BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **19**, No 4
Sofia • 2019 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2019-0038

Availability Modelling of Cluster-Based System with Software Aging and Optional Rejuvenation Policy

Richa Sharma[1], *Gireesh Kumar*[2]

[1]*Department of Mathematics, JK Lakshmipat University, Jaipur 302026, India*
[2]*Department of Computer Science and Engineering, JK Lakshmipat University, Jaipur 302026, India*
E-mails: aligarh.richa@gmail.com gireesh8@gmail.com

Abstract: *This investigation deals with modeling and availability analysis of cluster-based system inflicted with software aging. Software aging is a phenomenon in which a software system shows performance degradation with time and finally results in software failures. To cope up with this phenomenon, rejuvenation is an innovative concept to recover from software failures. As failures occur, server has the option either to take essential rejuvenation with probability* ***p*** *or may opt for optional rejuvenation with complementary probability* ***q****. To achieve high availability of the system, the concept of clustering is also taken into consideration. In this study, restart, reboot and standby concept is used for reducing the downtime cost. The sensitivity analysis of different parameters on system availability has been examined numerically. By integrating clustering, software aging and rejuvenation, the researchers intended to increase the availability and decrease the down time.*

Keywords: *Software aging, Cluster, Standby, Reboot, Optional rejuvenation, Availability, Downtime cost.*

1. Introduction

In the current era of machines, human dependency on machining system has been increasing day by day. Moreover, availability of service is a major concern of today's business environment. Availability is defined as the probability that a system is operational and capable to provide the required services [1]. To improve the availability of services of cluster-based system, an advanced concept as software rejuvenation for recovering from failures is carried out. Clustered system has been analysed with rejuvenation under varying workload. To avoid unexpected closure of individual knots; standard, delayed and mixed rejuvenation was applied in this study [2].

Software aging is caused by resource depletion and can lead to gradual degradation of performance or blockage [3]. Software aging [4] phenomenon refers to all software tendency to fail or the accretion of errors occurring in continuously running operational software system which leads to performance degradation and eventually hang on the software system. Different causes of software aging are such

as languidness of operating system resources, storage fragmentation, and accumulation of errors, etc. Error accumulation usually takes the form of bad resource management that leads to resource exhaustion, such as memory leaks, unterminated threads. The accumulation of errors usually takes the form of a poor resource management, which leads to a depletion of resources, such as memory leaks, data corruption, etc. Therefore, some rejuvenation policies such as application restart, rebooting and concept of standby are required. Software rejuvenation is a cost-effective approach to eliminate the effects of aging and avoid aging-related failures [5].

Software rejuvenation is an essential proactive fault avoidance technique [6], which is used to prevent performance degradation, and other failures subject to system crash due to software aging. Further, software rejuvenation may also employ fault tolerance to reduce the impact of individual failures on system availability [7]. To achieve rejuvenation, we have applied some most important concepts namely ***application restart, reboot and standby unit*** in our study. Application restart rejuvenation is a simple process to shut down the application and restart the application immediately. Various researchers [8, 9] tried to prevent performance degradation by this technique. Another familiar technique is virtual machine reboot. We have considered the concept of reboot when the complete virtual machine has failed. According to this, a shutdown signal is send to the guest operating system and then restarts virtual machine. These techniques focus on restarting the applications and system, so as to reduce the downtime cost as much as possible. The most significant approach is standby rejuvenation that reduces the failed requests. In this approach, standby virtual machine is always available to take over the active virtualize machine from the one being rejuvenated [10]. Repairable systems [11-13] has been investigated by many prominent researchers in different frameworks. Performance model has been examined for machine repair model with working breakdown, working vacation and warm standbys [14].

Cluster is a set of virtual machines and other related resources that act like a single system and provide high availability. These machines (also known as nodes) are usually identical. If one node fails, another can act as a backup. Therefore, cluster system can achieve higher availability in comparison to single machine system. Using a hierarchical approach, system availability of an *n*-processor VAX cluster has been determined [15]. Hardware and software reliability of clustered computing systems have been investigated. The pictorial view of a cluster system is shown in Fig. 1.

