# Proactive Software Rejuvenation Solution with Enhanced VM Migration Decision in IT Infrastructure

Aye Myat Myat Paing[1], Khin Moh Moh Lwin[2], Ni Lar Thein[1]
*University of Computer Studies, Yangon* [1], *Myanmar Maritime University* [2],
*University of Computer Studies, Yangon* [1]
*paing.ayemyat@gmail.com*[1], *khinmohlwin@gmail.com*[2]

## Abstract

*The availability of IT infrastructures is still a huge challenge nowadays. As server virtualization is used as an essential software infrastructure of various software services in IT environment and it is emerging as a technique to increase system reliability and availability. To prevent system failures caused by software aging, software rejuvenation can be applied in virtualized environment. Software aging of virtual machine monitors (VMMs) is becoming critical because performance degradation or crash failure of a VMM affects all virtual machines (VMs) on it. Live VM migration enables a running VM on a host server to move onto the other host server with very small interruption of the execution. VM migration depends on a variety of criteria and efficient decision support is required. The work presented in this paper aims to offer the high availability against software aging of virtualized server system by providing VM migration based VMM software rejuvenation solution. First, we present the resource usage as accepting as many services as in virtualized environment which support of VM migration. Second, we present migration based VMM rejuvenation analytic model and evaluate the steady-state system availability based on familiar Markovian analysis through the use of numerical analysis.*

Keywords: **Availability, Proactive software rejuvenation, Software Aging, Stochastic modeling, Virtualization.**

## 1. Introduction

Availability of information and services all the time and from everywhere is today a growing common requirement. One of the possible definitions of availability is the quality of being at hand when needed [10]. Virtualization is a rapidly growing new technology that is transforming the world of IT. Virtualization has proved to be a successful tool for the management of complex IT environments and it is emerging as a technique to increase system reliability and availability [5, 7]. Server virtualization becomes an essential software component of system infrastructure of various software services in IT environment. It is well known that, currently, computer system outages are more often due to software failure than hardware failure. Several studies [2, 3, 6, 12] have reported that one of the causes of the unplanned software outages is the software aging phenomena. Software aging will affect the performance of the application and eventually cause it to fail. Software aging has also been observed in widely used communication software like Internet Explorer, Netscape as well as commercial operating systems and middleware.

Recently, software aging of VMMs is becoming critical. Many VMs run on top of a VMM in one machine consolidating multiple servers and aging of the VMM directly affects all the VMs. Software rejuvenation is a preventive and proactive maintenance policy that is particularly useful for counteracting the software aging. One of the promises of virtualization is the ability to allow applications to dynamically move from one physical server to another as the demands and resource availabilities change, without service interruption. More recently, Xen and VMWare have implemented "live" migration of VMs that involve extremely short downtimes ranging from tens of milliseconds to a second.

To migrate, we must know which virtual machine needs to be migrated and when this relocation has to be done and, moreover, which host must be destined. In this work, to avoid the downtime during rejuvenation process, we take the advantage of live VM migration to offer a proactive software rejuvenation solution against software aging for virtualized IT infrastructure.

The organization of this paper is as follows. In section 2 we discuss the related work. The system architecture for resource usage management and VM migration decision policy are presented in section 3. The proposed model for enhancing system availability follows in section 4. Finally we conclude our paper in section 5.

## 2. Related Work

In this section we review selected publications related to our work. Huang et.al [3] considered degradation performance due to software aging phenomenon as a two-step stochastic process. From the clean state the software system jumps into a degraded state from which two actions are possible: rejuvenation with return to the clean state or transition to the complete failure state. They model a four-state process as a Continuous Time Markov Chain (CTMC) and derive the steady-state system availability and the expected operation cost per unit time in the steady state.

The authors [4] presented comprehensive availability models of three rejuvenation techniques for a server virtualized system with time-based rejuvenations for VM and VMM. The result of sensitivity analysis showed that Migrate-VM rejuvenation achieves the best steady-state availability as long as VM live migration is fast enough and other servers have capacity to receive the migrated VM. In our work we consider to improve software rejuvenation process in virtualized server system.

