ISSN 2047-3338

Stochastic Reward Nets Model for Time based Software Rejuvenation in Virtualized Environment

Aye Myat Myat Paing and Ni Lar Thein
University of Computer Studies, Yangon
aing.ayemyat@gmail.com

***Abstract*—Modern business has a 24x7 non-stop running and the availability of the business continuity for everything, from everywhere, at all time is a growing requirement. System outages are more often due to software fault, than hardware fault. Software availability is one of the weakest links in system availability. Several studies have reported that one of the causes of unplanned software outages is the software aging phenomenon. Server virtualization is becoming so reliable and cost effective solutions for the availability of the business continuity. In virtualization environments, the hypervisor or virtual machine monitor (VMM) itself or virtual machines (VM) can fail with software failure. Software rejuvenation is one of the promising techniques assuring high availability of server virtualized system. To prevent system failures caused by software aging in both VM and VMM, software rejuvenation can be applied. The work presented in this paper aims to offer high availability against software aging of virtualized server system by providing both VM clustering software rejuvenation and VM migration based software rejuvenation analytic model using stochastic reward nets (SRN) for time based rejuvenation policy. Numerical examples are presented to illustrate the applicability of the model. The numerical derivation results are validated with the evaluation results through SHARPE tool.**

***Index Terms*—Availability, Clustering, Software Aging, Software Rejuvenation, Stochastic Reward Nets and Virtualization**

I. INTRODUCTION

AVAILABILITY has long been a critical issue for online computer systems whose failure can crash business processes [1]. In some cases, a lack of the higher level of availability in business continuity, especially for a prolonged period of time, can threaten the very existence of business. Virtualization technology provides the cost effective and reliable solution in business continuity strategies. One method of virtualizing the hardware resources of a computer involves using a layer of software, called the Virtual Machine Monitor (VMM), to provide the illusion of real hardware for multiple virtual machines (VMs). Inside each VM, the operating system (often called the guest OS) and applications run on the VM's own virtual resources such as virtual CPU, virtual network card, virtual RAM, and virtual disks [2]. Virtualization has proved as a successful tool for management of complex IT-environments and it is emerging as a technique to increase system reliability [3]. The majority of today's high availability (HA) clusters is based on real physical hardware and virtualization is coming more and more popular nowadays, one has to think about possible combinations of virtualization and high availability clustering [4]. As business becomes increasingly dependent on information and computing technology, continuous availability is a universal concern. Failures of computer systems are more often due to software faults than due to hardware faults.

The state of software degrades with time is known as software aging. The causes of process aging are memory leaking, unreleased file locks, file descriptor leaking, data corruption in the operating environment of system resources, etc. Process aging will affect the performance of the application and eventually cause the application to fail. This phenomenon becomes critical in 24x7 applications [5]. The most natural procedure to counteract such software aging is to apply the well-known technique of software rejuvenation. Software rejuvenation techniques [5], [6] become necessary to mitigate or avoid the consequences of software aging. Recently, software aging of virtual machine monitors (VMMs) is becoming critical because many VMs run on top of a VMM in one machine consolidating multiple servers and aging of the VMM directly affects all the VMs. To postpone system failures caused by software aging in VMM, software rejuvenation can be applied to VMM. Software rejuvenation is a proactive recovery method that clears aging status by restarting or resetting the software execution environment [7], [8].

In this paper, we are mainly focused on the unplanned software outages due to software aging problem. We present a comprehensive availability model for both VM clustering software rejuvenation model and VM migration based software rejuvenation model. Our availability model captures software aging states of VM and VMM as well as their failures caused by aging. Using analytical modeling as a stochastic reward nets (SRN), we analyze multiple design choices when a single physical server with only VM rejuvenation and dual physical servers with both VM and VMM rejuvenation are used to host multiple virtual machines. Analysis results are included to show the performance of the proposed me-

thod. To evaluate the models through both analytic analysis and SHARPE tool simulation is presented.

The organization of this paper is as follows. In Section II we discuss the related work. The proposed system in order to offer high availability is described in section III. Our proposed two scenarios of VM clustering and VM migration based time-based rejuvenation model are presented in Section IV and V. Finally we conclude our paper in Section VI.

II. Related Work

In this section we describe literature review which related to our work. High availability and business continuity are critical; without them, adequate data protection is impossible and the problems that arise can be very serious.

Some studies incorporated software rejuvenation for VM into availability model and computed the down time cost or steady state availability of the system [4], [9], [10]. Time based rejuvenation for VM was modeled as a continuous time Markov chain for virtualized system. Different configurations of the consolidated servers in the form of one physical and two physical servers in the scheme of hot standby are considered [4]. A new technique for fast rejuvenation of Virtual Machine Monitors (VMMs) called the warm-VM reboot was proposed in [8]. 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.

VMM failure and VMM rejuvenation were also taken into account in the availability model for virtualized system using stochastic reward nets [9]. In [10], they provide stochastic process based models to evaluate availability of the system in case of without virtualization technology and in case when virtualization and software rejuvenation are used.

In [11] presented and analyzed three software rejuvenation policies for an operational software system with multiple degradations using stochastic Petri nets and continuous time markov chains are used to describe the analytic models.

A comprehensive availability model for a server virtualized system with time based rejuvenations for VM and VMM is presented. As a result of the comparison, Migrate VM rejuvenation is the best rejuvenation method for VMM in terms of steady state availability as long as live VM migration is fast enough [12]. Salfner et al. [13] presented and analyzed a coloured stochastic Petri net model of a redundant fault-tolerant system. They have simulated the Petri net model for different levels of utilization and have computed service unavailability in the different model configurations.

III. Proposed System

In this section we describe our proposal to offer high availability mechanism using time-based software rejuvenation methodology. First we present the ways of using virtualization to improve software rejuvenation for addressing the software aging issue. In the proposed system, virtualization technology and software rejuvenation can be used to prolong the availability of the services. Since a high availability clustering system generally provides an active/backup node structure by connecting two servers. The backup node monitors the active node while the active node occupies resources and provides services based on the occupied resources.

Clustering supports two or more servers running duplicate VMs. Failover technologies that allow a VM on a troubled server to migrate seamlessly to an available server are also available. Failover technologies also allow a failed VM to load from a storage snapshot and start up on another server. To counteract the software and hardware failure, the rejuvenation schedules for VM and VMM need to determined in proper way for the VM availability, since VMM rejuvenation effects VMs running on the VMM. The following two scenarios are studied in this paper.

