Availability Improvement in Software Aging Applications

May Tar Hla Myint, Thandar Thein
University of Computer Studies, Yangon
maytarhlamyint@gmail.com,thandartheinn@gmail.com

Abstract

With the explosive growth in Internet technology and the emergence of new and advanced applications, assured availability of computer systems has become a critical issue. In systems with high availability requirements, software aging can cause outages resulting in high costs. The challenge is to provide the desired availability and performance at a low cost software aging applications. To counteract this issue, we propose a combined method of software rejuvenation and virtualization. Virtualization can be helpful for software rejuvenation and fail-over in the occurrence of application failures and software aging. This paper constructs a stochastic model for virtualized servers to evaluate the effectiveness of our proposed combined method. The numerical derivation results are validated with the evaluation results through the SHARPE tool. The results demonstrate that our approach is a practical way to ensure uninterrupted availability and optimize performance for aging applications.

1. Introduction

When server applications execute continuously for long periods of time, the process corresponding to the software in executing age or slowly degrade with respect to effective use of their system resources. Exhaustion of system resources, data corruption, and numerical error accumulation are the primary symptoms of the degradation, which may eventually lead to performance degradation of the software, crash/hang failure, or other undesirable effects. That degradation of software is known as software aging. 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.

Software rejuvenation can be applied so as to mitigate adverse effects of software aging. Software rejuvenation is a proactive fault management technique aimed at cleaning up the system internal state to prevent the occurrence of more severe crash failures in the future [1]. To eliminate the service outage during software rejuvenation process, this paper combines rejuvenation with virtualization technology.

Software aging issues related to virtualization need to be addressed, especially for managing the availability [2]. Virtualization is a technology that combines or divides computing resources to present one or many operating environments using methodologies like hardware and software partitioning or aggregation, partial or complete machine simulation, emulation and others. A virtualization layer is a software layer that abstracts the physical resources for use by the Virtual Machines (VMs). A virtual machine is a tightly isolated software container that can run its own operating systems and applications as if were a physical computer.

This paper describes high availability mechanism for application servers. This approach has been designed to use over any server or service. The idea of the proposed system is to exploit the use of virtualization technology to improve the software rejuvenation for addressing the software aging problem.

The rest of this paper is organized as follows. Section 2 describes the related work. In section 3, the proposed system is described. Section 4 discusses the modeling, analysis of the model and validates the model with SHARPE (Symbolic Hierarchical Automated Reliability and Performance Evaluator) tool. Finally section 5 concludes the paper.

2. Related Work

A new technique for fast rejuvenation of Virtual Machine Monitors (VMMs) called the warm-VM reboot was proposed in [3]. The warm-VM reboot enables efficiently rebooting only a VMM by suspending and resuming VMs without accessing the memory images. To achieve this, they developed two mechanisms: on-memory suspend/resume of VMs and quick reload of VMMs.

Yong et al. [4] presented a software rejuvenation policy and set up the software rejuvenation model in a two-dimension state space

for a two-node application server cluster. Based on the organization structure and running state of the application server cluster, the system unavailability formula is derived using continuous time Markov process. Silva et al. [5] presented an approach for software rejuvenation based on automated self-healing techniques that can be easily applied to off-the-shelf Application Servers and Internet sites. They tested with a set of open-source Linux tools and the XEN virtualization and middleware.

In this paper, software rejuvenation and virtualization technology are combined to increase the system availability of the long running applications. The model of software system with virtualization is analysed and derive the steady-state availability of the system.

3. Proposed System

This section describes the proposed system to offer a high availability mechanism. This system applies the concept of virtualization technology and software rejuvenation. One Load Balancer Virtual Machine (LB-VM) and other operational Virtual Machines (VMs) are created on the top of the virtualization layer. LB VM will be used for monitoring purpose. The main application server will be running on one VM and the remaining VMs will be used for standby servers. Some software modules that will be responsible for the detection of software aging are installed in the Load Balancer VM. Each VM contains Software rejuvenation Agents (such as xSeries Software Rejuvenation Agent) [7] which were designed to monitor consumable resources.

As time degrades, the active VM running on the active physical server becomes software aging. When software aging or some potential anomaly happens in active server, the Load Balancer VM will trigger a rejuvenation operation. If the active VM is about to be rejuvenated, standby VM is started and then all the new requests and sessions are migrated from the active VM to standby VM. When the ongoing requests are finished in aging infected VM, these VM will be rejuvenated. If the system has more than one physical server, the process of the system will be continued to the first VM of another physical server. When the last VM of the active physical server becomes software aging, the in-flight requests and sessions are migrated to the standby VM in the standby physical server.

