \\ r_3 & -r_3 & 0 \\ r_1 & 0 & -r_1 \end{pmatrix} \quad (6)$$

Let P (t) be the matrix of transition probability function $p_{ij}(t)$ $(\forall I, j \epsilon \Omega)$. According to Kolmogorov forward Eq. 1, transition probability matrix P (t) satisfies:

$$\begin{aligned} P'(t) &= P(t)Q \\ P(0) &= I \end{aligned} \quad (7)$$

Where, I is the unit matrix.

Let p_j, $j \epsilon \Omega$ be the instantaneous steady probability of single application server in state j. According to the limit distribution theorem, p_j, $j \epsilon \Omega$ is given by:

$$p_j = \lim_{t \to \infty} p_{ij}(t)(\forall i, j \in \Omega) \quad (8)$$

Substitute Eq. 6 and 8 to Eq. 7, the following equation is derived:

$$\begin{aligned} &-(\mu_1+\lambda_1)p_0 + r_3p_1 + r_1p_2 = 0\\ &-r_3p_1 + \lambda_1p_0 = 0\\ &-r_1p_2 + \mu_1p_0 = 0\\ &\sum_{i=0}^{2} p_i = 1 \end{aligned} \tag{9}$$

Where p_i, i = 0, 1, 2 can be obtained by solving the Eq. 9. The application server is available for service requests in working state 0 and application server is unavailable for rejuvenation state 1 and failure state 2, Thereafter, the system availability for single application server is given by:

$$P_{A1} = P_0 \tag{10}$$

Software rejuvenation model of two-node application server cluster: We extend the software rejuvenation model of single application server to two-dimension state space, then derive software rejuvenation model of two-node application server cluster as shown in Fig. 3.

The states of application server cluster are denoted by a 2-tuple S, which is formally defined as: S = { (i, j) | i, j∈{u, d, r}}, where i is the state of the first application server and j is the state of the second application server. For the first applications sever, λ_1, i_1, r_1 and r_3 represents the failure rate from system working state to failure state, the transition rate to trigger software rejuvenation, the rejuvenation rate from the:

$$Q = \begin{pmatrix} -(\mu_1+\mu_2+\lambda_1+\lambda_2+\lambda_3) & \lambda_1 & \lambda_2 & \lambda_3 & u_1 & \mu_2 & 0 & 0\\ r_3 & -(r_3+\lambda_2) & 0 & \lambda_2 & 0 & 0 & 0 & 0\\ r_4 & 0 & -(r_4+\lambda_1) & \lambda_1 & 0 & 0 & 0 & 0\\ 0 & r_4 & r_3 & -(r_3+r_4) & 0 & 0 & 0 & 0\\ r_1 & 0 & 0 & 0 & -(r_1+\lambda_2) & 0 & \lambda_2 & 0\\ r_2 & 0 & 0 & 0 & 0 & -(r_2+\lambda_1) & 0 & \lambda_1\\ 0 & 0 & r_1 & 0 & r_4 & 0 & -(r_1+r_4) & 0\\ 0 & r_2 & 0 & 0 & 0 & r_3 & 0 & -(r_2+r_3) \end{pmatrix} \tag{11}$$