- VM clustering software rejuvenation (2VMs1PM)
- VM migration based software rejuvenation (2VMs2PMs)

To describe the behavior of these two scenarios, a stochastic reward nets (SRN) model [14] is presented. The assumptions for the proposed models are the time between software failures caused by aging-related faults is assumed to be hypo-exponentially distributed and failure detection time and switchover, migration, repair, rejuvenation, time are assumed to be exponentially distributed.

The applicability of the proposed model and solution methodology through numerical examples are illustrated. The exact model transition firing rates for the model are not known, a good estimate value for a range of model transition firing rates is assumed. For this purpose, we perform experiments using the following failure profile mentioned in Table I.

TABLE I. Transition Firing Rates

Transitions	**Firing Rates (h^{-1})**
$T_{interval}$	1time/a day
$1/T_{sw}$	3mins
T_d	1time/3 days
T_{h_fp}	1time/a month
$T_{h\text{-}interval}$	1time/a week
$1/T_{r1}$, $1/T_{r2}$,$1/T_{v\text{-}r}$	1 min
$1/T_{h\text{-}r}$	2 mins
$1/T_{rst}$	30 secs
$1/T_{mig}$	1 sec
$1/T_{v\text{-}repair}$, $1/T_{repair}$	30 mins
$1/T_{h\text{-}repair}$	1 hour
T_{fp}, T_{fp2} , T_{fp1}	2 times/ month

IV. VM Clustering Software Rejuvenation (2VMs1PM)

This scenario builds an HA cluster between two or more virtual machines on a single physical machine (1PM) as

shown in Fig. 1. Physical machine hosts the virtual machines. One monitoring VM and other operational VMs on the top of the virtualization layer (VMM) are created. The main application server will be running on one VM and the remaining VM will be used for standby server. Some software modules that will be responsible for the detection of software aging are installed in the monitoring VM. The monitoring VM will trigger a rejuvenation operation. If the active VM is about to be rejuvenated, standby VM will be started and then all the new requests and sessions are switched from the active VM to standby VM. So the physical machine itself is a SPOF (single point of failure).

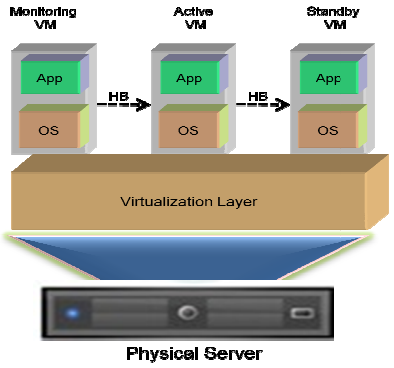

Fig. 1. Active-Standby VMs on single physical machine

A. *Proposed SRN model for VM clustering software rejuvenation*

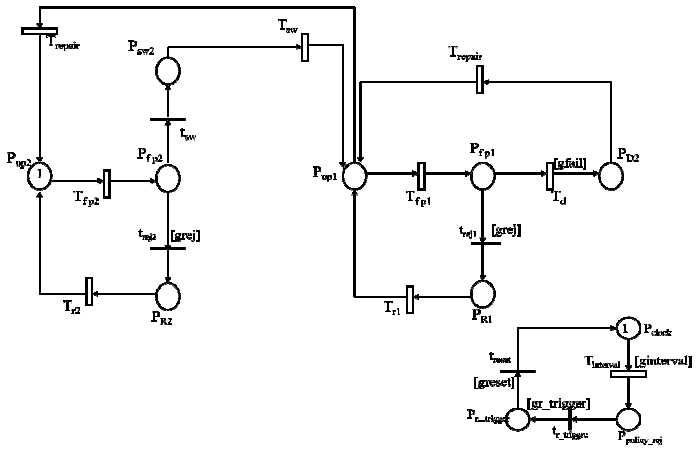

Fig. 2. SRN model of Time-based Rejuvenation for 2VMs1PM

Places		**Description**
P_{upi}	:	Healthy state of i^{th} virtual machines
P_{fpi}	:	Failure Probably state of i^{th} virtual machines
P_{D2}	:	Failure state
P_{Ri}	:	Rejuvenation state of i^{th} virtual machines
P_{sw2}	:	VM switch over state
P_{clock}	:	Rejuvenation Interval state of i^{th} virtual machines
P_{policy_rej}	:	Rejuvenation policy state of i^{th} virtual machines
$P_{r\ trigger}$	:	Rejuvenation state of i^{th} virtual machines (i= 1,2 where i= number of virtual machines)

TABLE II. GUARD FUNCTION FOR 2VMS1PM

ginterval	:	$((\#(P_{fp2})==1) \parallel (\#(P_{fp1})==1))$
grej	:	$(\#(P_{r\text{-}trigger})==1)$
gr_trigger	:	$(\#(P_{policy\text{-}rej})==1)$
greset	:	$((\#(P_{R2})==1)$ or $(\#(P_{R1})==1))$
gfail	:	$((\#(P_{fp1})==1))$

The SRN model for time based rejuvenation policy in single physical server is shown in figure 2. It consists of active-standby virtual machine servers. Both VMs are "healthy" working states, indicated by a token in place P_{up2}. As time progresses, active VM eventually transits to failure probably states in place P_{fpi} through the transition T_{fpi}. The VM is still operation in this state. But VM needs to switch over when token is placed in P_{fpi} and before rejuvenation. When transition t_{sw} is enabled, the operation of active VM is switched to standby VM and a token is moved to a place P_{sw2}. After that operation will be restarted on standby VM through T_{sw}. In this model, rejuvenation interval is determined by using a clock with guard function ginterval and there are tokens in the place P_{clock} and P_{fpi}. If there is a token in place $P_{r\text{-}trigger}$, and there is VM to be rejuvenated (a token is placed in P_{fpi}), immediate transitions t_{reji} is enabled through guard function grej. After VM has been rejuvenated, it goes back to healthy state with transition T_{ri}. Also immediate transition t_{reset} is enabled and a token is moved to P_{clock}. Standby VM has the same rejuvenation policy. When there is a physical server is crash (i.e., there is a token is placed in place P_{D2} by using transition T_d with guard function gfail. From a full system outage, the system can be repaired through the transition T_{repair} and all VMs are in healthy state in place P_{up2}.