Using this approach the VMs cannot easily reach the failure state. This approach uses VMs as containers for the replicas in order to avoid the need for additional hardware and it can provide continued services during rejuvenation. This paper illustrates three Virtual Machines in two Physical Machines as a scenario.

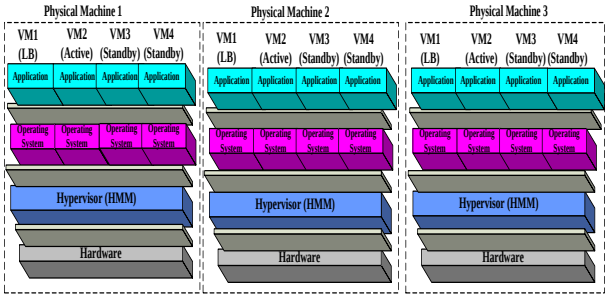

Figure 1. System Architecture of the proposed system

4. Modeling and Analysis

Using analytical modeling, this paper analyzes how the virtualization technology can improve the software rejuvenation action and can reduce the application's downtime and downtime cost. By mapping the actions as transitions in the stochastic models, mathematical steady-state solution of the chain is achieved. Figure 2 illustrates the state transition diagram of the proposed system. In this model, there are five states: healthy state (H_i), unstable state (U_i), switchover state (S_i), rejuvenation state (R_i) and failure state (H_{10}).

The assumptions used in the modeling are as follows:

- Failure rate (λ) and repair rate (μ) of the VMs are identical at all states.
- Unstable rate (λ_μ), rejuvenation rates (λ_r) of the VMs are identical at all states

As time progress, active VM eventually transit to unstable state with rate ($i * \lambda_\mu$). When the active VM is about to be rejuvenated, the application software may change from unstable state to switch over state at a rate of ($i * \lambda$). When ongoing requests are finished in the active VM, the active VM state may change from unstable state to rejuvenation state with rate ($i * \lambda_r$). In failure state (H_{10}), all VMs stop running and no available VM remains.

After the rejuvenation, one of the healthy rejuvenated VM takes over the role of primary VM. So service is not stopped even during the rejuvenation process. If the primary VM has encounters the software aging, the standby component takes the role of the primary VM and continues to operate based on the current state.
When all the requests are finished in the primary server, then the primary VM will be rejuvenated.

