Reliability Modeling and Analysis of Application Servers using Stochastic Petri Net Model

Ohnmar Nhway
University of Computer Studies, Mandalay,
The Union of Myanmar.
skynhway@gmail.com

***Abstract*- As modern society relies on the fault-free operation of complex computing systems, system fault-tolerance has become a matter of course. So, requests on software reliability and availability have progressed greatly due to the current applications. Software applications executing continuously for a long period of time show a degraded performance and/or an increased occurrence rate of hang/crash failure. In this paper, we present and analyze a proactive fault-tolerance using virtualization technology and rejuvenation (preventive maintenance) technique based on stochastic Petri Net Model. This proposed system presents a new model using the virtualized reliability recovery plan to offer high reliability for application servers. Moreover, our results show that the comparison of numerical calculation and SHARPE (Symbolic Hierarchical Automated Reliability and Performance Evaluator) tool simulation.**

Keywords: Reliability, stochastic Petri Net Model, Software rejuvenation, Virtualization.

I. INTRODUCTION

Today, requests on software reliability and availability have progressed greatly due to the current applications whereas malicious attacks and faults are increasingly common. As business depends on information and computing technology, continuous reliability is a universal concern.

The server processes in most client-server software systems are intended to run continuously forever except is however very difficult, if not impossible, to design and guarantee. When such software bugs (faults) are encountered during executions, they could lead to transient failures with unpredictable and costly after effects.

Fault tolerance provides continuous reliability and availability for virtual machines by creating and maintaining a secondary virtual machine that is identical to and continuously available to replace [4]. A fault tolerance virtual machine and its secondary copy are not allowed to run on the same host. Fault-tolerance techniques rely on redundancy in a system. The primary goals of this paper focus on the aspect of software for reliability of application servers since unplanned computer system outages are more likely to be the result of software failures than of hardware failures [3]. In this paper, empirical model analysis through SHARPE (Symbolic Hierarchical Automated Reliability and Performance Evaluator) tools [1] is performed to show the performance of the proposed methods. SHARPE is a well known package in the field of reliability and performability, used in universities as well as in companies. SHARPE allows its users to construct and analyze dependability, performance and performability models.

In this paper, we present preventive maintenance using a proactive fault-tolerance technique upon redundancy system. We determine service reliability by modeling the system as a simple queuing system processing jobs/requests using Stochastic Petri Net Model.

The rest of paper is organized as follows. Section 2 presents some background theory and related works. Section 3 explains Virtual Machine based Virtualized Reliability Recovery Framework and System Modeling using stochastic Petri Net Model and the evaluation of the proposed model shows on 4 VMs with a single physical server using virtualization technology and rejuvenation. Section 4 concludes conclusion.

II. BACKGROUND AND RELATED WORKS

Virtualization is a proven software technology that is rapidly transforming the IT landscape and fundamentally changing the way that people compute [10].

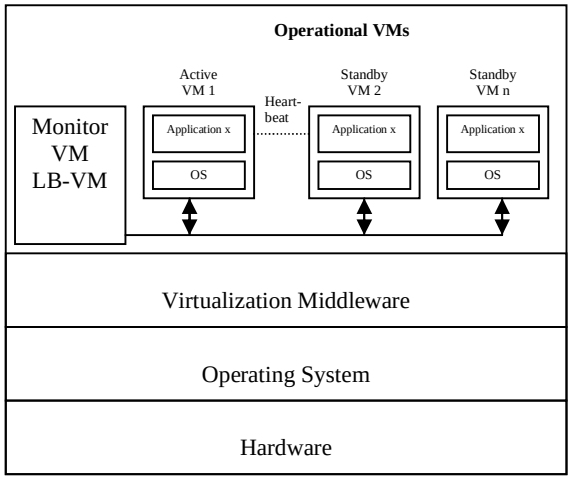

Fig.1. Virtualized single server architecture

Virtualization technology, which allows multiple operating systems to run different applications on a single computer, has caught the attention of IT managers for its promise to let them better manage and utilize corporate IT resources.