B. *Reachability Analysis*

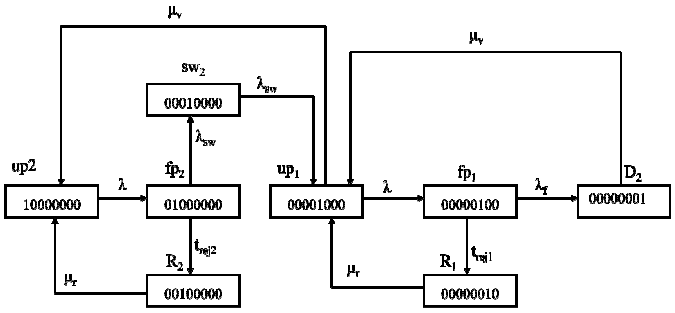

Fig. 3. Reachability Graph for the proposed SRN model

In this section, we construct the reachability graph for the proposed model is shown in figure 3. Let 8 tuples (P_{up2}, P_{fp2}, P_{R2}, P_{sw2}, P_{up1}, P_{fp1}, P_{R1} and P_{D2}) denote the marking with P_x =1, if a token is presented in place $P_{x,}$ and zero otherwise.

A marking is reachable from another marking if there exists a sequence of transition firings starting from the original marking that result in the new marking. The reachability set of a SRN is the set of all markings that are reachable from its initial marking. The marking process is mapped into a continuous time Markov chain (CTMC) with state space isomorphic to the reachability graph of the SRN.

Markings (states) enabling immediate transitions are passed through 0 time and are called vanishing. The resulting reachability graph, referred to as the extended reachability graph and there need to eliminate the vanishing markings to obtain the underlying CTMC as shown in figure 4.

Fig. 4 illustrates the extended reachability graph with squares representing the markings and arcs representing possible transition between the markings. Let λ, λ_{sw}, μ_r, and μ_v be the transition rates associated with T_{fpi}, T_{sw}, T_{ri} and T_{repair} respectively. By mapping through actions to this extended reachability graph with stochastic process, we get mathematical steady-state solution of the chain.

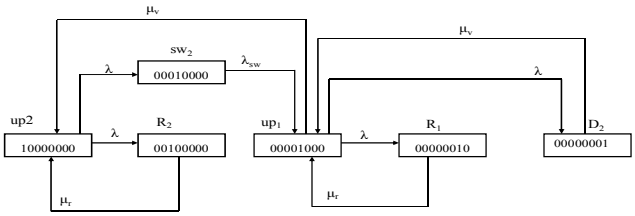

Fig. 4. Extended Reachability Graph for the proposed SRN model

We may compute the steady-state probability by first writing down the steady-state balance equations of figure as follows.

$$\mu_v P_{up_1} + \mu_r P_{R_2} = \boldsymbol{\lambda} P_{up_2} + \boldsymbol{\lambda} P_{up_2} \quad (1)$$

$$\boldsymbol{\lambda P_{up_2}} = \boldsymbol{\lambda_{sw} P_{sw_2}} \quad (2)$$

$$\boldsymbol{\lambda} P_{up_2} = \mu_r P_{R_2} \quad (3)$$

$$\boldsymbol{\lambda_{sw}} P_{sw_2} + \mu_v P_{D_2} + \mu_r P_{R_1} = \mu_v P_{up_1} + \boldsymbol{\lambda} P_{up_1} + \boldsymbol{\lambda} P_{R_1} \quad (4)$$

$$\boldsymbol{\lambda} P_{up_1} = \mu_r P_{R_1} \quad (5)$$

$$\boldsymbol{\lambda} P_{up_1} = \mu_v P_{D_2} \quad (6)$$

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

$$P_{up_2} + P_{sw2} + P_{up1} + P_{R_2} + P_{R_1} + P_{D2} = 1 \quad (7)$$

Combining the above-mentioned balance equations with the conservation equation, and solving these simultaneous equations, we acquire the closed-form solution for the system.

$$P_{R_2} = \frac{\boldsymbol{\lambda}}{\mu_r} P_{up_2} \quad (8)$$

$$P_{sw_2} = \frac{\boldsymbol{\lambda}}{\boldsymbol{\lambda_{sw}}} P_{up_2} \quad (9)$$

$$P_{up_1} = \frac{\boldsymbol{\lambda}}{\mu_v} P_{up_2} \quad (10)$$

$$P_{R_1} = \left[\frac{\boldsymbol{\lambda}}{\mu_r}\right]\left[\frac{\boldsymbol{\lambda}}{\mu}\right] P_{up_2} \quad (11)$$

$$P_{D_2} = \left[\frac{\boldsymbol{\lambda}}{\mu_v}\right]^2 P_{up_2} \quad (12)$$

$$P_{up_2} = \left\{1 + \frac{\boldsymbol{\lambda}}{\mu_r} + \frac{\boldsymbol{\lambda}}{\boldsymbol{\lambda_{sw}}} + \frac{\boldsymbol{\lambda}}{\mu_v} + \left[\frac{\boldsymbol{\lambda}}{\mu_r}\right]\left[\frac{\boldsymbol{\lambda}}{\mu_v}\right] + \left[\frac{\boldsymbol{\lambda}}{\mu_v}\right]^2\right\}^{-1} \quad (13)$$

The meaning of the probabilities as follows:

P_{up_i}	The probability of the VM is in healthy state
P_{R_i}	The probability of the VM is in rejuvenation state
P_{sw2}	The probability of the VM in switchover state
P_{D2}	The probability of both VMs are in failure state

(i= 1,2 where i= number of VMs)

C. Availability and Downtime in Analysis

It is a probability of a system which provides the services in a given instant time. Based on this, system availability and unavailability can be computed.