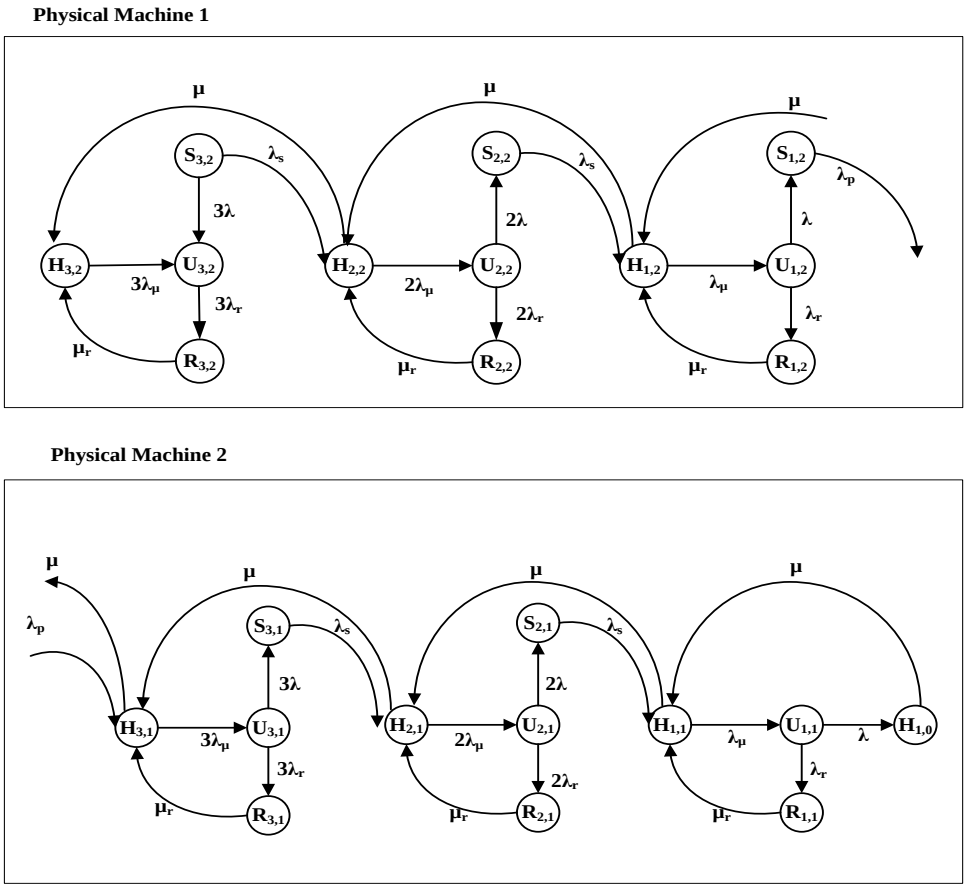

Figure 2. State transition diagram of the proposed model

The state transition diagram in figure 2 can be described as Markov process class. It can perform the steady state analysis of the diagram. The steady-state balance equations of figure 2 are as follows:

For state H_{nm}(n=1,2,3 & m=1,2)

$$n\lambda_{\mu}P_{H_{nm}} = \mu P_{H_{(n-1),m}} + \mu_r P_{R_{n,m}} \quad (1)$$

For state H_{ij} (i = 1,2,3 & j= 1,2)

$$(\mu + i\lambda_{\mu})P_{H_{ij}} = \lambda_s P_{S_{i+1}} + \mu P_{R_{ij}} + \mu P_{H_{(i-1),j}} \quad (2)$$

For state H_{10}

$$\mu P_{H_{10}} = \lambda P_{U_{11}} \quad (3)$$

For state U_{ij} (i = 1,2,3 & j = 1,2)

$$(\lambda + \lambda_r)P_{U_{ij}} = \lambda_{\mu} P_{U_{ij}} \quad (4)$$

For state R_{ij} (i = 1,2,3 & j = 1,2)

$$\mu_r P_{R_{ij}} = i\lambda_r P_{U_{ij}} \quad (5)$$

For state S_{ij} ((i = 1,2,3 if j = 2) , (i = 2,3 if j = 1))

$$\lambda_s P_{S_{ij}} = i\lambda P_{U_{ij}} \quad (6)$$

The conservation equation is obtained by summing the probabilities of all states in the system and set the sum equal to 1.

$$\sum_{i=1}^{n}\sum_{j=1}^{m} P_{H_{ij}} + P_{H_{10}} + \sum_{i=1}^{n}\sum_{j=1}^{m} P_{U_{ij}} + \sum_{i=1}^{n}\sum_{j=1}^{m} P_{R_{ij}} + \sum_{i=1}^{n}\sum_{j=1}^{m} P_{S_{ij}} = 1 \quad (7)$$

Combining the balance equations with the conservation equation, and solving these simultaneous equations, it is acquired the closed-form solutions for the model.

$$P_{U_{ij}} = \frac{3!}{i!}\left(\frac{\lambda}{\mu}\right)^{3-i}\left(\frac{\lambda_{\mu}}{\lambda + \lambda_r}\right)^{4-i} P_{H_{32}} \quad (8)$$

$$P_{R_{ij}} = i(i+1)..3\frac{\lambda_r}{\mu_r}\left(\frac{\lambda}{\mu}\right)^{3-i}\left(\frac{\lambda_{\mu}}{\lambda + \lambda_r}\right)^{4-i} P_{H_{32}} \quad (9)$$

$$P_{S_{ij}} = i(i+1)..3\frac{\lambda}{\lambda_s}\left(\frac{\lambda}{\mu}\right)^{3-i}\left(\frac{\lambda_{\mu}}{\lambda + \lambda_r}\right)^{4-i} P_{H_{32}} \quad (10)$$

$$P_{H_{ij}} = 3(3-i)(\frac{\lambda}{\mu})^{3-i}(\frac{\lambda_\mu}{\lambda+\lambda_r})^{3-i} P_{H_{32}} \quad (11)$$

$$P_{H_{32}} = [1+\frac{\lambda_\mu}{\lambda+\lambda_r}\{1+\frac{3\lambda_r}{\mu_r}+\frac{3\lambda}{\lambda_s}+\frac{3\lambda}{\mu}+ \quad (12)$$

$$A\left(1+\frac{2\lambda_r}{\mu_r}+\frac{2\lambda}{\lambda_s}+\frac{2\lambda}{\mu}\right)$$

$$+B\left(1+\frac{\lambda_r}{\mu_r}+\frac{\lambda}{\lambda_P}+\frac{\lambda}{\mu}\right)+C\left(1+\frac{3\lambda_r}{\mu_r}+\frac{3\lambda}{\lambda_s}+\frac{3\lambda}{\mu}\right)$$

