High Availability Solution for Virtualized Local Disaster Recovery

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

Abstract

Disaster recovery (DR) is an important element of the complex information technology (IT) systems. The availability of the IT for everything, from everywhere, at all time is a growing requirement. Effective IT strategies need to have both high availability (HA) and disaster recovery (DR). Nowadays, virtualized platforms have become the most popular option to deploy complex enough services. Software availability is one of the weakest links in system availability. Web servers have continuous execution of long duration and with rather varied workloads. Such characteristics make them potential candidates for a degenerative phenomenon called software aging. The work presented in this paper aims to offer the high availability solution against software aging of virtualized local disaster recovery (VLDR) by providing measurement based software rejuvenation. The idea behind our paper is two-fold. First, we present the framework seeks to maximize the number of services running simultaneously, while guaranteeing the resources needed by each service. Second, we estimate the time to aging-related failures and then which used as aging failure rates for measurement based software rejuvenation through a stochastic reward nets model. Finally, we perform the numerical analysis to evaluate the performance of the model.

Keywords: availability, local disaster recovery, measurement based software rejuvenation, stochastic reward nets, virtualization.

1. Introduction

The need for high availability (HA) and disaster recovery (DR) in IT environment is more importance than other sectors of enterprises [6]. The major reality of high availability (HA) cluster 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 [7]. The concern of disaster recovery, virtualization and high availability often into IT department's worry box'. Server virtualization and failover technology are capabilities that provide a high level of protection while keeping cost at minimum [11]. Disaster and its recovery processes involve unplanned interruption of service. Failures of computer systems are more often due to software faults than due to hardware faults. One of the causes of unplanned software outages is the software aging phenomenon. This phenomena becomes critical in 24x7 applications. A virtualization layer also called virtual machine monitor (VMM) is a software layer that abstracts the physical resources for use by the Virtual Machines (VMs).

Recently, software aging of 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. In Proactive/Predictive rejuvenation, system metrics are continuously monitored and the rejuvenation action is triggered when a crash or system hang up due to software aging seem to approach [1]. It has been verified, to a greater extent, in those software processes that are in execution for long periods of time, being also influenced by variations in their workload. 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 [3] and VMWare [5] have implemented "live" migration of VMs that involve extremely short downtimes ranging from tens of milliseconds to a second.

This paper presents the framework which seeks to maximize the number of services running simultaneously for the VLDR. We describe to offer a high availability by using a proactive and autonomic software rejuvenation technique for Internet Services in the presence of software aging due to resource consumption. A web server is a typical long running software system which should ideally run forever. Therefore, we develop a stochastic reward net of measurement based software rejuvenation model based on estimation of time to software aging failure in web server. Analysis results are included to show the performance of the proposed method. To evaluate the models through both analytic analysis and SHARPE tool [8] simulation are presented.

The organization of this paper is as follows. In section 2 we discuss the related work. Our proposed architecture and a framework which seeks to maximize the number of services running simultaneously, is presented in section 3. The proposed estimation of the times to aging-related failures and VM and VMM measurement based rejuvenation analytic model are follow in section 4 and 5. Finally we conclude our paper in section 6.

2. Related Work

In this section we describe literature review which related to our work. The authors [4] discuss high availability and disaster recovery solutions, and describe how HA and DR solutions differ from one another and how they can be combined to provide the highest levels of resiliency for IT infrastructures.

Some studies incorporated software rejuvenation for VM into availability model and computed the down time cost or steady state availability of the system. Thein et al. [7] proposed time based rejuvenation for VMs which 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. Then the estimated times to exhaustion of the system resources are calculated. Measurement-based studies of software aging and rejuvenation on general computer systems have been carried out in previous work [2]. Vaidyanathan et al. [10] proposed a measurement-based model to estimate the rate of exhaustion of system resources both as function of time and the system workload states.

3. Proposed Architecture for VLDR

The objective of a disaster recovery is to restore the operability of systems that support mission-critical and critical processes to normal operation as quickly as possible. With disaster recovery in place, organizations can resume operations at a secondary site. Local disaster recovery is that the surviving node can support the service for a failed node in the event of localized disaster such as floods, fire or building power outages.

In this section, we describe our proposal to offer high availability mechanism based on hosted virtualized clustering architecture with multiple virtual machines (VMs) on two active standby physical machines (PMs) as shown in Figure 1.