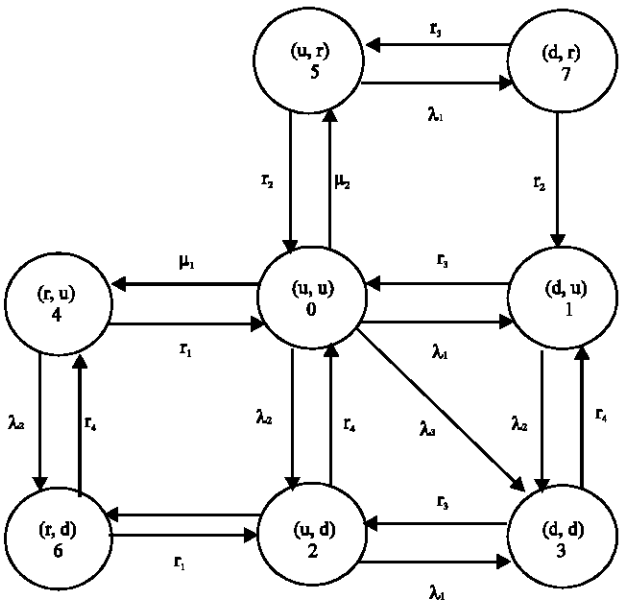

Fig. 3: Software rejuvenation model of two-node application server cluster

software rejuvenation state to working state and the recovery rate from system failure state to working state, respectively. Correspondingly, for the second application sever, λ_2, μ_2, r_2 and r_4 denotes the failure rate, the transition rate to trigger software rejuvenation, the rejuvenation rate and the recovery rate, respectively. λ_3 is the joint probability distribution function, which denotes the failure rate from system working state to failure state for both application servers. Note that software rejuvenation is not allowed for both application servers to be carried out concurrently.

It is assumed that Z_t is the state of the cluster at time t, $\Omega' = \{0, 1, 2...7\}$ is the state space set. Similarly, we use continuous time Markov process, denoted as $Z = (Z_t; t \geq 0)$, to describe the software rejuvenation model of two-node application server cluster. The transition probability function of Z is expressed as Eq. 11 and p_j, $j \in \Omega$ is given by:

$$p_j = \lim_{x \to \infty} p_{ij}(t) \ (\forall i, j \in \Omega') \tag{12}$$

Correspondingly, the transition probability matrix P (t) also satisfies the conditions in Eq. 7. As already used in previous section, substitute Eq. 11 and 12 to Eq. 7, the following equations can be derived:

$$\begin{aligned}
&-(\mu_1 + \mu_2 + \lambda_1 + \lambda_2 + \lambda_3)p_0 + r_3p_1 + r_4p_2 + r_1p_4 + r_2p_5 = 0\\
&-(r_3 + \lambda_2)p_1 + \lambda_1p_0 + r_4p_3 + r_2p_7 = 0\\
&-(r_4 + \lambda_1)p_2 + \lambda_2p_0 + r_3p_3 + r_1p_6 = 0\\
&-(r_3 + r_4)p_3 + \lambda_3p_0 + \lambda_2p_1 + \lambda_1p_2 = 0\\
&-(r_1 + \lambda_2)p_4 + u_1p_0 + r_4p_6 = 0\\
&-(r_2 + \lambda_1)p_5 + \mu_2p_0 + r_3p_7 = 0\\
&-(r_1 + r_4)p_6 + \lambda_2p_4 = 0\\
&-(r_2 + r_3)p_7 + \lambda_1p_5 = 0\\
&\sum_{i=0}^{7} p_i = 1
\end{aligned} \tag{13}$$

By solving the above equations, we can obtain the value of p_i, i = 0, 1, 2...7. According to the rejuvenation model in Fig. 3, the application server cluster is unavailable in the state of (d, d), (r, d) and (d, r). Thereafter, the availability of two-node application server cluster is given by:

$$\begin{aligned}
P_{A2} &= 1 - p_3 - p_6 - p_7\\
&= p_0 + p_1 + p_2 + p_4 + p_5
\end{aligned} \tag{14}$$

NUMERICAL RESULTS AND ANALYSIS

To acquire reliability measure of application server cluster, we perform numerical experiments by taking system unavailability as evaluation indicator. Meanwhile, the relationship between software rejuvenation interval and system unavailability is illustrated to investigate the affect of software rejuvenation on system reliability of application server cluster.

Table 1: Parameter value (unit: h^{-1})

r_1	r_2	r_3	r_4
1	1	0.1	0.1

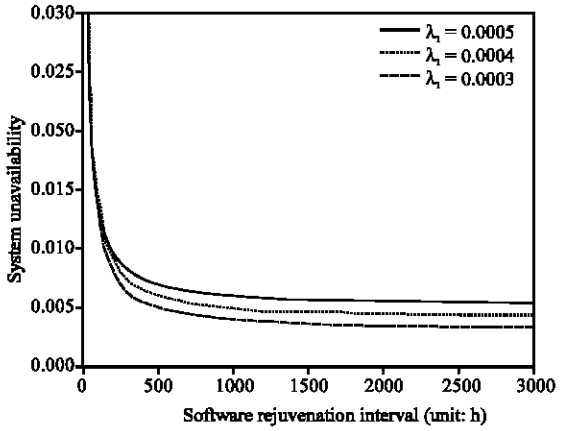

Fig. 4: The system unavailability versus software rejuvenation interval for single application server