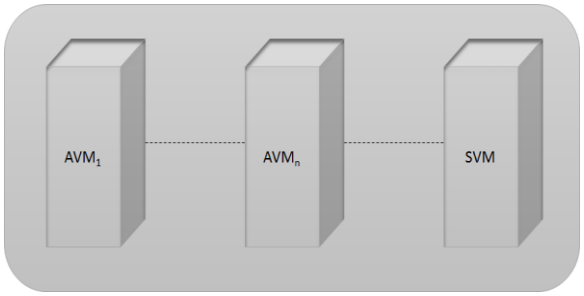


Fig. 1. Cluster-based system of active virtual machines with standby virtual machine

Clustering is a technique to achieve high availability of service by redundancy in hardware as well as software. To improve the availability of a system, redundancy is a common approach. In case of major failure clustering software immediately starts the applications on the standby system without any downtime. Jain and Preeti [16] used the concept of standby cluster system for availability analysis in their study. There is no doubt that redundancy increases the cost. Availability evaluation was done for machining system with multiple working vacations and standby [17]. However, if the cost of failure is high then the redundancy may be a smart choice [18]. Common mode failure associated with redundant systems wherein a failure can occur due to the installation faults, application errors, design errors and many more. This failure is one in which a single failure or condition affects the operation of multiple devices. Availability analysis of cluster-based system has been investigated by Hong, Wang and Shi [19]. Multi-layer cluster based system has been investigated by Sankar and Srinivasan [20]. In our study, we consider Cluster-based System (CS) having one Active Virtual Machine (AVM) and one Standby Virtual Machine (SVM). Initially, the CS is in working or healthy state. During aging ascertain process, administrator identify the number of faults occur to the considered CS which is marked as in unstable state or degradation mode. In rejuvenation state, the system administrator has a choice to use any of the three steps for reducing the down time, which are mentioned below:

- At rejuvenation state in AVM, administrator has a choice either to opt ***essential rejuvenation*** as first step or may go for ***optional rejuvenation service*** as second step. In the first step, we provide rejuvenation repair namely **application restart** as an essential rejuvenation. In the second step, we provide **reboot** to the system as an optional rejuvenation to reduce the downtime cost. Some researchers [21] have tried to reduce the downtime using reboot concept in their study.
- In the third step, he/she may decide to replace AVM with SVM. If one virtual machine has completely failed then another virtual machine is available to the system as a standby unit that can take over its workload. In SVM, administrator has again the choice to opt essential as first step or optional rejuvenation service as second step.

Fig. 2 shows the proposed CS having one AVM and one SVM wherein all above mentioned three steps are visualized.

From the literature survey, it has been noticed that the interest in studying the cluster system has mainly been focused on the system's behaviour depending upon its failure and repair characteristics without considering the concept of optional rejuvenation. We have not found any research work on cluster-based system with optional rejuvenation. Therefore, it is important to consider such system and advice to system developers how they can increase the system availability with the help of optional rejuvenation. If an error occurs in the cluster system, the system administrator can choose essential or optional rejuvenation. The main objective of the present investigation is to increase the availability of cluster system having standby machine. The rest of the paper is structured as follows. Section 2 describes assumptions and notations for the devolving cluster-based system. In Section 3, we develop the governing equations for considered system. Further, analysis has been done for obtaining the probability results in terms of availability. In Section 4, various

results are obtained for analysing the CS. To examine the effect of various parameters on the cluster system, numerical results are suggested in Section 5. Finally, concluding remarks are given in Section 6.