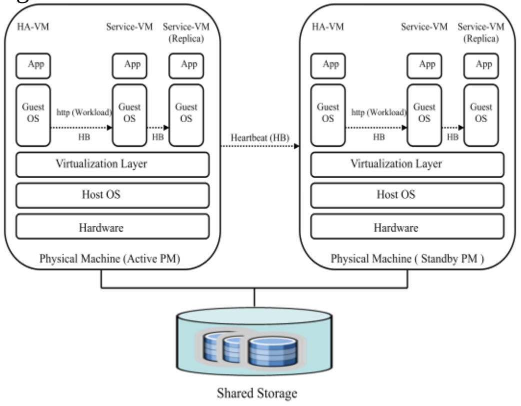


Figure 1. Hosted Virtualized Clustering Architecture

Clustering supports two or more servers running duplicate VMs. To enable live VM migration, standby physical server is connected on the same network and the disk image with active physical server. A heartbeat keep-alive system is used to monitor the health of the nodes between them.

We create one high availability (HA-VM) and other operational VMs such as service VM and replica of service VM on the top of the virtualization layer. The HA-VM contains two service providers, HA-LB and AgPreditor. ServiceVM implements as web server and also a monitoring agent, collecting the VM resource status like CPU, Memory, and number of threads or number of connections. These metrics are used to detect the software aging phenomenon and estimate the service time to crash. This estimation is conducted by AgPreditor and to estimate the time to crash due to complex software aging phenomenon. AgingPdr is running jointly with HA-LB, in their own virtual machine (HA-VM). HA-LB is load balancer which manages the software rejuvenation process. During the rejuvenation approach, a service replica (Replica VM) is needed. HA-LB automatically migrates new requests to the replica service. The on-going requests being processed by faulty service are allowed to finish. The other physical server will serve as different service and also standby physical server for disaster recovery. When the replica VM of the active physical server becomes software aging or the active physical server itself is not working, the in- flight requests and sessions are migrated to the service VM in the standby physical server.

3.1. A Framework for Resource Analysis

This framework solves the physical machine or VM failure based on service migration and to protect the software aging failure on service VM by using service replication. Service migration is conducted using the well-known technology offered by almost all virtualization manufacturers: live-migration. This technology allows us to migrate a running VM without any or minimal outage, from one physical node to another. Service Recovery is triggered due to the TTC (Time To Crash) of one or a set of VMs have violated the threshold (Time Limit) defined by the system administrators per each service or per the whole framework. The procedure of this framework is presented in table 1.

Table 1. Available Resource Checking Algorithm

```
    begin
1.  calculate the maximum number of VMs on available physical resource
2.  Select the VM that needed to migrate
3.  For all VMs on all PMs-1 do
4.      If (size of Migrated VM + currently created VMs sizes on PM <= available resource for selected PM)
5.      then VM migration to selected PM
6.  end for
7.  If (VM's TTC <= TL)
8.    For all VMs on PMs do
9.        If (size of replica VM + currently created VMs sizes on PM <= available resource for selected PM)
10.       then place the replica for that VM on selected PM
11.   end for
12. end if
    end
```

It is need to carry out these assumptions to achieve the goal. The first assumption is that every physical machine can manage a limited number of VMs, according to its resources. The VM's size is predefined and fixed on the available physical resources in our framework. We note that during a short period of time the service VM and replica VM are running simultaneously.

The second assumption has to manage the software rejuvenation approach. When a service VM crash is imminent due to software aging, a replica VM has to be created, to guarantee the availability of the service. We also describe the scenario for calculating the maximum number of services taking into account the virtual machine and the physical machine features as shown in table 2.

Table 2. Resource Capacity Description for VM and PM

Role	Memory Size
PM1,PM2	4096 MB
Host OS	512 MB
Virtualization Layer	2036 MB
Service VM	512 MB
HA-VM	512 MB
Replica VM	512 MB

In this scenario, the VM size is only computed with the memory required and based on hosted virtualization architecture. The framework is composed of two physical machines, and we deployed HA-VM separate with service VMs: then we compute the maximum numbers of VMs are as followings:

Available Memory for each PM =4096-(512+2036) (1)

Total Available Memory = PM1 +PM2 (2)