In the proposed model, services are not available when both VMs are in failure state. We also define the availability of the proposed model as:

Availability = 1- Unavailability

$$Availability = 1 - P_{D2} \quad (14)$$

The expected total downtime of the application with rejuvenation in an interval of T time units is

$$Downtime(T) = T * P_{D2} \quad (15)$$

Fig. 5 illustrates the availability changes for the proposed model with different VM aging rates and different repair rates had been applied. The repair transition firing rates are assumed 1 time/30 minutes and 1 time/60 minutes. It can be

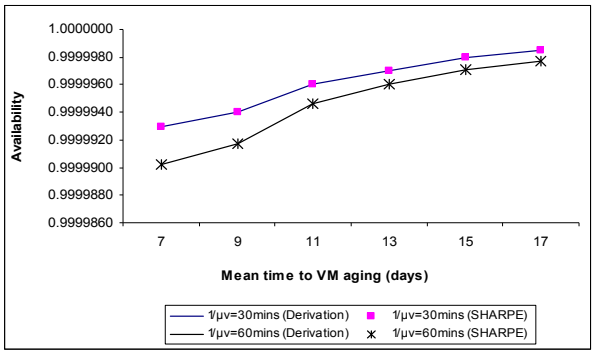

Fig. 5. Availability vs. different VM aging and different repair rates

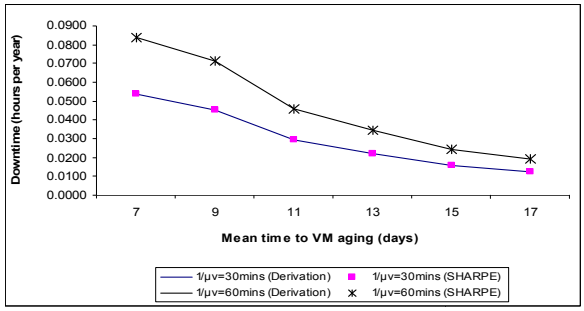

Fig. 6. Downtime vs. different VM aging and different repair rates

observed that the repair rate increases for VM, the higher availability can be achieved. Figure 6 plotted the downtime as a function of the VM aging for different repair rates. Increases in repair rates for VM lowered the system downtime.

V. VM MIGRATION BASED SOFTWARE REJUVENATION (2VMS2PMS)

In this scenario Active-standby virtualized clustering architecture is employed. An HA cluster is built between two or more virtual machines, each of them running on different physical machines (2 PMs) is shown in figure 7. The SPOF of a single physical machine is eliminated. It has Active physical server and standby physical server. Both physical servers can access shared storage. A heartbeat keep-alive system is used to monitor the health of VMs and the physical servers. At active physical server, VMs are created as monitoring VM, active VM and standby VM as well as standby physical server. Both VM and VMM time-based rejuvenation mechanism is considered in this scenario. Time based rejuvenation policy for VM is same as active-standby VMs hosted on 1PM.

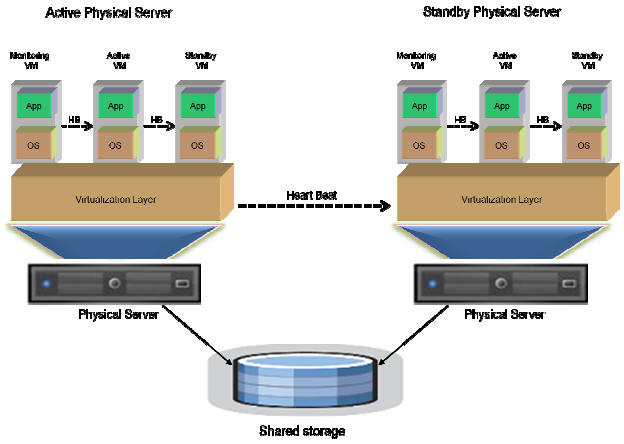

Fig. 7. Active-Standby VMs on dual physical server

A. Proposed SRN model for VM migration based software rejuvenation

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. When VMM need to be rejuvenation, the hosted VMs can move onto other physical server. It can return back to the original host after the completion of the VMM rejuvenation by live VM migration again. In the event of an active physical server outage, the virtualized recovery server at standby physical server can be activated to take over the running of the workload immediately using live migration. The down time of a VM caused by live VM migration is very small and the VM continues the execution even while the original host is down.