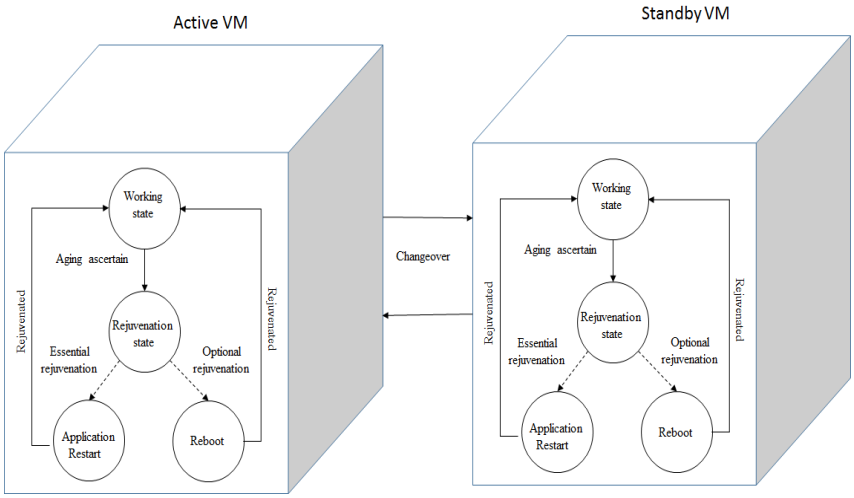


Fig. 2. Proposed system tectonics

2. Model description

To offer high level of availability for the server, we consider a CS having one AVM and SVM under the consideration optional software rejuvenation. To analyze cluster system, Markov modelling technique enables us to account for combinations of component failures sequence dependent consequences and also helps to identify the system performance characteristics that may need improvement for the optimization of the system. The AVM and SVM are subject to failure and repair. Once an AVM fails, SVM replaces it; the failure characteristic of the SVM becomes same as that of the AVM. For developing mathematical model of CS, we have provided various assumptions and notations.

Assumptions

- We consider a CS having AVM and SVM both in healthy state. The state of CS denotes by m (m=H_i, U_i, R_i, O_i, i=1, 2, and F, where i=1 represents the AVM and i=2 represents the SVM) which are given as

$$m = \begin{cases} H_i, & i = 1,2, \quad \text{Healthy State of } \dfrac{\text{AVM}}{\text{SVM}}, \\ U_i, & i = 1,2, \quad \text{Unstable State of } \dfrac{\text{AVM}}{\text{SVM}}, \\ R_i, & i = 1,2, \quad \text{Essential Rejuvenation State of } \dfrac{\text{AVM}}{\text{SVM}}, \\ O_i, & i = 1,2, \quad \text{Optional Rejuvenation State of } \dfrac{\text{AVM}}{\text{SVM}}, \\ F, & \quad \text{Failed State of CS.} \end{cases}$$

- As time goes on, AVM eventually transfer to unstable state. In unstable state, CS is in operational state but works under degraded mode. If any error occurs then the system goes under rejuvenation state. At this state, operator decides to take essential rejuvenation with probability $\boldsymbol{p}$ or opt optional rejuvenation with complementary probability $\boldsymbol{q} = 1 - \boldsymbol{p}$. On the other hand, if there is a major fault during unstable state then it is immediately replaced by SVM. If CS has suffered unplanned failure then the system is under failure state.
- In CS, the rate by which AVM transfer to unstable state is $i * \lambda_u$. From unstable state, the rate by which AVM transfer to rejuvenation state is $i * \lambda_r$ and to SVM is $i * \lambda_s$. The rate by which SVM transfer to failed state is λ.
- The switch over times from failure to repair, from repair to standby and from standby to operating states are negligible.
- The life-time and repair time of CS follow the exponential distribution with mean $1/\lambda$ and $1/\mu$.
- As unplanned failure occurs, CS stops working completely and can be restored by the reboot. After rebooting, the CS works with the same efficiency as before failure occurs.

Some notations are used for modelling purpose.

Notations

P_{H_1}: steady state probability that AVM is in Healthy state, i.e., fully working state.

P_{U_1}: steady state probability that AVM is in Unstable state, i.e., Degradation state.

P_{R_1}: steady state probability that AVM is in Essential Rejuvenation state.

P_{O_1}: steady state probability that AVM is in Optional Rejuvenation state.

P_{H_2}: steady state probability that SVM is in Healthy state, i.e., Fully Working state.

P_{U_2}: steady state probability that SVM is in Unstable state, i.e., Degradation state.

P_{R_2}: steady state probability that SVM is in Essential Rejuvenation state.

P_{O_2}: steady state probability that SVM is in Optional Rejuvenation state.

P_F: steady state probability that CS is in Completely Failed state.

λ_u: transition rate from Healthy state to Unstable state due to the effect of software aging.

λ_r: transition rate from Unstable state to Rejuvenation state.

λ_s: transition rate from transferring AVM to SVM.

λ: failure rate of CS.

μ_r: rejuvenation rate.

μ_o: optional rejuvenation rate.

μ: repair rate of CS.

3. Governing equations and analysis

The software aging models are being generally used to predict the system availability and other reliability indices. In our study, we use Markov analysis for devolving state transition model of a CS. For this purpose, we have provided governing equations of the model being concerned, which is depicted in Fig. 3.