The authors [1] present a technique that offers high availability for Grid services based on concepts like virtualization, clustering and software rejuvenation. In paper [9], virtual machine based software rejuvenation (VMSR) model is developed using stochastic modeling. But it only captured VM level behavior.

## 3. System Architecture for Resource Usage Management

This system consists of virtualized cluster severs and virtualized management server. We consider the resource usage management that is based on the system architecture as shown in Figure 1. The virtualized management server consists of VM that includes resource manager, rejuvenation manager and aging detector. The aging detector of management server is responsible for the detection of software aging. When software aging happens in active VMM, the rejuvenation manager will trigger the rejuvenation action. The resource manager is responsible for getting resource information and generating migration decisions.

This section describes how to solve the software aging failure in VMM using software rejuvenation and live VM migration. Live VM migration technology allows us to migrate a running VM without any or minimal outage, from one physical machine to another. As time degrades, the VMM running on the physical machine (PM) becomes software aging.

When the software aging or some potential anomaly happens in one of the PM in the resource pool, the rejuvenation manager of management server will trigger a rejuvenation operation. If the aging affected PM is about to be rejuvenated, all the new requests and sessions are migrated from the VMs from the aging affected PM to VMs from other PMs in the resource pool.

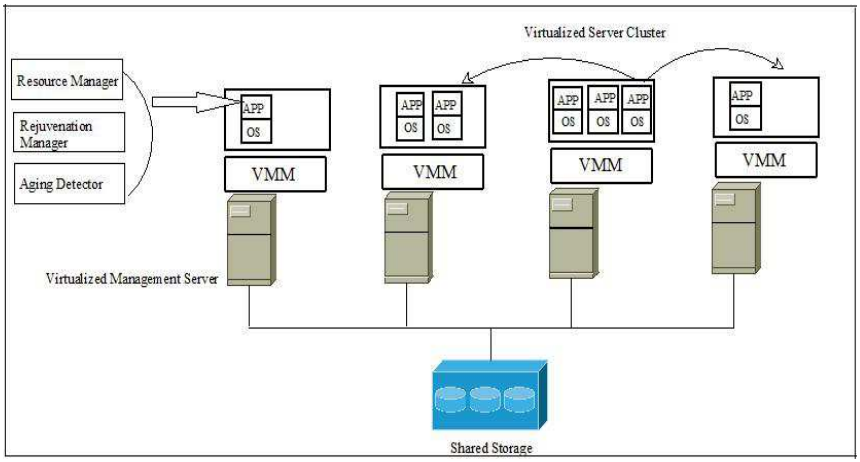

**Figure 1. System architecture**

When the ongoing requests are finished in aging infected PM, these VMM will be rejuvenated. With the advent of the virtualization, live migration can be performed without service interruption. In order to carry out the appropriate VM live migration, the proposed Resource Manager (RM) model carries out how to allocate the VMs to accept the maximum number of services on the virtualized infrastructure according to the resources available.

The assumptions used in the resource management are as follow.

- It does not allowed same service in one PM because it can be decided to which PM of each VM must be allocated such that number of VMs is maximized the services and VMs requirements are satisfied without exceeding PM's capacity and whole infrastructure capacity limits.
- One of the PMs in the resource pool can suffer software aging at a time. A heartbeat keep-alive system is used to monitor the health of the nodes between them.

## 3.1 VM Migration Decision Policy

In this sub section, the VM migration decision policy is presented that is used for VMM rejuvenation. When one of the PMs in the resource pool suffers aging problem, all of the VMs on the aging affected PM are needed to migrate onto other PMs in the resource pool before the aging PM is repaired. In order to carry out the appropriate VM live migration, the proposed Resource Manager (RM) model carries out the required processes for migration decisions as follows:

i. Detecting the VMM aging that needed to be rejuvenate
ii. Monitoring resource information of all active PMs
iii. Ranking VMs on aging affected PM and other active PMs
iv. Selecting destination PMs for VMs on aging affected PM
v. Providing Migration Decision for Live Migration

# 4. Proposed Model for Enhancing System Availability

In this section, we present state transaction diagram with time-based VMM rejuvenation policy for virtualized servers system as shown in Figure 2.