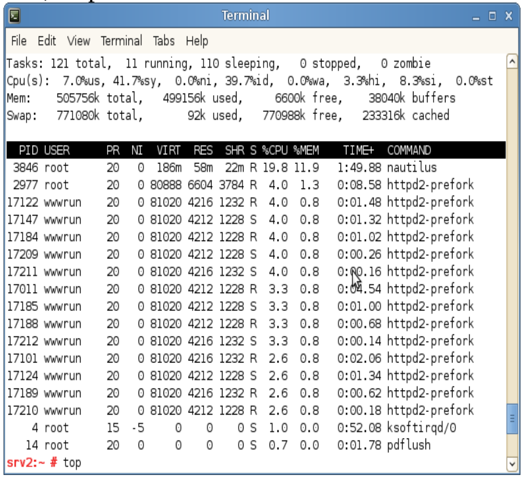
3096 MB = 1548+1548 (3)

Maximum number of VMs = Total Available Memory / size of each VM (4)

6 VMs = 3096/512 (5)

4. Estimation of Time to Software Aging Failure in Web Server

We have focused our research on software aging caused by resource exhaustion and used Apache version 2.2.10 for that case. Apache provides some features similar to software rejuvenation. In our experimental set up for estimation of time to software aging failure, we have used three virtual machines on top of VMware workstation v6.0.2 based on hosted virtualization architecture. One virtual machine was used as traffic generator and two being the web server and replica server. On the server side the httpd processes were monitored. In order to conduct the generation of workloads the httperf tool [Mosberger and Jin 1998] was used. Httperf is a web server workload generator used in our experiments to generate requests to the web server and monitor the performance of web server. Due to the capacity of the server we used 6 httpd processes, thus supporting a maximum number of 27000 simultaneous connections. The measurements provided by httperf include reply rate, response time and number of timeouts.



```
Tasks: 121 total,  11 running, 110 sleeping,   0 stopped,   0 zombie
Cpu(s):  7.0%us, 41.7%sy,  0.0%ni, 39.7%id,  0.0%wa,  3.3%hi,  8.3%si,  0.0%st
Mem:    505756k total,   499156k used,     6600k free,    38040k buffers
Swap:   771080k total,       92k used,   770988k free,   233316k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 3846 root      20   0  186m  58m  22m R 19.8 11.9   1:49.88 nautilus
 2977 root      20   0 80888 6604 3784 R  4.0  1.3   0:08.58 httpd2-prefork
17122 wwwrun    20   0 81020 4216 1232 R  4.0  0.8   0:01.48 httpd2-prefork
17147 wwwrun    20   0 81020 4212 1228 S  4.0  0.8   0:01.32 httpd2-prefork
17184 wwwrun    20   0 81020 4212 1228 R  4.0  0.8   0:01.02 httpd2-prefork
17209 wwwrun    20   0 81020 4212 1228 S  4.0  0.8   0:00.26 httpd2-prefork
17211 wwwrun    20   0 81020 4216 1232 S  4.0  0.8   0:00.16 httpd2-prefork
17011 wwwrun    20   0 81020 4212 1228 R  3.3  0.8   0:04.54 httpd2-prefork
17185 wwwrun    20   0 81020 4212 1228 S  3.3  0.8   0:01.00 httpd2-prefork
17188 wwwrun    20   0 81020 4212 1228 R  3.3  0.8   0:00.68 httpd2-prefork
17212 wwwrun    20   0 81020 4216 1232 S  3.3  0.8   0:00.14 httpd2-prefork
17101 wwwrun    20   0 81020 4216 1232 R  2.6  0.8   0:02.06 httpd2-prefork
17124 wwwrun    20   0 81020 4212 1228 S  2.6  0.8   0:01.34 httpd2-prefork
17189 wwwrun    20   0 81020 4216 1232 R  2.6  0.8   0:00.62 httpd2-prefork
17210 wwwrun    20   0 81020 4212 1228 R  2.6  0.8   0:00.18 httpd2-prefork
    4 root      15  -5     0    0    0 S  1.0  0.0   0:52.08 ksoftirqd/0
   14 root      20   0     0    0    0 S  0.7  0.0   0:01.78 pdflush
srv2:~ # top
```

Figure 2. Aging occurs in httpd processes

Figure 2 shows a memory snapshot (top command output), taken from our experiments, during a preliminary accelerated test with httpd. Initially, all httpd processes started with approximately 4000 KB. Process 2977 showed a significant increase in its memory size (RES) in comparison with the other httpd processes. In this test, we controlled the exposition of httpd processes to the factor that causes memory leaks, and intentionally exposed process (2977) more than others. We let the application run until the server failure due to memory exhaustion. Table 3 shows the pilot sample of accelerated failure times (TTF).

Table 3. Pilot Sample Time to Failure

TTF(S1)	TTF(S2)	TTF(S3)
84	34	20
86	36	21
88	37	22
93	38	23
95	39	23
97	40	24

We decided to use the pilot sample to estimate the time to software aging failure due to resource exhaustion. The time to crash could be then estimated by the following.