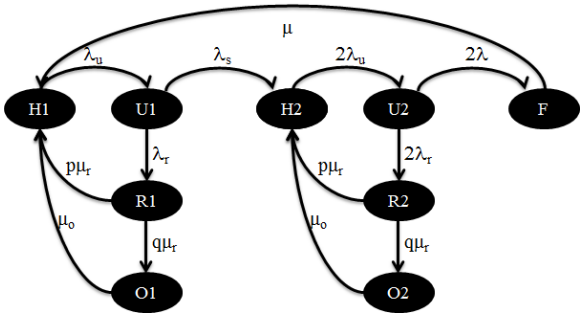


Fig. 3. State transition diagram for CS having AVM and SVM

(1) $$\lambda_u P_{H_1} = p\mu_r P_{R_1} + \mu_r P_{O_1} + \mu P_F,$$

(2) $$(\lambda_s + \lambda_r) P_{U_1} = \lambda_u P_{H_1},$$

(3) $$2\lambda_u P_{H_2} = \lambda_s P_{U_1} + p\mu_r P_{R_2} + \mu_r P_{O_2},$$

(4) $$(\lambda + \lambda_r) P_{U_2} = \lambda_u P_{H_2},$$

(5) $$\mu P_F = 2\lambda P_{U_2},$$

(6) $$\mu_r P_{R_i} = i\lambda_r P_{U_i}, \qquad i = 1, 2,$$

(7) $$\mu_o P_{O_i} = q\mu_r P_{R_i}, \qquad i = 1, 2,$$

Normalizing condition

(8) $$\sum_{i=1}^{2} P_{H_i} + \sum_{i=1}^{2} P_{U_i} + \sum_{i=1}^{2} P_{R_i} + \sum_{i=1}^{2} P_{O_i} = 1.$$

To obtain the closed-form solution of above equations (1)-(8), we apply product type method. The normalizing condition is combined with equations (1)-(8) and solving simultaneously, we have

(9) $$P_{U_1} = \frac{\lambda_u}{(\lambda_s+\lambda_r)} P_{H_1},$$

(10) $$P_{U_2} = \frac{\lambda_u \lambda_s}{2\lambda(\lambda_s+\lambda_r)} P_{H_1},$$

(11) $$P_{R_1} = \frac{\lambda_r \lambda_u}{\mu_r(\lambda_s+\lambda_r)} P_{H_1},$$

(12) $$P_{R_2} = \frac{\lambda_r \lambda_u \lambda_s}{\lambda\mu_r(\lambda_s+\lambda_r)} P_{H_1},$$

(13) $$P_{O_1} = \frac{q\lambda_r \lambda_u}{\mu_o(\lambda_s+\lambda_r)} P_{H_1},$$

(14) $$P_{O_2} = \frac{q\lambda_r \lambda_u \lambda_s}{\lambda\mu_o(\lambda_s+\lambda_r)} P_{H_1},$$

(15) $$P_{H_2} = \frac{\lambda_s(\lambda+\lambda_r)}{2\lambda(\lambda_s+\lambda_r)} P_{H_1},$$

(16) $$P_F = \frac{\lambda_u \lambda_s}{\mu(\lambda_s+\lambda_r)} P_{H_1},$$

where

$$P_{H_1} = \left[1 + a\left\{\frac{b}{2\lambda\lambda_r} + \frac{c}{\lambda_s} + \frac{q}{\lambda\mu_o}\right\}\right]^{-1},$$

and

$$a = \frac{\lambda_r \lambda_u \lambda_s}{(\lambda_s+\lambda_r)},\ b = \left[\frac{(\lambda+\lambda_r)}{\lambda_u} + 3\right],\ c = \left(\frac{1}{\lambda_r} + \frac{1}{\mu_r} + \frac{q}{\mu_o}\right).$$

4. Performance measures

To analyze the CS having AVM and SVM, the steady state probabilities of developed model are evaluated using product type method. It is very important to predict the performance of the CS in terms of steady state. Therefore, we have derived various performance measures that are mentioned below:

- **Availability of the system:** Availability (AV) models capture failure and repair behavior of CS, i.e., the cluster system is not available during rejuvenation and reboots states. Therefore, the availability of the CS is given by