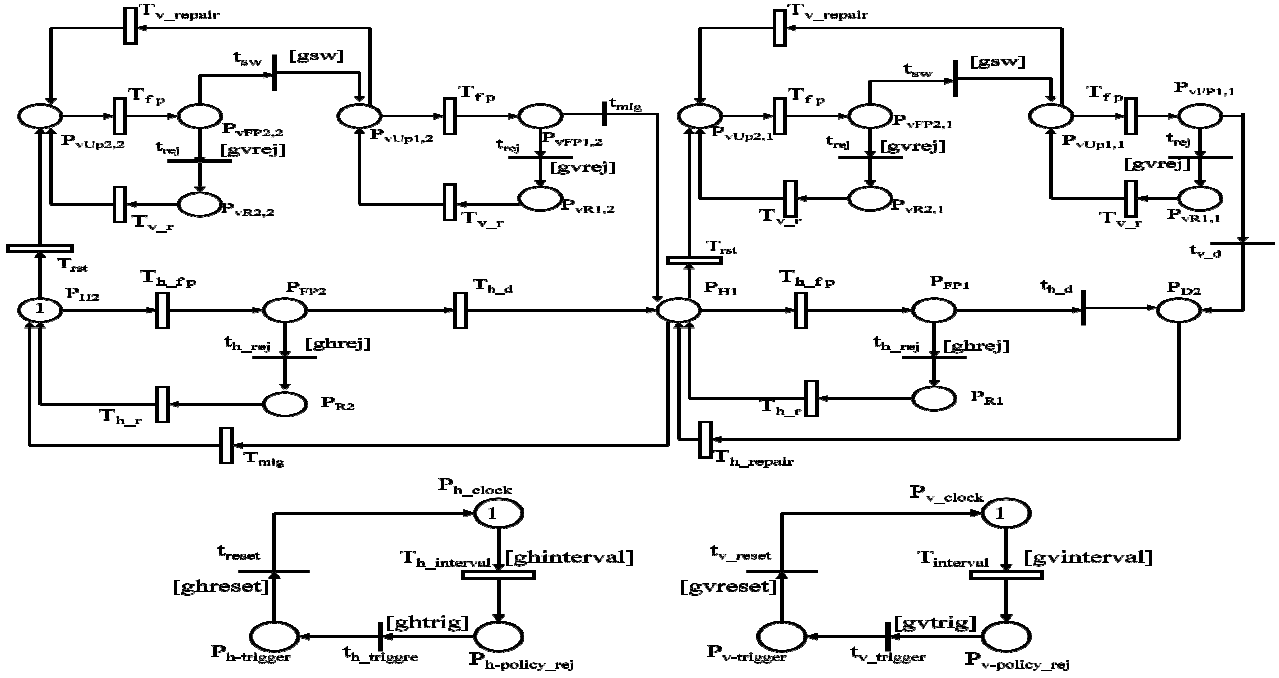

Fig. 8. SRN model of Time-based Rejuvenation for 2VMs2PMs

Places		**Description**
$P_{vUpi,j}$	:	Healthy state of i^{th} VM on j^{th} VMM
$P_{vFPi,j}$	:	Failure Probably state of i^{th} VM on j^{th} VMM
$P_{vRi,j}$	:	Rejuvenation state of i^{th} VM on j^{th} VMM
P_{Hj}	:	Healthy state of j^{th} VMM
P_{FPj}	:	Failure Probably state of j^{th} VMM
P_{Rj}	:	Rejuvenation state of j^{th} VMM
P_{D2}	:	Failure state of all VMs in both VMMs
$P_{h\text{-}clock}$	:	Rejuvenation Interval state of j^{th} VMM
$P_{h\text{-}policy\text{-}rej}$	:	Rejuvenation policy state of j^{th} VMM
$P_{h\text{-}trigger}$	:	Rejuvenation state of j^{th} VMM
$P_{v\text{-}clock}$	:	Rejuvenation Interval state of i^{th} VM
$P_{v\text{-}policy\text{-}rej}$	:	Rejuvenation policy state of i^{th} VM
$P_{v\text{-}trigger}$	:	Rejuvenation state of i^{th} VM
		i= 1,2 (number of VMs) j=1,2 (number of VMMs)

TABLE III. GUARD FUNCTION FOR TIME-BASED REJUVENATION

ghinterval	:	$((\#(P_{FP2})==1) \parallel (\#(P_{FP1})==1))$
gvinterval	:	$((\#(P_{vFP2,2})==1) \parallel (\#(P_{vFP1,2})==1) \parallel (\#(P_{vFP2,1})==1) \parallel (\#(P_{vFP1,1})==1))$
ghrej	:	$(\#(P_{h\text{-}trigger})==1)$
gvrej	:	$(\#(P_{v\text{-}trigger})==1)$
gsw	:	$((\#(P_{vFP2,2})==1)$ or $(\#(P_{vFP2,1})==1))$
gvreset	:	$((\#(P_{vR2,2})==1) \parallel (\#(P_{vR1,2})==1) \parallel (\#(P_{vR2,1})==1) \parallel (\#(P_{vR1,1})==1))$
ghreset	:	$((\#(P_{R2})==1) \parallel (\#(P_{R1})==1))$
ghtrig	:	$(\#(P_{h\text{-}policy\text{-}rej})==1)$
gvtrig	:	$(\#(P_{v\text{-}polivy\text{-}rej})==1)$

The SRN model for time based rejuvenation policy in dual physical servers with both VMs and VMMs is shown in figure 8. It consists of two physical servers. Both physical servers have active-standby VMs. Both physical servers are "healthy" working state, indicated by a token in place P_{H2}.

As time progresses, each VMM eventually transits to failure probably states in place P_{FPi} through the transition T_{fp}. The VMM is still operation in this state. But VMs need to migrate when token is placed in P_{FPi} and before rejuvenation. When the immediate transition t_{mig} is enabled, the VM is migrated to another physical server and a token is moved to a place P_{H1}.

After migration the VM will be restarted on another physical server through T_{rst}. In this model, rejuvenation interval is determined by using a clock with guard function ghinterval and there are tokens in the place P_{clock} and P_{FPi}. If there is a token in place $P_{h\_trigger}$ through guard function ghtrig, and there are VMMs to be rejuvenated (a token is placed in P_{FPi}), immediate transitions t_{h_rej} is enabled through guard function grej. After VMMs have been rejuvenated, it goes back to healthy state with transition T_{h_r}.

Also immediate transition is enabled by using guard function greset and a token is moved to P_{h_clock}. When there is a system is crash (i.e., there is a token is placed in place P_{D2} by using immediate transition t_{h_d}.

From a full system outage, the system can be repaired through the transition T_{h_repair} and VM is migrated through transition T_{mig}. After that all VMMs are in healthy state in place P_{H2}. Additionally, the active-standby VMs' rejuvenation in both physical servers is considered. Both VMs are "healthy" working states, a token is placed in place $P_{vUp2,2}$. As time progresses, active VM eventually transits to failure probably states in place $P_{vFPi,j}$ through the transition T_{fp}. The VM is still operation in this state.

But VM needs to switchover when token is placed in $P_{vFPi,j}$ and before rejuvenation. When transition t_{sw} is enabled, the operation of active VM is switched to standby VM and a token is moved to a place $P_{vUp1,j}$. In this model, rejuvenation interval is determined by using a clock with guard function gvinterval and there are tokens in the place P_{clock} and $P_{vFPi,j}$. If there is a token in place $P_{v\_trigger}$, and there are VMs to be rejuvenated (a token is placed in $P_{vFPi,j}$), immediate transitions t_{rej} is enabled through guard function grej.

After VMs have been rejuvenated, it goes back to healthy state with transition T_{v_r}. Also immediate transition t_{v_reset} is enabled and a token is moved to P_{v_clock}. When there is all VMs are crash (i.e., there is a token is placed in place $P_{vFP1,1}$ by using immediate transition t_{v_d}. From VMs outage, the system can be repaired through the transition T_{v_repair} and all VMs are in healthy state in place $P_{vUpi,j}$.

B. Reachability Analysis

In this section, we construct the reachability graph for the proposed model. Let 19 tuples (P_{H2}, P_{FP2}, P_{H1}, P_{FP1}, P_{D2}, P_{R2}, P_{R1}, $P_{vUp2,2}$, $P_{vFP2,2}$, $P_{vR2,2}$, $P_{vUp1,2}$, $P_{vFP1,2}$, $P_{vR1,2}$, $P_{vUp2,1}$, $P_{vFP2,1}$, $P_{vR2,1}$, $P_{vUp1,1}$, $P_{vFP1,1}$ and $P_{vR1,1}$,) denote the marking with P_x =1, if a token is presented in place $P_{x,}$ and zero otherwise.

Markings (states) enabling immediate transitions are passed through 0 time and are called vanishing. The resulting reachability graph, referred to as the extended reachability graph and there need to eliminate the vanishing markings to obtain the underlying CTMC as shown in Fig. 9.