Let T_1, T_2, . . ., T_r be the observed times to failure so that $T_1 \leq T_2 \leq ... \leq T_r$. Specific values of these random variables are denoted by t_1, t_2 ,...

t_r. Let θ be the MTTF to be estimated and assume that components follow an exponential failure law. Since (n-r) components have not failed when the test is completed, the likelihood function is defined in the following way. Assume n is 10 and T_{r+1}, . . , T_n are the times to failure of the remaining components, whose failures will not actually be observed [9]. Then

$$L(\theta) = \frac{1}{\theta^r} exp\left[-\frac{(\sum_{i=1}^{r} t_i) + (n-r)t_r}{\theta}\right] \quad (6)$$

Let $S_{n;i} = (\sum_{i=1}^{r} t_i) + (n-r)t_r$

Then the maximum-likelihood estimator (MLE) of the mean life is

$$\hat{\theta} = \frac{S_{n;i}}{r} = \frac{(\sum_{i=1}^{r} T_i) + (n-r)T_r}{r} \quad (7)$$

Equation (7) gives the estimation of time to software aging failure as shown in Table 4.

Table 4. Estimated MTTF and Time to aging failure

Workload stress level	MTTF	Confidence Interval (95%)		Estimated time to software aging failure rate
		Lower	Upper	
S1 (200KB)	155.167	79.788	422.817	0.0064
S2 (400KB)	64.000	32.909	174.395	0.0156
S2 (600KB)	38.166	19.625	104.001	0.0262

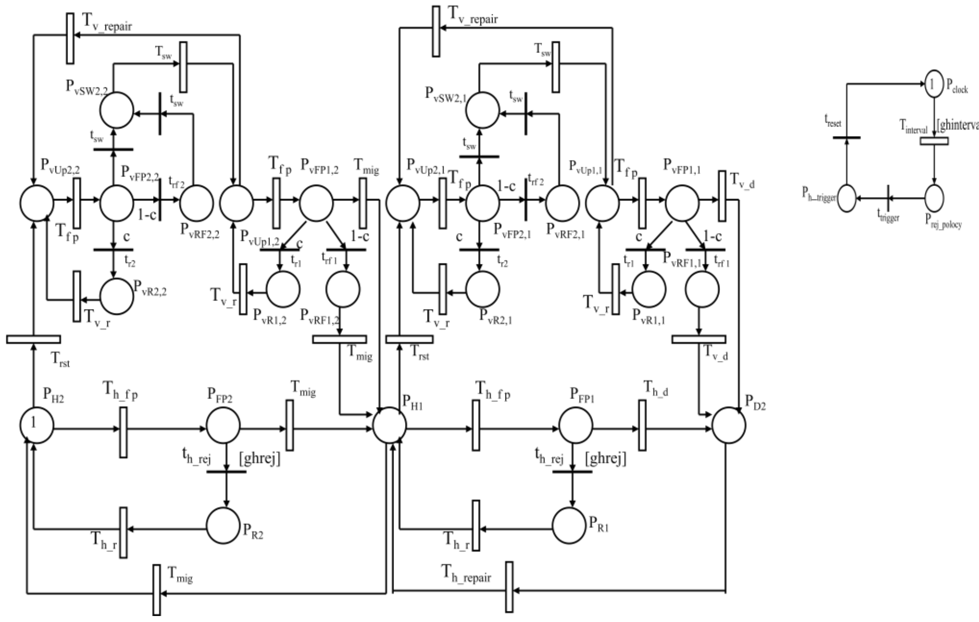
5. SRN Model for Measurement Based Software Rejuvenation

In this section, we present stochastic reward nets (SRN) model for measurement based rejuvenation policy in local disaster recovery with two physical machines and two virtual machines on each physical machine as shown in figure 3. In this policy, the rejuvenation action is determined by memory exhaustion which discussed in section 4. If the active VMM is about to be rejuvenated cause of software aging, all the VMs on problematic PM are migrated to other PMs using live migration 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 SRN model for measurement based rejuvenation policy consists of dual physical servers with both VMs and VMMs. 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. At that time the 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 immediate transition $t_{trigger}$, and there are VMMs to be rejuvenated (a token is placed in P_{FPi}), immediate transitions t_{h_rej} is enabled through guard function ghrej.