According to Eq. 10, the unavailability of single application server P_{U1} can be evaluated as follows:

$$P_{U1} = 1 - P_{A1} = p_1 + p_2 \tag{15}$$

By Eq. 14, system unavailability of two-node application server cluster is calculated using the following formula:

$$P_{U2} = 1 - P_{A2} = p_3 + p_6 + p_7 \tag{16}$$

The system parameter default values in software rejuvenation model are given in Table 1 (unit: h^{-1}), in which the rejuvenation rate is 1 and the recovery rate is 0.1. All the parameter values are selected by experimental experience for demonstration purposes.

Figure 4 shows the system unavailability of single application server versus rejuvenation interval with varying failure rate λ_1. From the figure we can see that rejuvenation interval strongly influences system reliability. When the rejuvenation interval $1/\mu_1$ is very small, the single application server is almost unavailable. With the rejuvenation interval increasing, the system unavailability reduces rapidly and goes to a steady value, namely, the value of optimum rejuvenation interval. It also can be concluded that for certain rejuvenation interval is

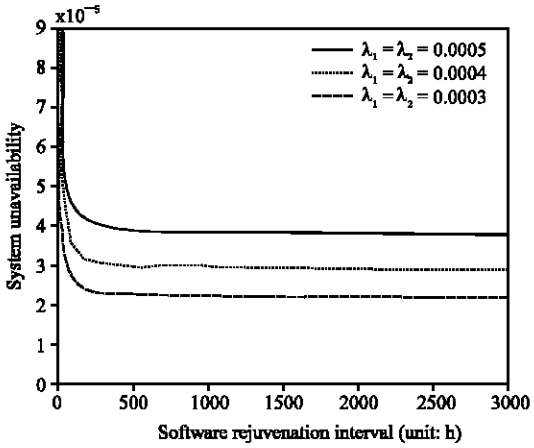

Fig. 5: System unavailability versus software rejuvenation interval for two-node application server cluster

given, if the failure rate becomes larger, the application server tends to be less reliable, the optimal rejuvenation interval can be taken rather smaller value, namely, software rejuvenation should be carried out more and more frequently.

For the two-node application server cluster, Fig. 5 illustrates the system unavailability versus rejuvenation interval with varying failure rate. For simplify the numerical experiment, we assume the failure rate of both application server node $\lambda_1 = \lambda_2$, the joint failure rate λ_3 is 2. 5×10^{-7} h^{-1} and the rejuvenation rate satisfies $\mu_1 = \mu_2$. As can be observed that the curve of system unavailability has similar behavior with that of single application server, but the unavailability value is less than that of single application server. The result reveals that two-node application server cluster has higher reliability than single application server node. It also indicates that the optimal rejuvenation interval can be identified when the minimum value of system unavailability is achieved. The systematic unavailability and corresponding optimal software rejuvenation interval $1/\mu_1^*$ for different failure rate is shown in Table 2.

Finally, we examine the influence of joint failure rate λ_3 on the system reliability of two-node application server cluster. Figure 6 shows the system unavailability under different rejuvenation interval. The failure rate for single application server node is set to 0.0005 h^{-1}, the rejuvenation rates for two application servers are equal. While $\lambda_3 = 0$, it implies that two application servers are completely independent in terms of system failure occurrence, namely, the two nodes will not fail at the same time. While λ_3 increases, the correlation of two application servers becomes stronger. It can be observed that the greater the value of λ_3 is, the higher the system unavailability becomes. It indicates that the correlation parameter λ_3 play a significant role to attain the target reliability level.