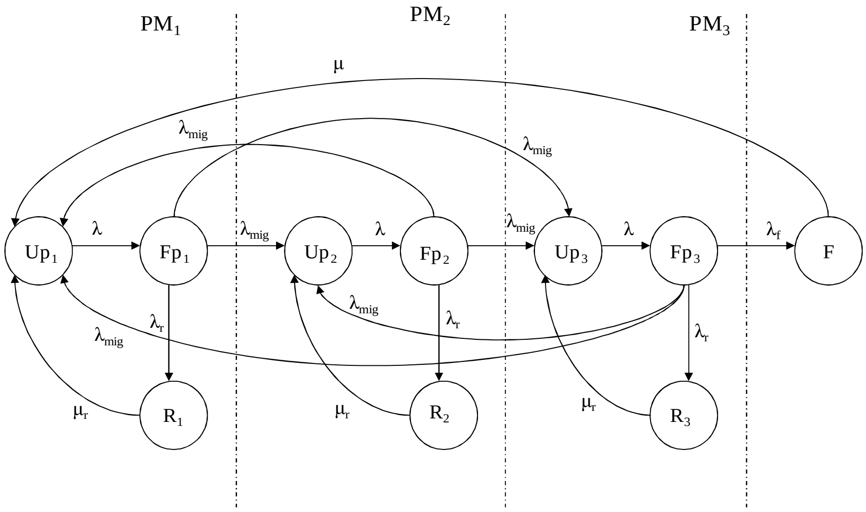


**Figure 2. State transaction diagram of system availability**

If the active VMM is about to be rejuvenated cause of software aging, all the VMs on aging affected PM are migrated to other PMs according to migration decision and then will be started for the new requests and sessions. It can return back to the original PM after the completion of the VMM rejuvenation through live VM migration.

The above state diagram has ten states in total. Each state has probability of being in its own state. The states are given as per $PM_1$, $PM_2$ and $PM_3$ in the resource pool. In $PM_1$, there are three states like Up state ($UP_1$), Failure Probably state ($FP_1$) and Rejuvenation state ($R_1$). We use Markov chain formulation for analyzing the above states. $PM_1$ provides the services in the Up state ($UP_1$). If VMM degrades then it moves to the Failure Probably state ($FP_1$) through rate $\lambda$. The resource manager (RM) model is smart enough to move from $PM_1$ to Up states of other PMs in the resource pool according to their available resources with live migration rate $\lambda_{mig}$. It depends upon the how fast RM model gets activated to move to other PMs. $PM_2$ and $PM_3$ also have three states and work as $PM_1$. From $FP_1$ state, moves to the VMM rejuvenation state ($R_1$) with rate $\lambda_r$. We try to consider almost all possibilities of going down when there is no active PMs in the resource pool. From the system outage, all PMs can be repaired through the rate of $\mu$.

**Table 1. Steady-state probabilities**

| Symbol | Meaning |
|---|---|
| $\pi Up_1$ | Probability of VMM Up state in $PM_1$ |
| $\pi Fp_1$ | Probability of VMM Failure Probably state in $PM_1$ |
| $\pi R_1$ | Probability of VMM Rejuvenation state in $PM_1$ |
| $\pi Up_2$ | Probability of VMM Up state in $PM_2$ |
| $\pi Fp_2$ | Probability of VMM Failure Probably state in $PM_2$ |
| $\pi R_2$ | Probability of VMM Rejuvenation state in $PM_2$ |
| $\pi Up_3$ | Probability of VMM Up state in $PM_3$ |
| $\pi Fp_3$ | Probability of VMM Failure Probably state in $PM_3$ |
| $\pi R_3$ | Probability of VMM Rejuvenation state in $PM_3$ |
| $\pi F$ | Probability of being in Failure state |

The conservation equation of Figure 2 is obtained by summing the probabilities of all states in the system and the sum of the equation is 1. The meaning of steady state probabilities is shown in Table 1.

$$\sum_{i=1}^{3}\pi_{Up_i}+\sum_{i=1}^{3}\pi_{FP_i}+\sum_{i=1}^{3}\pi_{R_i}+\pi_F=1 \quad (1)$$

We acquire the closed-form solution for the system.