After VMMs have been rejuvenated, it goes back to healthy state with transition T_{h_r}. Also immediate transition t_{reset} is enabled by and a token is moved to P_{clock}. When there is a system is crash (i.e., there is a token is placed in place P_{D2} by using 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 service and replica 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, service 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 service VM is switched to replica VM and a token is moved to a place $P_{vSW2,j}$.



$P_{vUpi,j}$	:	Healthy state of ith VM on jth VMM	P_{Rj}	:	Rejuvenation state of jth VMM
$P_{vFPi,j}$	:	Failure Probably state of ith VM on jth VMM	P_{D2}	:	Failure state of all VMs in both VMMs
$P_{vRi,j}$	:	Rejuvenation state of ith VM on jth VMM	P_{clock}	:	Rejuvenation Interval state of jth VMM
$P_{vSWi,j}$	:	Switchover state of ith VM on jth VMM	$P_{rej-policy}$	:	Rejuvenation policy state of jth VMM
P_{Hj}	:	Healthy state of jth VMM	$P_{h-trigger}$	:	Rejuvenation state of jth VMM
P_{FPj}	:	Failure Probably state of jth VMM			i= 1,2 (number of VMs)
$P_{vRFi,j}$	:	Failure state due to estimation of software aging failure of ith VM on jth VMM			j=1,2 (number of VMMs)

Figure 3. SRN model for measurement based Rejuvenation

In this model, we assume that the resource exhaustion time can be successfully estimated with probability c, which is represented by the firing transitions t_{ri}. With probability 1-c, it will not be rejuvenated in the aged state and will eventually fail through the firing transitions t_{rfi}. After VM has been rejuvenated, it goes back to healthy state with transition T_{v_r}.

Replica VM has the same rejuvenation policy. When there is all VMs are crash (i.e., there is a token is placed in place $P_{vFP1,1}$ by using 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 $P_{vUpi,j}$.

Table 5. Guard function for VMM rejuvenation

ghinteval	:	$((\#(P_{FP2})==1) \parallel (P_{FP1})==1))$
ghrej	:	$(\#(P_{h\text{-}trigger})==1)$

A. Reachability Analysis

In this section, we construct the reachability graph for the proposed model. Let 25 tuples (P_{H2}, P_{FP2}, P_{H1}, P_{FP1}, P_{D2}, P_{R2}, P_{R1}, $P_{vUp2,2}$, $P_{vFP2,2}$, $P_{vSW2,2}$, $P_{vR2,2}$, $P_{vRF2,2}$, $P_{vUp1,2}$, $P_{vFP1,2}$, $P_{vR1,2}$,

$P_{vRF1,2}$, $P_{vUp2,1}$, $P_{vFP2,1}$, $P_{vSW2,1}$, $P_{vR2,1}$, $P_{vRF2,1}$, $P_{vUp1,1}$, $P_{vFP1,1}$, $P_{vR1,1}$ and $P_{vRF1,1}$,) denote the marking with P_x =1, if a token is presented in place P_x, and zero otherwise. 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.

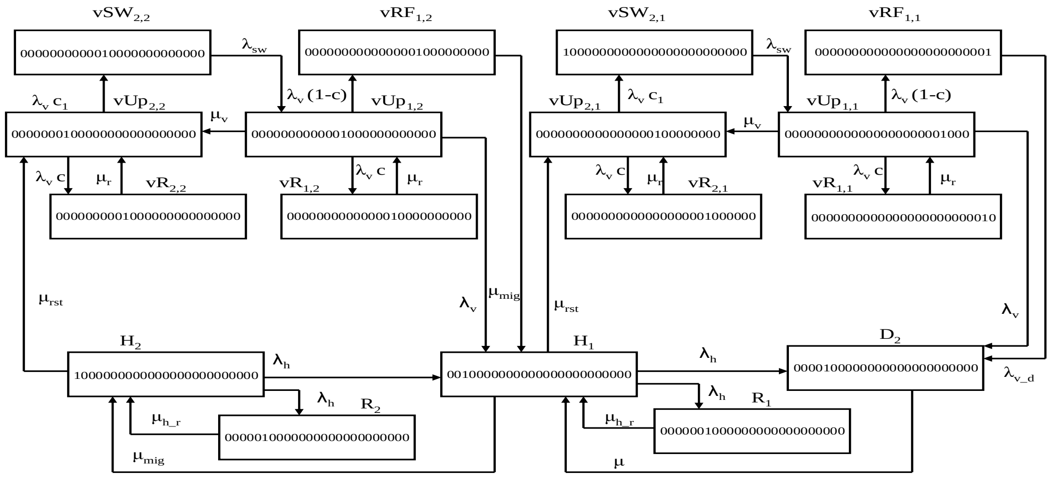


Figure 4. Extended Reachability Graph for proposed SRN model

This figure 4 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_{h_r}, μ_v, c, 1-c, c_1 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}, probability of successfully estimated for software aging failure time, probability of do not estimate for software aging failure time, probability of switchover to another service VM 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.