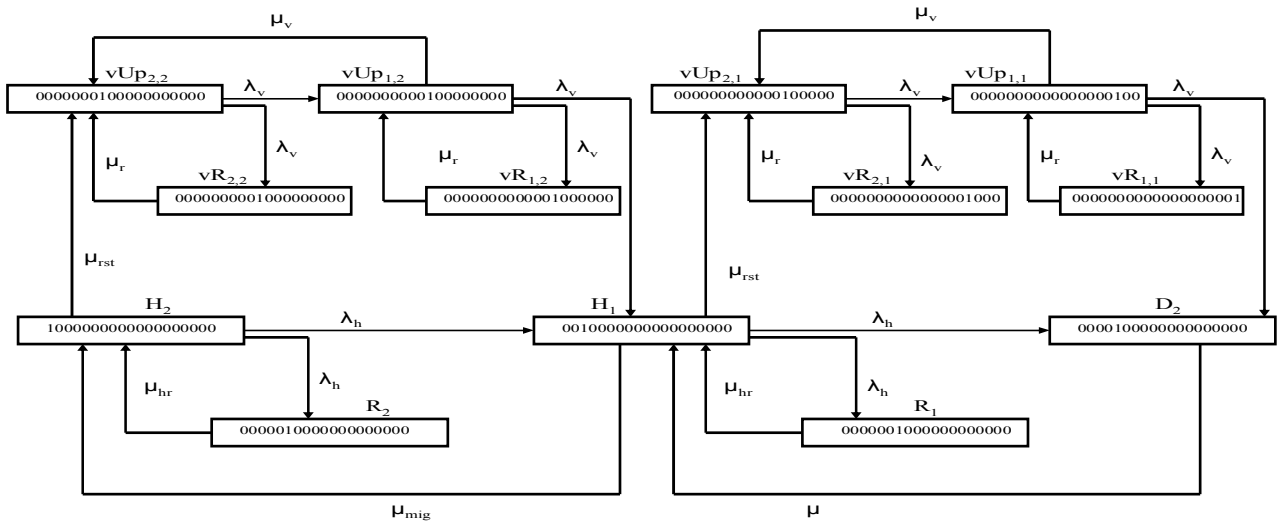

Fig. 9. Extended Reachability Graph for the proposed SRN model

This figure illustrates the extended reachability graph with squares representing the markings and arcs representing possible transition between the markings. Let λ_v, λ_h, μ_r, μ_{rst}, μ_{mig}, μ_{hr}, μ_v, and μ be the transition rates associated with T_{fp}, T_{h_fp},T_{v_r}, T_{rst}, T_{mig}, T_{h_r}, T_{v_repair},and T_{h_repair} respectively. By mapping through actions to this extended reachability graph with stochastic process, we get mathematical steady-state solution of the chain.

We may compute the steady-state probability by first writing down the steady-state balance equations of figure as follows.

$$\mu_{mig}P_{H_1} + \mu_{hr}P_{R_2} = \mu_{rst}P_{H_2} + \lambda_h P_{H_2} + \lambda_h P_{H_2} \tag{16}$$

$$\lambda_h P_{H_2} + \lambda_v P_{Up_{1,1}} + \mu P_{D_2} + \mu_{hr}P_{R_1} = \mu_{mig}P_{H_1} + \lambda_h P_{H_1} + \lambda_h P_{H_1} + \mu_{rst}P_{H_1} \tag{17}$$

$$\lambda_h P_{H_1} + \lambda_v P_{Up_{1,2}} = \mu P_{D_2} \tag{18}$$

$$\lambda_h P_{H_2} = \mu_{hr}P_{R_2} \tag{19}$$

$$\lambda_h P_{H_1} = \mu_{hr}P_{R_1} \tag{20}$$

$$\mu_{rst}P_{H_2} + \mu_r P_{vR_{2,1}} + \mu_v P_{vUp_{1,1}} = \lambda_v P_{vUp_{2,1}} + \lambda_v P_{vUp_{2,1}} \tag{21}$$

$$\lambda_v P_{vUp_{2,1}} + \mu_r P_{vR_{1,1}} = \lambda_v P_{vUp_{1,1}} + \lambda_v P_{vUp_{1,1}} + \mu_v P_{vUp_{1,1}} \tag{22}$$

$$\lambda_v P_{vUp_{1,1}} = \mu_r P_{vR_{1,1}} \tag{23}$$

$$\lambda_v P_{vUp_{2,1}} = \mu_r P_{vR_{2,1}} \tag{24}$$

$$\mu_{rst}P_{H_1} + \mu_r P_{vR_{2,2}} + \mu_v P_{vUp_{1,2}} = \lambda_v P_{vUp_{2,2}} + \lambda_v P_{vUp_{2,2}} \tag{25}$$

$$\lambda_v P_{vUp_{2,2}} + \mu_r P_{vR_{1,2}} = \mu_v P_{vUp_{1,2}} \tag{26}$$

$$\lambda_v P_{vUp_{2,2}} + \mu_r P_{vR_{1,2}} = \mu_v P_{vUp_{1,2}} + \lambda_v P_{vUp_{1,2}} + \lambda_v P_{vUp_{1,2}} \tag{27}$$

$$\lambda_v P_{vUp_{2,2}} = \mu_r P_{vR_{2,2}} \tag{28}$$

$$\lambda_v P_{vUp_{1,2}} = \mu_r P_{vR_{1,2}} \tag{29}$$

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

$$\sum_{i=1}^{2}\sum_{j=1}^{2} P_{vUp_{i,j}} + \sum_{i=1}^{2}\sum_{j=1}^{2} P_{vR_{i,j}} + \sum_{j=1}^{2} P_{H_j} + \sum_{j=1}^{2} P_{R_j} + P_{D_2} = 1 \tag{30}$$

Combining the above-mentioned balance equations with the conservation equation, and solving these simultaneous equations, we acquire the closed-form solution for the system.