Figure 1 shows a simple example of a system we create in this paper. In this figure expresses the virtualized single server architecture for virtual machine combining with software rejuvenation and virtualization technology. On top of the virtualization middleware layer, we consider virtual machines (n+1) in a single physical application server, containing one Load Balancer VM (LB-VM) and 'n' operational virtual machines.

The primary and secondary VMs continuously exchange heartbeats. This allows the virtual machine pair to monitor the status of one another to ensure that Fault Tolerance is continually maintained.

A transparent failover occurs if the host running the primary VM fails, in which case the secondary VM is immediately activated to replace the primary VM.

Load Balancer will be provided IP fail-over capabilities and policies and also used for the monitoring purpose. While the main application server will be running on one VM, the remaining VMs will work as standby servers such as replica of the application server.

A new secondary VM is started and Fault tolerance redundancy is reestablished within a few seconds. If the host running the secondary VM fails, it is also immediately replaced. In either case, users experience no interruption in service and no loss of data [3].

Related Works

In other VMs, we will setup a SRA (Software Rejuvenation Agent) that is responsible for the rejuvenation operation. Using analytical modeling, we analyze multiple design choices when a single physical server is used to host multiple virtual machines.

Dong Seong Kim et al. [2] construct non-virtualized and virtualized two hosts system models using a two-level hierarchical approach in which fault trees are used in the upper level and homogeneous Continuous Time Markov chains (CTMC) are used to represent sub models in lower level. Then, they incorporate not only hardware failures (e.g., CPU, memory, power, etc) but also software failures including Virtual Machine Monitor (VMM), Virtual Machine (VM), and application failures. To improve high availability (HA) service and VM live migration in the virtualized system and use metrics according to system steady state availability, downtime in minutes per year and capacity oriented availability.

The authors of [5] presented a coloured stochastic Petri net model of a redundant fault-tolerant system to increase service availability in all load scenarios and improve service availability by almost 90% in a highly loaded system. Moreover, they analyze by adding a second and third redundant server yields further but much lower improvement and under low load the benefit of additional servers is not as pronounced.

Fumio Machida et al. [6] present the comparison of three techniques in terms of steady-state availability and the number of transactions lost in a year and finds the optimal combination of rejuvenation trigger intervals for each rejuvenation technique by a gradient search method based on a server virtualized system.

Ralf H. Reussner et al. [8] focus on rich architecture definition language (RADL) oriented towards modem industrial middleware platforms such as Microsoft's .NET and Sun's EJB. Their methods involve parameterised contractual specifications based on state machines and thus permit efficient static analysis. They show how RADL allows software architects Moreover, they advocate a reliability model parameterized by required component reliability in a deployment context.

In this paper, we mainly emphasize the healing of software aging of application server in a single physical machine and hardware failure is not considered in this paper. Analytical models are mathematical models which are an abstraction from the real world system and relate only to the behavior and characteristics of interest [7]. We construct the state transition models to describe the behavior of the virtualized cluster systems with software rejuvenation. We use stochastic Petri Net approach to build models and evaluate the models through both analytic analysis and SHARPE tool simulation.

III. Proposed Framework And System Modeling

A. VRR (Virtualized Reliability Recovery) Plan Framework

In this section, we present how reliability of our system makes use of the virtualization technology to apply the rejuvenation process. The proposed model in this paper is to hold multiple replicas of the application in a single physical server configuration, and trigger the rejuvenation of each one in response to detection of software bugs (faults). Moreover, we will use migration process when the primary application in the active virtual machine is to be rejuvenated.

B. VRR Model

Analytical models are mathematical models which are an abstraction from the real world system and relate only to the behavior and characteristics of interest. We apply stochastic Petri Net based approach to build the model. For this purpose, the parameters are chosen as shown in Table I.