$$P_{H_1} = \frac{\lambda_h + \mu_{rst}}{\mu_{mig}} P_{H_2} \tag{8}$$

$$P_{D_2} = \left[\lambda_h \left(\frac{\lambda_h + \mu_{rst}}{\mu_{mig}}\right) + \lambda_{v_d}\left[\frac{\lambda_v(1-c)}{\lambda_{v_d}} A\right] + \lambda_v A\right] P_{H_2} \tag{9}$$

$$P_{R_2} = \frac{\lambda_h}{\mu_{h\_r}} P_{H_2} \tag{10}$$

$$P_{R_1} = \frac{\lambda_h}{\mu_{h\_r}}\left(\frac{\lambda_h + \mu_{rst}}{\mu_{mig}}\right) P_{H_2} \tag{11}$$

$$P_{vUp_{2,2}} = \frac{1}{\lambda_v c_1 (1-c)} B P_{H_2} \tag{12}$$

$$P_{vR_{2,2}} = \frac{\lambda_v c}{\mu_r}\left[\frac{1}{\lambda_v c_1(1-c)} B\right] P_{H_2} \tag{13}$$

$$P_{vsw_{2,2}} = \frac{1}{\lambda_{sw}} B P_{H_2} \tag{14}$$

$$P_{vUp_{1,2}} = \left(\frac{\mu_{rst}}{\lambda_v(1-c) + \lambda_v}\right) P_{H_2} \tag{15}$$

$$P_{vRF_{1,2}} = \frac{\lambda_v c}{\mu_r}\left(\frac{\mu_{rst}}{\lambda_v(1-c) + \lambda_v}\right) P_{H_2} \tag{16}$$

$$P_{vR_{1,2}} = \frac{\lambda_v(1-c)}{\mu_{mig}}\left(\frac{\mu_{rst}}{\lambda_v(1-c) + \lambda_v}\right) P_{H_2} \tag{17}$$

$$P_{vUp_{2,1}} = \frac{1}{\lambda_v c_1(1-c)}\left[\mu_v A + \left[\left(\frac{\lambda_h+\mu_{rst}}{\mu_{mig}}\right)\mu_{rst}\right]\right]P_{H_2} \qquad (18)$$

$$P_{vR_{2,1}} = \frac{\lambda_v c}{\mu_r}\left[\frac{1}{\lambda_v c_1(1-c)}\left[\frac{1}{\lambda_{sw}}\left[\mu_v A + \left[\left(\frac{\lambda_h+\mu_{rst}}{\mu_{mig}}\right)\mu_{rst}\right]\right]\right]\right]P_{H_2} \qquad (19)$$

$$P_{vsw_{2,1}} = \frac{1}{\lambda_{sw}}\left[\mu_v A + \left[\left(\frac{\lambda_h+\mu_{rst}}{\mu_{mig}}\right)\mu_{rst}\right]\right]P_{H_2} \qquad (20)$$