$$\pi_{Fp_1}=\left(\frac{\lambda}{2\lambda_{mig}+\lambda_r}\right)\pi_{Up1} \quad (2)$$

$$\pi_{R_1}=\frac{\lambda_r}{\mu_r}\left(\frac{\lambda}{2\lambda_{mig}+\lambda_r}\right)\pi_{Up1} \quad (3)$$

$$\pi_{Up_2}=\left(\frac{2\lambda_{mig}+\lambda_r}{\lambda}\right)B\pi_{Up1} \quad (4)$$

$$\pi_{Fp_2}=B\pi_{Up1} \quad (5)$$

$$\pi_{R_2}=\frac{\lambda_r}{\mu_r}B\pi_{Up1} \quad (6)$$

$$\pi_{Up_3}=\left(\frac{2\lambda_{mig}+\lambda_f+\lambda_r}{\lambda}\right)A\pi_{Up1} \quad (7)$$

$$\pi_{Fp_3}=A\pi_{Up1} \quad (8)$$

$$\pi_{R_3}=\frac{\lambda_r}{\mu_r}A\pi_{Up1} \quad (9)$$

$$\pi_F=\frac{\lambda_f}{\mu}A\pi_{Up1} \quad (10)$$

$$\pi_{Up1}=\left[\begin{array}{l}1+\left(\frac{\lambda}{2\lambda_{mig}+\lambda_r}\right)+\frac{\lambda_r}{\mu_r}\left(\frac{\lambda}{2\lambda_{mig}+\lambda_r}\right)+\left(\frac{2\lambda_{mig}+\lambda_r}{\lambda}\right)B\\ B+\frac{\lambda_r}{\mu_r}B+\left(\frac{2\lambda_{mig}+\lambda_f+\lambda_r}{\lambda}\right)A+A+\frac{\lambda_r}{\mu_r}A+\frac{\lambda_f}{\mu}A\end{array}\right]^{-1} \quad (11)$$

Where $A=\left(\frac{\lambda_{mig}}{2\lambda_{mig}+\lambda_f}\right)\left(\frac{\lambda}{2\lambda_{mig}+\lambda_r}\right)$

$$B=\frac{1}{\lambda_{mig}}\left[\lambda-\lambda_{mig}A+\lambda_r\left(\frac{\lambda}{2\lambda_{mig}+\lambda_r}\right)+\lambda_f A\right]$$

## 4.1 Availability and Downtime Analysis

Availability is a probability of a system which provides the services in a given instant time. In this model, services are not available when all VMMs are in failure state. Markov chain will be classified as up states or down states. The system is not available in the failure state (F). Downtime is the expected total downtime of the application with rejuvenation in an interval of T time units is:

The system availability in the steady-state and Downtime are defined as follows:

*Availability = 1-Unavailability* (12)

*Availability* = 1-$\pi$F (13)

*Downtime (T)* = T* $\pi$F (14)

In order to examine the behavior of the system studied in this paper, we perform numerical analysis using the system-parameters [3, 8] as shown in Table 2.

**Table 2. Parameters values and description**

| Parameter | Description | Values |
|---|---|---|
| $\lambda$ | Failure Probably Rate | 3 times/ day |
| $1/\lambda_{mig}$ | Migration Time | 5 sec |
| $\lambda_f$ | Failure Rate | 3 times/month |
| $\lambda_r$ | Rejuvenation Trigger Rate | 1 time/ week |
| $1/\mu_r$ | Rejuvenation Time | 5 mins |
| $1/\mu$ | Repair Time | 2 hours |
| T | Unit Time Interval | 24*30*12 days |

The influence of VM migration time along with different VMM failure probably time on availability is shown in Figure 3. The VM migration times are assumed 5 sec and 10 sec. The lower mean time to VM migration, the higher availability of our system model can be achieved. Therefore, the availability is dependent on the VM migration time.