$$+D\left(1+\frac{2\lambda_r}{\mu_r}+\frac{2\lambda}{\lambda_s}+\frac{2\lambda}{\mu}\right)+E\left(1+\frac{\lambda_r}{\mu_r}+\frac{\lambda}{\mu}\right)\}]^{-1}$$

$$A = \frac{3\lambda}{\mu}\frac{\lambda_\mu}{\lambda+\lambda_r}$$

$$B = \frac{2\lambda}{\mu}\frac{\lambda_\mu}{\lambda+\lambda_r}A$$

$$C = \frac{\lambda}{\mu}\frac{\lambda_\mu}{\lambda+\lambda_r}B$$

$$D = \frac{3\lambda}{\mu}\frac{\lambda_\mu}{\lambda+\lambda_r}C$$

The system is not available in the rejuvenation state (R_{11}) in the last VM in the last physical server and the failure state (H_{10}). The system availability in the steady-state is defined as follows:

$$Availabiltiy = 1-(P_{R_{11}}+P_{H_{10}}) \quad (13)$$

Planned shutdown cost is less than that of unexpected shutdown ($C_f >> C_r$), where C_r is the unit cost of unexpected shutdown of a server, and C_r is the unit cost of rejuvenation process. The expected total downtime and downtime cost of the system with rejuvenation in an interval of T time units are:

$$Downtime(T) = (P_{R_{11}}+P_{H_{10}})\times T \quad (14)$$

$$DowntimeCost(T) = (P_{H_{10}}\times C_f + P_{R_{11}}\times C_r)\times T \quad (15)$$

The applicability of the model and solution methodology through numerical examples is illustrated in this section. The exact model parameter values for the model are not known, a good estimate value for a range of model parameters is assumed. For this purpose, the failure profile [1] and [14] are used in the experiments.

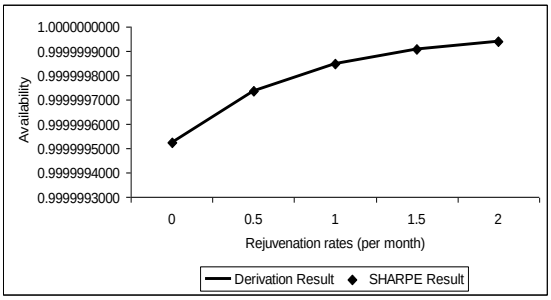

Figure 3. Availability vs. different rejuvenation rates

Figure 3 illustrates the numerical results for three virtual machines in two physical machines (3VMs 2Ps). It can be observed that the faster the rejuvenation rate, the system can achieve the higher availability.

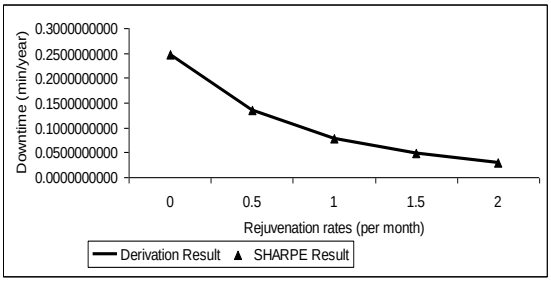

Figure 4. Downtime vs. different rejuvenation rates

Figure 4 plotted the downtime as a function of the rejuvenation rates.

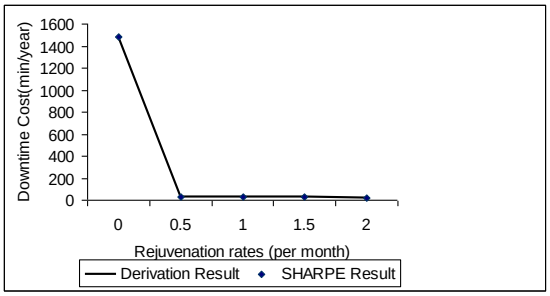

Figure 5. Downtime cost vs. different rejuvenation rates

The downtime cost of planned shutdown is much lower than that of an unplanned shutdown. Figure 5 shows the downtime cost of our system. From the result, it is apparent that the proposed rejuvenation system is a cost effective way to build a high availability system and virtualization technology can improve software rejuvenation. According to Figure 4, 5 and 6, it is found that the derivation results and SHARPE tool simulation results are the same.

5. Conclusion

Software can cause software aging as time degrades. Software aging decreases the availability

of the system. To counteract the software aging, the most effective way is software rejuvenation. This paper presents a simple approach for enhanced software rejuvenation that has proved to be highly effective. Markov model for analyzing software rejuvenation in continuously running applications are presented and express availability, downtime and downtime costs in terms of the parameters in the model. The numerical results are validated with the evaluation results through SHARPE tool. It is found that the derivation results and the SHARPE result are the same. Virtualization and software rejuvenation can be used to prolong the availability of the services. It can be shown that virtualization can be helpful for software rejuvenation and fail-over in the occurrence of application failures and software aging.