TABLE I
SYSTEM-OPERATING PARAMETERS

Transition Parameters	Firing Rate(hr^{-1})
$T\lambda_d$	1 time/year
$T\mu$	1 time/2 hour
$T\lambda_s$	1 time/10 sec
$T\lambda_f$	1 time/2 year
$T\lambda$	1 time/1 hour
$T\lambda_r$	1 time/2 min
$T\mu_r$	1 time/60 sec
TM1,H	2
TD,M1	0.3
TD,M2	0.3
TM2,H	3
Tup	1

In this section, we construct the state transition model to describe the behavior of virtualized cluster systems as shown in Figure 2 and 3. In our models there are 4 states: Healthy state, Detecting state, Rejuvenation state and Failure state. Initially all of the VMs are in a healthy working state.

The assumptions used in the modeling are as follows:

- Time to repair rate is transition $T(\mu)$.
- Time to failure rate is transition $T(\lambda_f)$.
- The rejuvenation rate is transition $T(\mu_r)$ of the VM are identical at all states.
- Time to detection and switchover is triggered after a mean duration $1/\lambda_d$ and $1/\lambda_s$ respectively.

According to this framework, first the active VM and all standby VMs are in the healthy state (H_i). When software applications execute continuously for long periods of time the processes corresponding to the software in execution age or slowly degrade with respect to effective use of their system resources.

After detection has completed, no action is taken when the system is in healthy state of its lifetime. Whereas the active VM is about to be rejuvenated, the application software may change from the unstable state to the switchover state at a rate of $(i*\lambda)$. When all ongoing requests are finished in the active VM, the active VM state may change from the unstable state to the rejuvenation state with rate $(i*\lambda_r)$.

After rejuvenating the VM, the unstableness of the system is removed automatically. In the failure state (F_i), all VMs stop running and no available VM remains. The tasks of fault detection and/or switchover from the active VM to the standby VM are represented in the switchover state (S_i). If the active VM has encountered the software aging, the standby VM will take the role of the active VM and continues to operate based on the current state. So, during the rejuvenation process, continuous service is possible. We define the steady-state probabilities of the system as follows:

• Probabilities in the healthy state:

$$\sum_{i=1}^{n} = P_{Hi}$$

• Probabilities in the detecting state:

$$\sum_{i=1}^{n} = P_{Di}$$

• Probabilities in the rejuvenation state:

$$\sum_{i=1}^{n} = P_{Ri}$$

• Probabilities in the failure state:

$$\sum_{i=1}^{n} = P_{Fi}$$

• Probability in the failure state: P_{H0}

(n = the number of operational VMs)

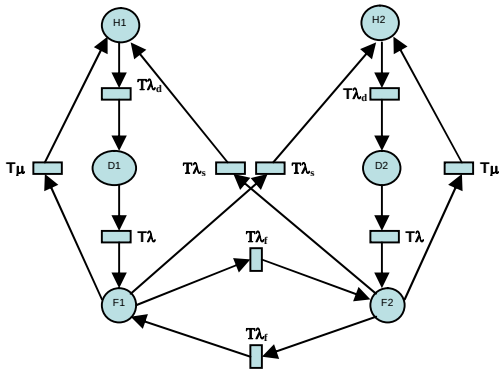

Fig.2. Reliability Models with 2VMs1P
(Without rejuvenation)

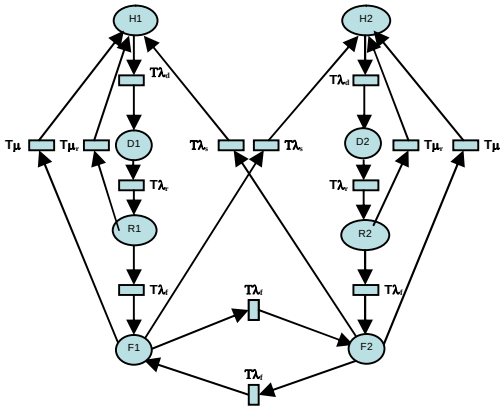

Fig.3. Reliability Models with 2VMs1P
(With rejuvenation)