(17) $$\mathrm{AV} = 1 - \left(\sum_{i=1}^{2} P_{R_i} + \sum_{i=1}^{2} P_{O_i} + P_F\right).$$

- **Long run probabilities of cluster system:** The long run probabilities of the system is given by when the CS is in essential rejuvenation state ($P(R)$) and optional rejuvenation state ($P(O)$), respectively as

(18) $$P(R) = \sum_{i=1}^{2} P_{R_i},$$

(19) $$P(O) = \sum_{i=1}^{2} P_{O_i}.$$

- **Downtime cost:** The downTime Cost (TC) of CS can be calculated using unavailable states of cluster system in terms of corresponding cost element

(20) $$\mathrm{TC} = \left[C_1 \times \sum_{i=1}^{2} P_{R_i} + C_2 \times \sum_{i=1}^{2} P_{O_i} + C_F \times P_F\right] \times T,$$

where

T is the operational time,
C_1 is the unit cost of the essential rejuvenation state of CS,
C_2 is the unit cost of the optional rejuvenation state of CS,
C_F is the unit cost of the reboot of CS.

5. Numerical results

In this this section, we provide numerical results for considered CS having AVM and SVM to evaluate system availability under the consideration of optional rejuvenation policy. By taking the numerical illustration, we have illustrated how different system parameters such as failure rates (λ_r, λ_s and λ) and rejuvenation rates (μ_r, μ_o and μ) would affect the availability of the system.

We have examined the effect of different parameters on various performance measures like availability, downtime cost, etc., for one year, i.e., T=1, by setting the default parameters as λ=0.5, λ_u=0.4, λ_r=0.2, λ_s=0.3, μ_r=0.8, μ_o=0.9, $\boldsymbol{p}$=0.5 and $\boldsymbol{q}$=0.5. The different cost elements for determining the values for downtime cost of CS are

chosen as C_1=\$50, C_2=\$25 and C_F=\$80. The flow chart of the model being concerned is depicted in Fig. 4.

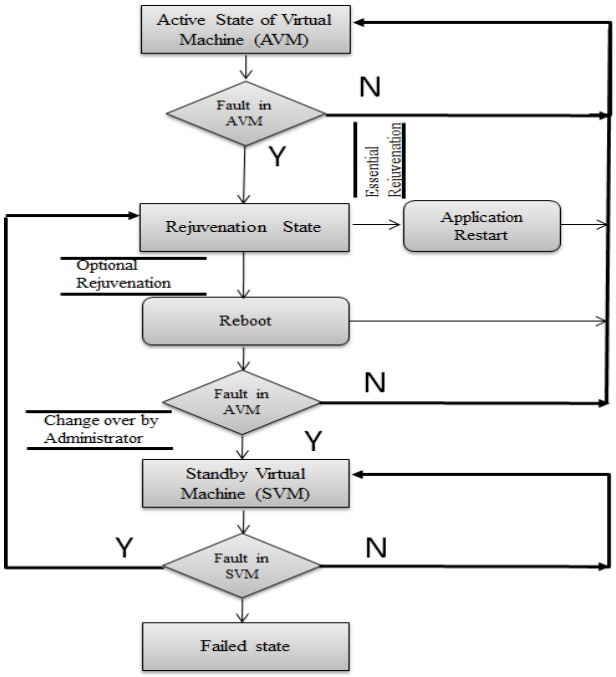


Fig. 4. Flow chart of proposed cluster system

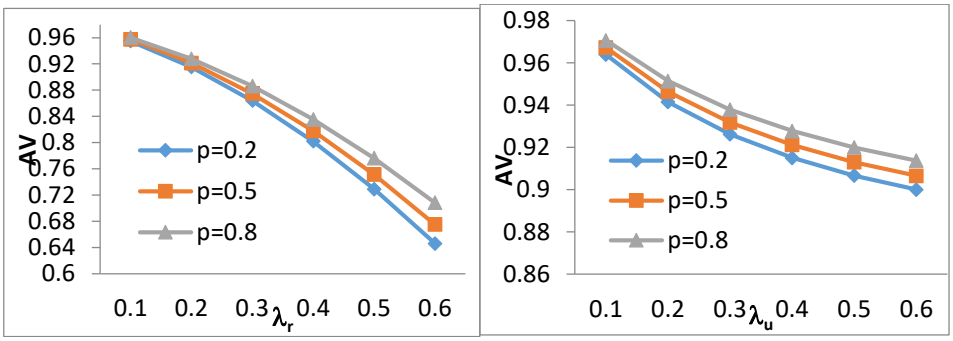
In Table 1, we provide the results for AVM which fails completely and we have replaced it by SVM. It can be easily seen from Table 1 that the values of long run probabilities in essential ($P(R)$) and optional ($P(O)$) rejuvenation state increases gradually. On the other hand, $P(R)$ increases as we increase the values of $\boldsymbol{p}$ and $P(O)$ shows the reverse trend.