$$P_{vUp_{1,1}} = AP_{H_2} \qquad (21)$$

$$P_{vR_{1,1}} = \frac{\lambda_v c}{\mu_r}AP_{H_2} \qquad (22)$$

$$P_{vRF_{1,1}} = \frac{\lambda_v(1-c)}{\lambda_{v\_d}}AP_{H_2} \qquad (23)$$

$$P_{H_2} = \left\{\begin{aligned} & 1+\frac{\lambda_h+\mu_{rst}}{\mu_{mig}}+\frac{\lambda_h}{\mu_{h_r}}+\frac{1}{\lambda_{sw}}B+ \\ & \left(\frac{\mu_{rst}}{\lambda_v(1-c)+\lambda_v}\right)+A+\frac{\lambda_v c}{\mu_r}A+ \\ & \frac{\lambda_h}{\mu_{h_r}}\left(\frac{\lambda_h+\mu_{rst}}{\mu_{mig}}\right)+\frac{\lambda_v(1-c)}{\lambda_{v_d}}A+ \\ & \left[\lambda_h\left(\frac{\lambda_h+\mu_{rst}}{\mu_{mig}}\right)+\lambda_{v_d}\left[\frac{\lambda_v(1-c)}{\lambda_{v_d}}A\right] + \lambda_v A+\frac{1}{\lambda_v c_1(1-c)}B+\frac{\lambda_v c}{\mu_r}\right] \\ & \left[\frac{1}{\lambda_v c_1(1-c)}B\right]+\frac{\lambda_v c}{\mu_r} \\ & \left(\frac{\mu_{rst}}{\lambda_v(1-c)+\lambda_v}\right)+\frac{\lambda_v(1-c)}{\mu_{mig}} \\ & \left(\frac{\mu_{rst}}{\lambda_v(1-c)+\lambda_v}\right)+\frac{1}{\lambda_v c_1(1-c)} \\ & \left[\mu_v A+\left[\left(\frac{\lambda_h+\mu_{rst}}{\mu_{mig}}\right)\mu_{rst}\right]\right]+ \\ & \frac{1}{\lambda_{sw}}\left[\mu_v A+\left[\left(\frac{\lambda_h+\mu_{rst}}{\mu_{mig}}\right)\mu_{rst}\right]\right]+\frac{\lambda_v c}{\mu_r} \\ & \left[\frac{1}{\lambda_v c_1(1-c)}\left[\frac{1}{\lambda_{sw}}\left[\mu_v A+\left[\left(\frac{\lambda_h+\mu_{rst}}{\mu_{mig}}\right)\mu_{rst}\right]\right]\right]\right] \end{aligned}\right\}^{-1} \qquad (24)$$

Where

$$A = \left(\frac{\lambda_h+\mu_{rst}}{\mu_{mig}}\right)\left(\frac{\mu_{rst}}{\lambda_v(1-c)+\lambda_v}\right)$$

$$B = \mu_v\left(\frac{\mu_{rst}}{\lambda_v(1-c)+\lambda_v}\right)+\mu_{rst}$$

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
$P_{vRF_{i,j}}$	:	The probability of the VM is in failure due to estimation of software aging failure state
P_{H_j}	:	The probability of the VMM is in healthy state
$P_{vSW_{i,j}}$	:	The probability of the VM is in switchover 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

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

B. Availability and Downtime Analysis

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

Availability = 1- Unavailability (25)

Availability= $1 - P_{D2}$ (26)

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

Downtime(T)=T $* P_{D2}$ (27)

C. Numerical Examples

In this subsection, we illustrate the applicability of the SPN model and solution methodology through numerical examples. 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.

Moreover, we used the time to aging failure from the estimation of our experiment. For this purpose, we perform numerical analysis using the following failure profile mentioned in Table 6.

Table 6. Transition Firing Rates

Transitions	Firing Rates (h^{-1})
$1/t_{sw}$ (c_1)	0.3
T_{h_d}	1time/a month
T_{v_d}	1time/3 days
T_{h_fp}	1time/a week
$T_{interval}$	1time/a week
$1/T_{r1}$, $1/T_{r2}$,$1/T_{v-r}$	1 min
$1/T_{h-r}$	2 mins
$1/T_{rst}$	30 secs
$1/T_{mig}$	4 sec
$1/T_{v-repair}$, $1/T_{repair}$	30 mins
$1/T_{h-repair}$	1 hour
T_{fp}	1 time/a week
T_{fp2} , T_{fp1}	Estimation of time to software aging failure
c (estimation coverage)	variable

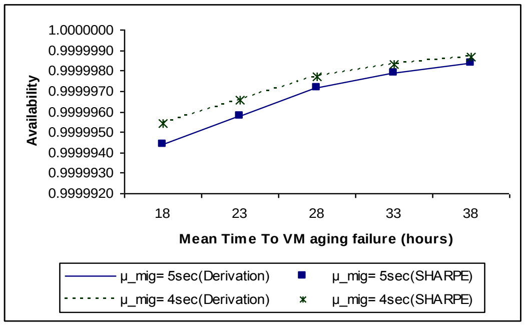


Figure 5. Availability vs different MTTF aging and VM migration

Figure 5 illustrates the availability changes for the proposed model with different VM aging failure rates and different mean time to migration had been applied. The mean time to VM migration is assumed 5 sec and 4 sec respectively. It can be observed that the VM migration rate increases, the higher availability can be achieved.

Figure 6 plotted the downtime as a function of the VM aging failure for different VM migration rates. Increases in migration rates for VM lowered the system downtime.