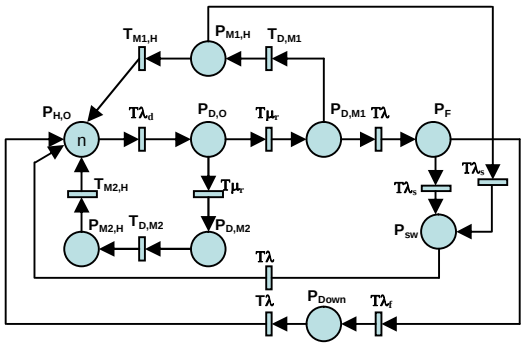

Fig.4. Virtualized Reliability Recovery Plan Framework

Now, we consider the state transition diagrams of 2VMs1P (without Rejuvenation and with it) as shown in Figure 2 and 3. We express the numerical calculation based on the steady-state probabilities by first writing down the steady-state balance equations of Figure 2 are as follows:

$$T\lambda_d PH_1 = T\mu PF_1 + T\lambda_s PF_2 \quad \dots (1)$$

$$T\lambda\ PD_1 = T\lambda_d PH_1 \quad \dots (2)$$

$$T\lambda_d PH_2 = T\mu PF_2 + T\lambda_s PF_1 \quad \dots (3)$$

$$(T\lambda_f + T\mu + T\lambda_s)\ PF_1 = T\lambda_f PF_2 + T\lambda\ PD_1 \quad \dots (4)$$

$$T\lambda PD_2 = T\lambda_d PH_2 \quad \dots (5)$$

$$(T\lambda_f + T\mu + T\lambda_s)\ PF_2 = T\lambda_f PF_1 + T\lambda\ PD_2 \quad \dots (6)$$

Reliability models capture failure and repair behavior of systems and their components. States of the underlying Stochastic Petri Net Model will be classified as up states or down states.

Our proposed VR process is shown in Figure 4. Using a stochastic Petri Net model, we describe the behavior of preventive maintenance (PM) model for virtualized reliability recovery plan (VRR). The circles represent places and n represents the tokens that held inside that place. The place $P_{H,O}$ models the robust and healthy state. When transition $T_{\lambda d}$ fires, the VM enters the detection state and moves from $P_{H,O}$ to $P_{D,O}$. $P_{D,O}$ finds and detects the faults in VMs. Firstly it makes minor maintenance using $T_{\mu r}$. Second, it does major maintenance using $T_{\mu r}$. During minor maintenance, an active VM can provide continued service because it is a partial system clean up. On the other way, one of the active VMs at primary site needs to do major maintenance, the services are migrated to one of the standby VMs at virtualized reliability recovery plan.

The system is reliable before the rejuvenation process in Detection state (D) and after the detection process in Failure state (F). According to [9], the system reliability in the steady-state is defined as follows:

Reliability + Unreliability = 1
Reliability = 1- Unreliability

Reliability *(2VMs1P)* = 1- (P_{D1}+P_{D2}+P_{F1}+P_{F2})

Reliability (nVMs1P) = 1- ($\Sigma D_n + \Sigma F_n$)

System notation
1 VM: only one active VM
2 VMs: one active VM and one standby VM
3 VMs: one active VM and two standby VMs.
4 VMs: one active VM and three standby VMs.

C. Simulation Results

We describe the simulation results of the proposed model through SHARPE tool. So, we may calculate firstly Mean Time between Failure (MTBF) rate based on one time per hour in Figure 5, 6 and 7. The change in the reliability of a system with the different number of VMs and different MTBF rates is plotted.