$$P_{H_1} = \left[\frac{\lambda_h + \mu_{rst}}{\mu_{mig}}\right] P_{H_2} \tag{31}$$

$$P_{D_2} = \frac{1}{\mu}[-\lambda_h - \mu_{rst} - A + C]P_{H_2} \tag{32}$$

$$P_{R_2} = \frac{\lambda_h}{\mu_{hr}} P_{H_2} \tag{33}$$

$$P_{R_1} = \frac{\lambda_h}{\mu_{hr}}\left[\frac{\lambda_h + \mu_{rst}}{\mu_{mig}}\right]P_{H_2} \quad (34)$$

$$P_{vUp_{2,1}} = \frac{1}{\lambda_v}\left[\mu_{rst} + \frac{\mu_{rst}\,\mu_v}{\lambda_v}\right]P_{H_2} \quad (35)$$

$$P_{vUp_{1,1}} = \frac{\mu_{rst}}{\lambda_v}P_{H_2} \quad (36)$$

$$P_{vR_{1,1}} = \frac{1}{\mu}\mu_{rst}P_{H_2} \quad (37)$$

$$P_{vR_{2,1}} = \frac{1}{\mu_r}\left[\mu_{rst} + \frac{\mu_{rst}\,\mu_v}{\lambda_v}\right]P_{H_2} \quad (38)$$

$$P_{vUp_{2,2}} = \frac{1}{\lambda_v}\left\{B + \mu_v\left\{\begin{matrix}\left(-\frac{1}{\lambda_v}\right)A + \lambda_h \\ +\mu_{rst} + A - C\end{matrix}\right\}\right\}P_{H_2} \quad (39)$$

$$P_{vUp_{1,2}} = -\frac{1}{\lambda_v}[A + \lambda_h + \mu_{rst} + A - C]P_{H_2} \quad (40)$$

$$P_{vR_{1,2}} = -\frac{1}{\mu_r}[A + \lambda_h + \mu_{rst} + A - C]P_{H_2} \quad (41)$$

$$P_{vR_{2,2}} = \frac{1}{\mu_r}\left\{B + \mu_v\left\{\begin{matrix}\left(-\frac{1}{\lambda_v}\right)A + \lambda_h \\ +\mu_{rst} + A - C\end{matrix}\right\}\right\}P_{H_2} \quad (42)$$

$$\begin{aligned}P_{H_2} &= \left\{1 + \left[\frac{\lambda_h + \mu_{rst}}{\mu_{mig}}\right] + \frac{1}{\mu}(-\lambda_h - \mu_{rst} - A + C)\right. \\ &+ \frac{\lambda_h}{\mu_{hr}} + \frac{\lambda_h}{\mu_{hr}}\left[\frac{\lambda_h + \mu_{rst}}{\mu_{mig}}\right] + \frac{1}{\lambda_v}\left[\mu_{rst} + \frac{\mu_{rst}\,\mu_v}{\lambda_v}\right] \\ &+ \frac{\mu_{rst}}{\lambda_v} + \frac{1}{\mu}\mu_{rst} + \frac{1}{\mu_r}\left[\mu_{rst} + \frac{\mu_{rst}\,\mu_v}{\lambda_v}\right] \\ &+ -\frac{1}{\lambda_v}(A + \lambda_h + \mu_{rst} + A - C) \\ &+ -\frac{1}{\mu_r}(A + \lambda_h + \mu_{rst} + A - C) \\ &+ \frac{1}{\mu_r}\left\{B + \mu_v\left\{\begin{matrix}\left(-\frac{1}{\lambda_v}\right)A \\ +\lambda_h + \mu_{rst} \\ +A - C\end{matrix}\right\}\right\} \\ &\left.+ \frac{1}{\lambda_v}\left\{B + \mu_v\left\{\left(-\frac{1}{\lambda_v}\right)A + \lambda_h + \mu_{rst}\right\}\right\}\right\}^{-1}\end{aligned} \quad (43)$$

Where, $A = \lambda_h\left[\frac{\lambda_h + \mu_{rst}}{\mu_{mig}}\right]$

$$B = \mu_{rst}\left[\frac{\lambda_h + \mu_{rst}}{\mu_{mig}}\right]$$

$$C = \left(2\lambda_h + \mu_{rst} + \mu_{mig}\right)\left[\frac{\lambda_h + \mu_{rst}}{\mu_{mig}}\right]$$

The meaning of the probabilities as follows:

$P_{vUp_{i,j}}$	The probability of the VM is in healthy state
$P_{vR_{i,j}}$	The probability of the VM is in rejuvenation state
P_{H_j}	The probability of the VMM is in healthy state
P_{R_j}	The probability of the VMM is in rejuvenation state
P_{D2}	The probability of both VMs and VMMs are in failure state

(i= 1,2 where i= number of VMs)
(j=1,2, where j= number of physical machines)

C. Availability and Downtime in Analysis

In the proposed model, services are not available when both VMs and VMMs are in failure state.

We also define the availability of the proposed model as:

Availability = 1- Unavailability

$$Availability = 1 - P_{D2} \quad (44)$$

The expected total downtime of the application with rejuvenation in an interval of T time units is

$$Downtime(T) = T * P_{D2} \quad (45)$$

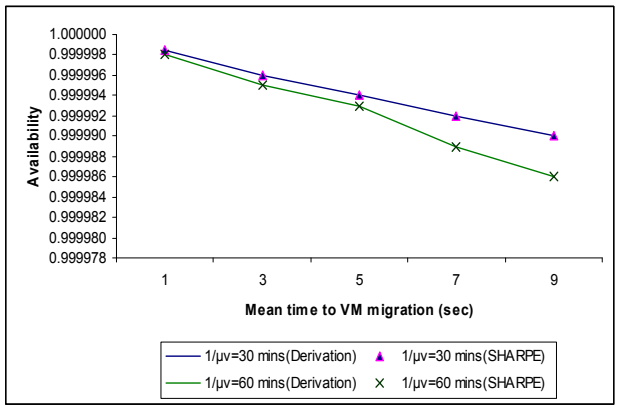

Fig. 10. Availability vs. different VM migration and different repair rates

Fig. 10 illustrates the availability changes for the proposed model with different VM migration rates and different VM repair rates had been applied. The mean time to repair transitions is assumed 30 minutes and 60 minutes. The lower mean time to VM migration, the higher availability of the proposed model can be achieved. Therefore, the availability is dependent on the VM migration transition rate.

Fig. 11 plotted the downtime as a function of the VM migration for different VM repair rates. Increases in migration rates for VM lowered the system downtime.

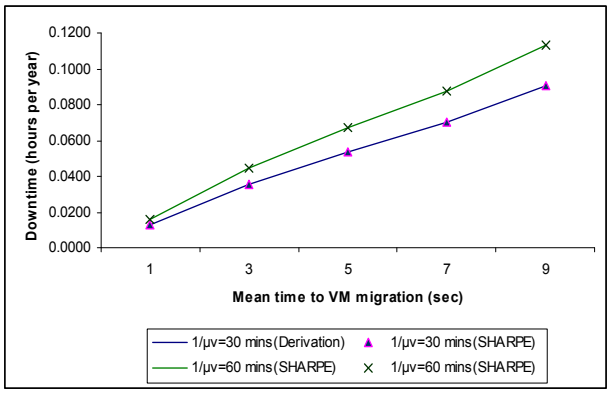

Fig. 11. Downtime vs. different VM migration and different repair rates

We compare the steady-state availability of time-base rejuvenation for only VM in single physical server and VMM and VM in dual physical servers as shown in figure 12. The downtime as a function of VM aging rates with different rejuvenation policy is plotted in Fig. 13.