Table 1. Effect on various performance measures of cluster system by varying λ_s and $\boldsymbol{p}$

λ_s	$\boldsymbol{p}$=0.2		$\boldsymbol{p}$=0.5		$\boldsymbol{p}$=0.8	
	$P(R)$	$P(O)$	$P(R)$	$P(O)$	$P(R)$	$P(O)$
0.1	0.0114	0.0084	0.0118	0.0054	0.0123	0.0022
0.2	0.0201	0.0150	0.0208	0.0097	0.0216	0.0040
0.3	0.0311	0.0237	0.0322	0.0154	0.0335	0.0063
0.4	0.0444	0.0344	0.0460	0.0223	0.0478	0.0092
0.5	0.0600	0.0471	0.0622	0.0305	0.0647	0.0126
0.6	0.0779	0.0618	0.0808	0.0400	0.0840	0.0166

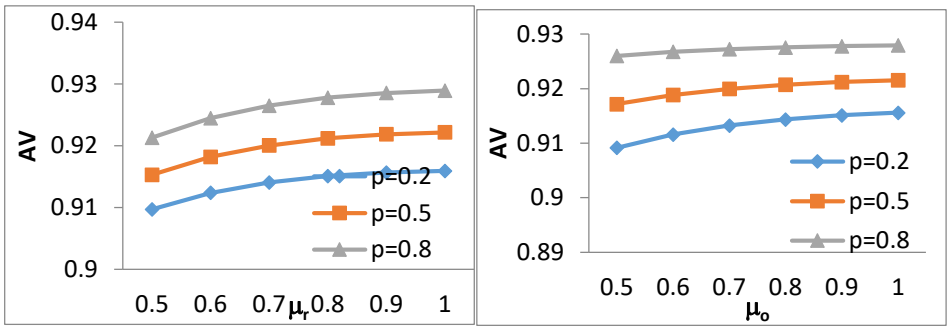
The graphical representation of availability performance and downtime cost of CS are summarized in Figs 5a-8b. From Figs 5a-b, it is observed that the availability of the CS shows a sharp decrement for increasing values of λ_r and λ_u. It can be easily observed form Figs 6a-b that as rejuvenation rates attains the higher values, system availability show the sharp increment which tally with many real life situations. Also, from theses Figs, it is observed that AV increases if system opt the essential rejuvenation.

The combined effect of probability, failure and rejuvenation rates are depicted in Figs 7a-8b. As the values of ***p*** increases, the downtime cost shows the sharp decrement. Form Figs 7a-b, it can be easily noticed that down time cost increases as λ_r and λ_u increases. The reverse trend of downtime cost can be observed in Figs 8a-b for higher values of rejuvenation rates.



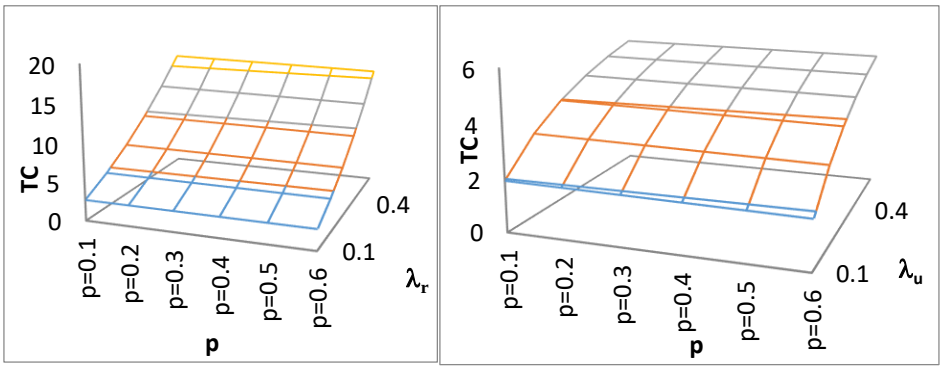
(a) (b)