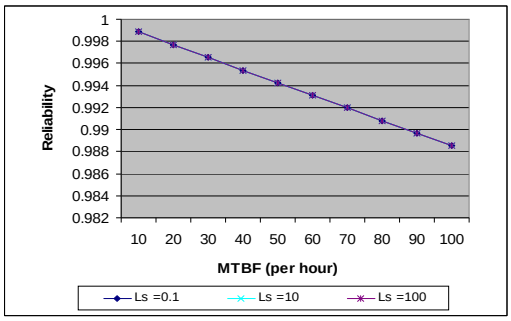

Fig.5. Reliability Vs MTBF for 2VMs1P with (Without rejuvenation) λ_s

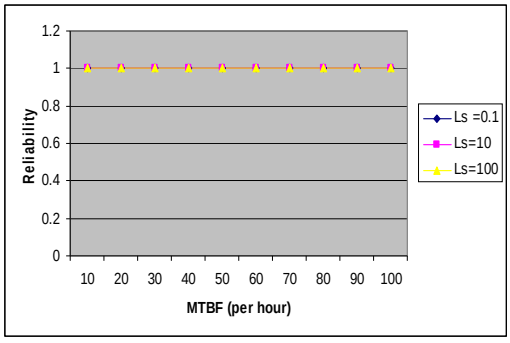

Fig.6. Reliability Vs MTBF for 2VMs1P with (With rejuvenation) λ_s

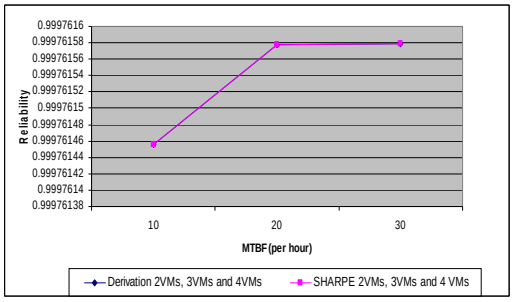

Fig.7. Reliability Vs MTBF for Virtualized Reliability Recovery Plan

We used several different values of MTBF rates based on one time per hour ($1/T_\lambda$ = 10 to $1/T_\lambda$ = 100). Moreover, we have found that our derivation results and SHARPE tool results are the same using 2VMs, 3VMs and 4 VMs in Figure 7.

IV. Conclusions

In this paper, we have presented to access our Virtualized Reliability Recovery (VRR) approach for the reliability model using a Stochastic Petri Net based on the steady-state availability system in which three kinds of model (without rejuvenation, with rejuvenation and VRR plan). We employed active-standby clustering architecture in the proposed model. We also calculate reliability based on Mean Time between Failure rates upon one time per hour for firing rates of each transition in this model with evaluation by using SHARPE tool. Our experimental results show that the proposed approach of preventive maintenance can aid the virtualized reliability recovery plan.

REFERENCES

[1] C.Hirel, R.Sahner, X.Zang, K. Trivedi, "Reliability and Performability Modeling using SHARPE 2000", Computer Performance Evaluation/TOOLS 2000:345-349

[2] Dong Seong Kim et al., "Availability Modeling and Analysis of a Virtualized System", 15th IEEE Pacific Rim International Symposium on Dependable Computing, 2009.

[3] Fault Tolerance from "Advanced Server Technology Information and Communication Technology Training Institute", Union of Myanmar [Virtualization], ICTTI, Union of Myanmar, Pg-192-244.

[4] FaultTolerancSystem.ppt from http://www.ecs.umass.edu/ece/koren/FaultTolerantSystems

[5] Felix Salfner et al., "A Petri Net model for Service Availability in Redundant Computing Systems", Proceedings of the 2009 winter Simulation Conference, 2009.

[6] Fumio Machida et al., "Modeling and Analysis of Software Rejuvenation in a Server Virtualized System", 2010.

[7] Petri Net from http://en.wikipedia.org/wiki/Petri_net

[8] Ralf H. Reussner et al., "Reliability prediction for component-based software architectures", the Journal of Systems and Software 66, 2003: 241-252.

[9] Reliability Prediction Basics, Copyright 2007, ITEM Software, Inc.

[10] Virtualization from http://en.wikipedia.org/wiki/ Virtualization