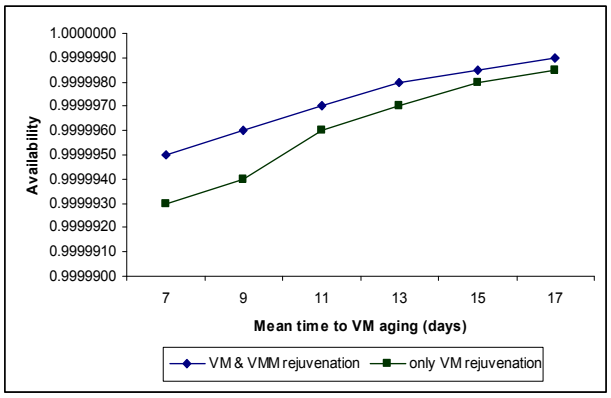

Fig. 12. Availability vs. different VM aging

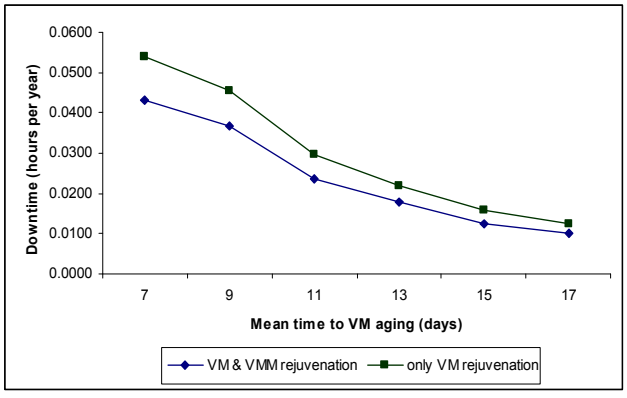

Fig. 13. Downtime vs. different VM aging

SHARPE [15] is well known package in the field of reliability and performability. It is possible to use different kinds of models hierarchically for different physical or abstract levels of the system and to use different kinds of models to validate each other's results. So for our models' validity we use this tool.

From the result, it is apparent that the proposed model is a high availability in order to integrate virtualization technology, clustering and software rejuvenation mechanism for both VM and VMM. According to the figures it is found that the derivation results and SHARPE tool simulation results are the same.

VI. Conclusion

In the world that requires 24x7 business continuity; high availability has reached a new level through virtualization. In this paper we presented possible combinations of server virtualization technology, high availability cluster and time-based software rejuvenation. This approach has been designed to use over any server or service. Active-Standby VMs, single physical machine setup is a very comfortable way without requiring much hardware. Active-Standby VMs, Active-Standby physical machines setup can eliminate the single point of failure. Moreover, it is reliable solution rather than traditional solution. A stochastic reward nets models for analyzing availability of the proposed virtualized server system with time-based rejuvenation is presented. We have described comprehensive availability models for both VM clustering rejuvenation and VM migration based rejuvenation and have shown some numerical results. The experiment results with the evaluation results through SHARPE are validated. It is found that the derivation results and SHARPE results are same.

References

[1] E. Vargas, "High Availability Fundamentals", Sun Blueprints Series, 2000. (http:// www.sun.com/blueprint)

[2] H. V. Ramasamy and M. Schunter, "Architecting Dependable Systems Using Virtualization." OnlineAvailable: www.opentc.net/publications/OTC_Architecting_Dependable_Systems.pdf

[3] Nagarajan, F. Mueller, C. S.Engelmann, "Proactive fault tolerance for HPC with Xen Virtualization." In: ICS07, 2007

[4] T. Thein, S. Chi and J. Park, Availability Modeling and Analysis on Virtualized Clustering with Rejuvenation, International Journal of Computer Science and Network Security, vol.8, no. 9, pp.72-80, 2008.

[5] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton, Software rejuvenation: Analysis, module and applications, In processing of International Symposium on Fault Tolerant Computing (FTCS 1995), pp. 381-390, 1995.

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

[7] K. Kourai and S. Chiba, Fast software rejuvenation of virtual machine monitor, In IEEE Transition on dependable and secure computing, 2010.

[8] K. Kourai and S. Chiba, A fast rejuvenation technique for server consolidation with virtual machines, In proceeding of International Conference on Dependable Systems and Networks (DSN 2007), pp. 245-255, 2007.

[9] Rezaei and M. Sharifi, Rejuvenation high available virtualized systems, In proceedings of 5th International Conference on Availability, Reliability and Security (ARES2010), 2010.

[10] T. Thein, S. Chi, J. Park, "Improving Fault Tolerance by Virtualization and Software Rejuvenation", In Proceedings of the Second Asia International Conference on Modeling & Simulation, Kuala Lumpur, Malaysia, pp. 855–860, 2008.

[11] X. Du, D. Hou, X. Zhong, Y. Qi, Ying Chen, "Modeling and Performance Analysis of Software Rejuvenation Policies for Multiple Degradation Systems", the 33rd Annual IEEE International Computer Software and Applications Conference, 2009.

[12] F. Machida, D. Kim, and K. S. Trivedi, Modeling and Analysis of Software Rejuvenation in a Server Virtualized System.In Proc. of the 2rd International Workshop on Software Aging and Rejuvenation,(WoSAR2010), 2010.

[13] F. Salfner, K. Wolter, "A Petri Net Model for Service Availability in Redundancy Computing Systems", In proceeding of the 2009 Winter Simulation Conference, IEEE, 2009.

[14] G. Ciardo, J.K. Muppala, and K.S. Trivedi, SPNP: Stochastic Petri Net Package, In Proc. Int'l Workshop on Petri Nets and Performance Models, pp.142-151, 1989.

[15] K. S. Trivedi, SHARPE 2002: Symbolic Hierarchical Automated Reliability and Performance Evaluator. In Proc. Int. Conference on Dependable Systems and Networks, pp. 544, 2002.