Fig. 5. Effect of (a) λ_r and (b) λ_u on availability by varying ***p***



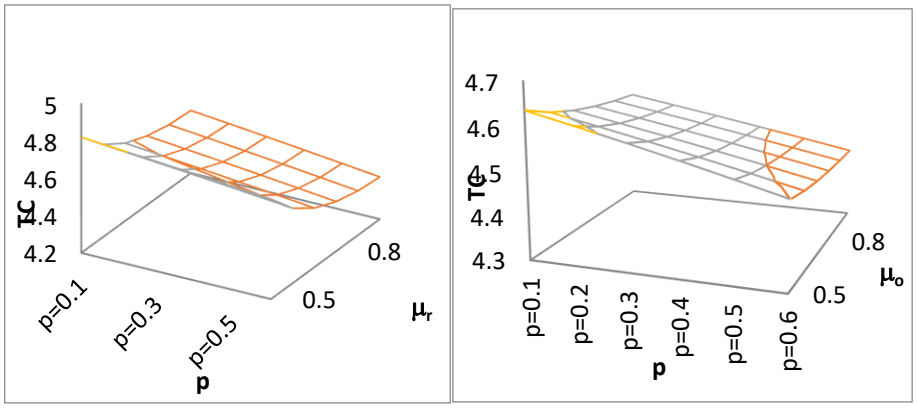
(a) (b)

Fig. 6. Effect of (a) μ_r and (b) μ_o on availability by varying ***p***



(a) (b)

Fig. 7. Combined effect of (a) λ_r and (b) λ_u on downtime cost by varying ***p***



(a) (b)

Fig. 8. Combined effect of (a) μ_r and (b) μ_o on downtime cost by varying ***p***

Finally, it can be judged on overall observation that the cluster system can achieve higher availability if we provide higher rejuvenation rates at appropriate time. Moreover, the downtime and downtime cost of CS approach decrease as rejuvenation rates increases.

6. Conclusions

This paper investigated a clustered system having active and standby virtual machine inflicted with software aging. To neutralize the software aging, we have used the concept of rejuvenation, which is highly effective to increase the availability of CS. This study proves that optional rejuvenation is a powerful approach in preventing the failures in the running process of the software executions. With the help of optional rejuvenation, the downtime cost of CS decreases tremendously. In this study, the CS having AVM and SVM deals with many concepts simultaneously such as software aging, software rejuvenation and reboot. The outcome of considered work may be applicable to many real time applications in which the prevention of the software failure is recommended for smooth running of the software applications under techno-economic constraints. It is expected that our study will be helpful for the system designers for improving the availability of cluster-based systems wherein software failures cannot be avoided.

References

1. Kaushik, M., G. Kumar, Preeti, R. Sharma. Availability Analysis for Embedded System with N-Version Programming Using Fuzzy Approach. – International Journal of Software Engineering, Technology and Applications, Vol. **1**, 2015, No 1, pp. 90-101.
2. Wang, D., W. Xie, K. S. Trivedi. Performability Analysis of Clustered Systems with Rejuvenation under Varying Workload. – Performance Evaluation, Vol. **64**, 2007, No 3, pp. 247-265.
3. Zhao, J., Y. Wang, G. R. Ning, K. S. Trivedi, R. Matias, K. Y. Cai. A Comprehensive Approach to Optimal Software Rejuvenation. – Performance Evaluation, Vol. **70**, 2013, No 11, pp. 917-933.