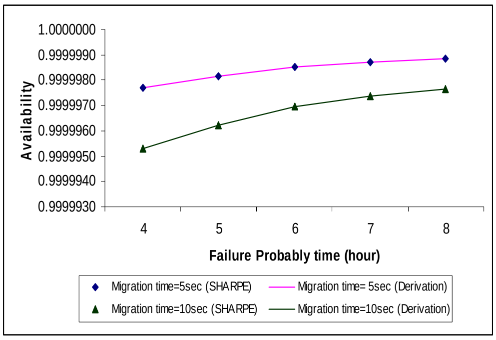


**Figure 3. Availability vs different VMM failure probably and VM migration time**

The Figure 4 shows the differences in downtime with different VM migration time and different VMM failure probably time. From the result, it is apparent that by combining virtualization technology and migration decision based software rejuvenation mechanism can enhance the availability of IT infrastructure.

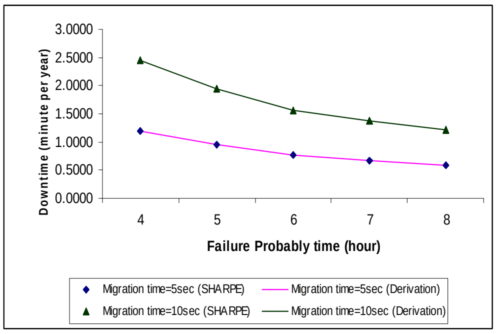


**Figure 4. Downtime vs different VMM failure probably and VM migration time**

SHARPE [11] is well known package in the field of reliability and performability. According to Figure 3 and 4, it is found that the derivation results and SHARPE tool simulation results are the same.

## 5. Conclusion

In this paper, we have presented a highly effective software solution against software aging phenomena. Our solution makes use of live VM migration and it is found that live VM migration is very helpful for rejuvenation process of VMM. We have presented a Markov model of time-based proactive rejuvenation policy for virtualized cluster servers system and have shown some numerical results. We have evaluated system availability, downtime. According to the numerical results, combined approach of virtualization technology and proactive software rejuvenation methodology can enhance the availability of virtualized IT infrastructure and can reduce the downtime of the system. Moreover we also presented a resource manager model to guarantee the availability of the services deployed and support VM Migration decision in order to achieve minimize downtime even in case of service restart.

## References


[1] Alonso J., Silva L., Andrzejak A., Silva P., Torres J. High-Available Grid Services through the use of Virtualized Clustering, In Proceedings of the 8th IEEE/ACM International Conference on Grid computing, Austin, Texas, Sept. 19-20, 2007, pp.34-41.

[2] Grottke M. J. R, and Trivedi K. S. The fundamentals of software aging. In Proceedings of the 1st International Workshop on Software Aging and Rejuvenation/19th IEEE International Symposium on Software Reliability Engineering, 2008.

[3] Huang K. C. K. N., Yennun and Fulton N. D. Software rejuvenation: Analysis, module and applications. In FTCS '95: In Proceedings of the Twenty-Fifth International Symposium on Fault-Tolerant Computing, page 381,

Washington, DC, USA, 1995. IEEE Computer Society.

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

[5] Nanda S and Chiueh T. A survey on virtualization technologies, Stony Brook University, Tech. Rep. TR-179, Feb 2005.

[6] Peret S. and Narasimham P. Causes of Failure in Web applications. Technical Report CMU-PDL-05-109, Carnegie Mellon Univ, Dec. 2005.

[7] Silva L M, Alonso J, Silva P, Torres J, and Andrzejak A. Using virtualization to improve software rejuvenation. In Proceedings 6th IEEE International Symposium, on Network Computing and Applications, July 12-14, 2007, pp. 33-44.

[8] Software Rejuvenation. Department of Electrical and Computer Engineering, Duke University, http://www.software-rejuvenation.com/

[9] Thein T. and Park J. S., Availability Analysis of Application Servers Using Software Rejuvenation and Virtualization, J. Computer Science and Technology, 24(2), 2009

[10] The dictionary of Princeton University: http://wordnet.Princeton.edu/ [consulted May, 2010]

[11] Trivedi K. S.. SHARPE 2002: Symbolic Hierarchical Automated Reliability and Performance Evaluator. In Proceedings of International Conference on Dependable Systems and Networks, 2002, pp. 544.

[12] Trivedi K. S., Vaidyanathan K., and Goseva-Popstojanova K. Modeling and analysis of software aging and rejuvenation. In SS ’00: Proceedings of the 33rd Annual Simulation Symposium, page 270, Washington, DC, USA, 2000. IEEE Computer Society