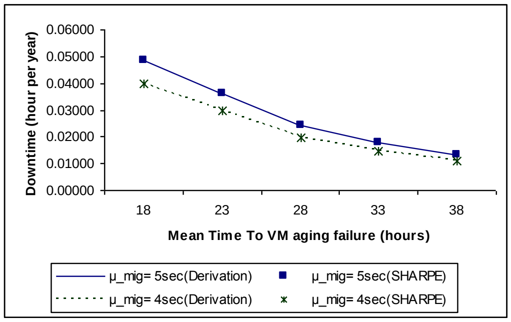


Figure 6. Downtime vs different MTTF aging and VM migration

Figure 7 illustrates the availability changes for the proposed model with different VM aging failure rates and different estimation coverage (c) for VM resource exhaustion had been applied. The probability of estimation coverage is assumed 0.9 and 0.8. It can be observed that the higher the probability of estimation coverage for VM, the higher availability can be achieved.

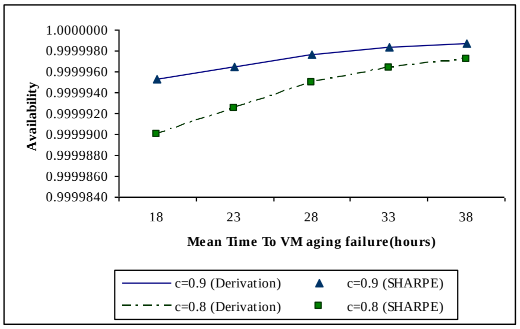


Figure 7. Availability vs. different VM aging and different c (estimation converge)

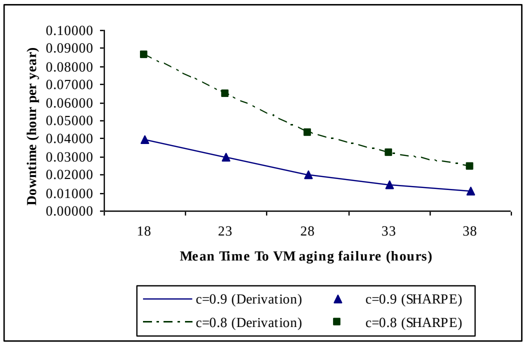


Figure 8. Downtime vs. different VM aging and different c (estimation converge)

Figure 8 plotted the downtime as a function of the VM aging failure for different estimation coverage (c) for VM resource exhaustion. Increases in the probability of estimation coverage for VM lowered the system downtime.

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.

6. Conclusion

In this paper, we have presented a framework for virtualized local disaster recovery which accepts as many services as possible. A stochastic reward nets model is constructed to describe the behavior of the proposed VLDR. We have discussed comprehensive availability model for measurement software rejuvenation on virtualization 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. The obtained results showed that the proposed framework with measurement based software rejuvenation can helpful the virtualized local disaster recovery.

References

[1] J. Alonso, I. Goiri, J. Guitart, R. Gavaldà and J. Torres, "Optimal resource allocation in a virtualized software aging platform with software rejuvenation. " In 22nd IEEE International Symposium on Software Reliability Engineering, 2011

[2] V. Castelli, R.E. Harper, P. Heidelberger, S.W. Hunter, K.S.Trivedi, K. Vaidyanathan and W.P. Zeggert, "Proactive Management of Software Aging," IBM JRD, Vol 45, No. 2, pp.311–332, Mar. 2001

[3] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, andA. Warfield. Live migration of virtual machines. In Proc. NSDI '05, May 2005.

[4] D. Clitherow, M. Brookbanks, N. Clayton, and G. Spear, ‘‘Combining High Availability and Disaster Recovery Solutions for Critical IT Environments’’, IBM Systems, Journal 47, No. 4, 563–575 (2008).

[5] F. Machida, D. Kim, J. Park and K. S. Trivedi, Toward Optimal Virtual Machine Placement and Rejuvenation Scheduling in a Virtualized Data Center, In Proc. of the 1st Int'l Workshop on Software Aging and Rejuvenation (WoSAR2008), 2008.

[6] B. C. Martin, "Disaster Recovery Plan Strategies and Processes", Version 1.3, February 2002.

[7] 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.

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

[9] K. S.Trivedi, "Probability and Statistics with Reliability, Queuing, and Computer Science Applications", John Wiley and Sons, 2002.

[10] K. Vaidyanathan and K.S. Trivedi, "A Measurement-based Model for Estimation of Resource Exhaustion in Operational Software Systems," In Proc. of ISSRE 1999, Boca Raton, FL, Nov. 1999.

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