4. B a o, Y., X. S u n, K. S. T r i v e d i. A Workload-Based Analysis of Software Aging, and Rejuvenation. – IEEE Transactions on Reliability, Vol. **54**, 2005, No 3, pp. 541-548.
5. C o t r i n e o, D., R. N a t e l l a, R. P i e t r a n t u o n o, S. R u s s o. A Survey of Software Aging and Rejuvenation Studies. – ACM Journal on Emerging Technologies in Computing Systems, Vol. **10**, 2014, No 1, pp. A1-A35.
6. T r i v e d i, K. S., K. V a i d y a n a t h a n, K. G o s e v a-P o p s t o j a n o v a. Modeling and Analysis of Software Aging and Rejuvenation. – In: Proc. of 33rd Annual Simulation Symposium, 2000, p. 270.
7. N a g a r a j u, V., V. V. B a s a v a r a j, L. F i o n d e l l a. Software Rejuvenation of a Fault-Tolerant Server Subject to Correlated Failure. – In: Proc. of Annual Reliability and Maintainability Symposium, 2016, DOI: 10.1109/RAMS.2016.7448076.
8. G r o t t k e, M., L. L i, K. V a i d y a n a t h a n, K. S. T r i v e d i. Analysis of Software Aging in a Web Server. – IEEE Transactions on Reliability, Vol. **55**, 2006, No 3, pp. 411-420.
9. H u a n g, Y., C. K i n t a l a, N. K o l e t t i s, N. F u l t o n. Software Rejuvenation: Analysis, Module and Applications. – In: Proc. of 15th International Symposium on Fault-Tolerant Computing, 1995, pp. 381-390.
10. A l o n s o, J., R. M a t i a s, E. V i c e n t e, A. M a r i a, K. S. T r i v e d i. A Comparative Experimental Study of Software Rejuvenation Overhead. – Performance Evaluation, Vol. **70**, 2013, No 3, pp. 231-250.
11. S h a r m a, R., M. K a u s h i k, G. K u m a r. Reliability Analysis of an Embedded System with Multiple Vacations and Standby. – International Journal of Reliability and Applications, Vol. **16**, 2015, No 1, pp. 35-53.
12. Threshold N-Policy for Machine Interference Problem with Additional Repairman and Spares under Bernoulli Vacation Schedule. – In: Proc. of 11th International Conference on Reliability, Maintainability and Safety, IEEE Explore, 2017, pp. 1-5. DOI:10.1109/ICRMS.2016.8050058.
13. S h a r m a, R. Reliability Analysis for a Repairable System under N-Policy and Imperfect Coverage. – In: Proc. of International Multi Conference of Engineers and Computer Scientists, Vol. **2,** 2015, pp. 1001-1004.
14. J a i n, M., R. S h a r m a, R. M e e n a. Performance Modeling of Fault Tolerant Machining System with Working Vacation and Working Breakdown. – Arabian Journal for Science and Engineering, Vol. **44**, 2019, No 3, pp. 2825-2836.
15. I b e, O. C., R. C. H o w e, K. S. T r i v e d i. Approximate Availability Analysis of Vax Cluster Systems. – IEEE Transactions on Reliability, Vol. **38**, 1989, No 1, pp. 146-152.
16. J a i n, M., P r e e t i. Availability Analysis of Software Rejuvenation in Active/Standby Cluster System. – International Journal of Industrial and Systems Engineering, Vol. **19**, 2015, No 1, pp. 75-93.
17. J a i n, M., S. R a n i. Transient Analysis of Hardware and Software Systems with Warm Standbys and Switching Failures. – International Journal of Mathematics in Operational Research, Vol. **6**, 2013, No 1, pp. 1-28.
18. S h a r m a, R., G. K u m a r. Availability Improvement for the Successive K-out of-N Machining System Using Standby with Multiple Working Vacations. – International Journal of Reliability and Safety, Vol. **11**, 2017, No 3/4, pp. 256-267.
19. H o n g, Z., Y. W a n g, M. S h i. Advances in Future Computer and Control Systems. – Advances in Intelligent and Soft Computing, Berlin, Heidelberg, Springer, 2012, pp. 121-125.
20. S a n k a r, S., P. S r i n i v a s a n. Multi-Layer Cluster Based Energy Aware Routing Protocol for Internet of Things. – Cybernetics and Information Technologies, Vol. **18**, 2018, No 3, pp. 75-92.
21. Y a m a k i t a, K., H. Y a m a d a, K. K o n o. Phase-Based Reboot: Reusing Operating System Execution Phases for Cheap Reboot-Based Recovery. – In: Proc. of IEEE/IFIP 41st International Conference on Dependable Systems & Networks, 2011, DOI. 10.1109/DSN.2011.5958216.

Received: 26.07.2019; Second Version: 07.10.2019; Accepted: 19.10.2019