Table 2: Optimal rejuvenation interval and corresponding system unavailability for two-node application server cluster

	$\lambda_1 = \lambda_2$		
	0.0003	0.0004	0.0005
$1/\mu_1$ (h)	1200	1400	1700
P_{UP2}	2.175	2.877	3.775

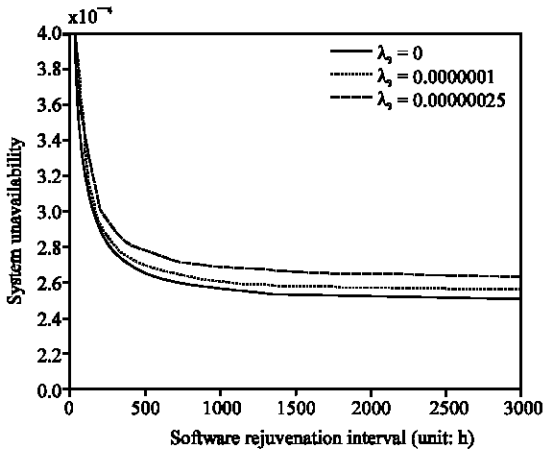

Fig. 6: System unavailability versus software rejuvenation interval for two-node application server cluster under different λ_3

CONCLUSION AND FUTURE WORK

We presented a software rejuvenation policy and set up the software rejuvenation model in two-dimension state space for two-node application server cluster. Based on the organization structure and running state of application server cluster, the system availability formula is derived using continuous time Markov process. The numerical results show that selected optimal software rejuvenation interval, which minimizes the system unavailability, can greatly reduce expected systematic downtime and improve systematic reliability of application server cluster.

Future work includes researches on fine-granularity software rejuvenation model for multi-node application server cluster and the software aging forecasting model considering the organization mechanism and runtime states of application server cluster.

ACKNOWLEDGMENTS

The author would like to thank the sponsors of the National Natural Science Foundation of China under Grant No. 60473098, Doctoral Fund No. 20060698018 and IBM China Research Laboratory Joint Project.

REFERENCES

Avritzer, A. and E.J. Weyuker, 1997. A monitoring smoothly degradating systems for increased dependability. Empirical Software Eng. J., 2 (1): 59-77.

Bao, Y.J., X.B. Sun and K.S. Trivedi, 2005. A workload-based analysis of software aging and rejuvenation. IEEE Trans. Reliabil., 54 (3): 541-548.

Dohi, T., P.K. Goseva and K.S. Trivedi, 2000. Statistical non-parametric algorithms to estimate the optimal software rejuvenation schedule. In: Pacific Rim International Symposium on Dependable Computing, pp: 77-85.

Garg, S., A. Puliafito, M. Telek and K.S. Trivedi, 1998. Analysis of preventive maintenance in transactions based software systems. IEEE Trans. Comput., 47 (1): 96-107.

Huang, Y., C. Kintala, N. Kolettis and N. Fulton, 1995. Software rejuvenation: Analysis, module and applications. In: IEEE International Symposium on Fault Tolerant Computing, pp: 381-390.

Matias, Jr. R. and J.F. Filho, 2006. An experimental study on software aging and rejuvenation in Web servers. In: Proceedings-30th Annual International Computer Software and Applications Conference, pp: 189-196.

Parnas and D. Lorge, 1994. Software aging. In: International Conference on Software Engineering, pp: 279-287.

Pfening, A., S. Garg, A. Puliafito, M. Telek and K.S. Trivedi, 1996. Optimal software rejuvenation for toleration software failures. Performance Evaluation, pp: 27-28.

Vaidyanathan, K., R.E. Harper, S.W. Hunter and K.S. Trivedi, 2001. Analysis and implementation of software rejuvenation in cluster systems. ACM SIGMETRICS Performance, 29 (1): 62-71.

Vaidyanathan, K. and K.S. Trivedi, 2005. A comprehensive model for software rejuvenation. IEEE Transaction on Dependable and Secure Computing, 2 (2): 124-137.

Wei, X., H. Yiguang and K.S. Trivedi, 2004. Software rejuvenation policies for cluster systems under varying workload. In: Proceedings of the 10th IEEE Pacific Rim International Symposium on Dependable Computing, pp: 122-129.