6. References

[1] Y. Huang, C. Kintala, N. Kolettis, N. D. Fulton, "Software rejuvenation: Analysis, module and application," In Proc. the 25th Int. Symp., Fault-Tolerant Computing, Pasadena, CA, June. 1995, pp.381-390. doi: 10.1109/FTCS.1995.466961

[2] F. Machida, D. S. Kim, J. S. Park, K. S. Trivedi, "Towards Optimal Virtual Machine Placement and Rejuvenation Scheduling in a Virtualized Data Center," Proc. Int. Workshop on Software Aging and Rejuvenation (WoSAR 2008), IEEE Computer Society, 2008.

[3] K. Kourai, and S. Chiba, "A fast rejuvenation technique for server consolidation with virtual machines," In Int. Conf. on Dependable Systems and Networks (DSN), June. 2007, pp. 245-255. doi: 10.1109/DSN.2007.67

[4] Q. Yong, M. Haining, H. Di and C.Ying, "A Study on Software Rejuvenation Model of Application Server Cluster in Two-Dimension State Space Using Markov Process", Information Technology Journal, vol. 7 (1) , 2008, pp.98-104, doi: 10.3923/itj.2008.98.104

[5] L. M.Silva, J. Alonso, P. Silva, J. Torres, A. Andrzejak, "Using virtualization to improve software rejuvenation," Proc. The 6th IEEE Int. Symp. Network Computing and Applications, (NCA 2007), July. 2007, pp.33-44, doi: 10.1109/NCA.2007.53.

[6] K. S. Trivedi, K. Vaidyanathan, K. Goseva-Popstojanova, "Modeling and Analysis of Software Aging and Rejuvenation", Proc. 33rd Annunal Simulation Symposium, April. 2000 pp. 270-279, doi:10.1109/SIMSYM.2000.844925

[7] V. Castelli, R. Harper, P. Heidelberg, S. Hunter, K. S. Trivedi, K. Vaidyanathan, and w. Zeggert, "Proactive management of software aging," IBM J. Research & Development, 45(2), 2001.

[8] S.Garg, A. Puliafio, M. Telek, K. S. Trivedi, "Analysis of preventive maintenance in transitions based software system," IEEE Transactions on Computers, vol. 47 (1), Jan. 1998, pp. 96-107, doi:10.1109/12.656092

[9] H. Ramasamy and M. Schunter, "Architecting Dependable Systems Using Virtualization," Proc. DSN 2007 Workshop on Architecting Dependable Systems. 2007, pp. 127-149. doi: 10.1007/978-3-540-85571-2_6.

[10] A. Andrzejak, M. Moser, L. M. Silver (2007). "Managing Performance of Aging Application via Synchronized Replica Rejuvenation," Proc. 18th IFIP/IEEE International workshop on Distributed Systems: Operations and Management Managing Virtualization of Networks and Services (DSOM 2007), Oct 2007, pp. 98-109.

[11] M. Grottke, L. Li, K. Vaidyanathan, K. S. Trivedi, "Analysis of software aging in a Web server," IEEE Transactions on Reliability," Sept. 2006, vol. 55(3), pp.411-420, doi: 10.1109/TR.2006.879609.

[12] B. Barham , B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, "Xen and the art of virtualization," In Proc. the 19th Int. ACM Symp. Operating Systems Principles, New York, USA, ACM Press, 2003, pp.164-177.

[13] K. Vaidyanathan, K. S. Trivedi, "A measurement-based model for estimation of software aging in operational software systems," In Proc. the 10th IEEE Int. Symp., Software Reliability Engineering, Boca Raton, FL, USA, Nov. 1-4, 1999, pp.88-93.

[14] Software rejuvenation. Department of Electrical and Computer Engineering, Duke University, [Online] Available form: http://www.software-rejuvenation.com

[15] K. S. Trivedi, "SHARPE 2002: Symbolic hierarchical automated reliability and performance evaluator", Proc. Int. Conference on Dependable Systems and Networks, 2002, p.544.

[16] M. F. Mergen, V. Uhlig, O. Krieger, J. Xenidis, "Virtualization for high-performance computing," Operating Systems Review 40 (2): 8-11.

[17] K. Rinsaka and T. Dohi, "Behavioral Analysis of a Fault-tolerant Software System with Rejuvenation," Proc. Autonomous Decentralized Systems (IEICE 2005), April. 2005, pp.159-166, doi: 10.1109/ISADS.2005.1452042.