Availability Modeling and Analysis on Virtualized Clustering with Rejuvenation

Thandar Thein, Sung-Do Chi, Jong Sou Park

Computer Engineering Department, Korea Aerospace University, South Korea

Summary
Nowadays, more stringent compliance regulations, businesses of all sizes are required to implement security measures to ensure their systems and data are readily available and quickly recoverable. Virtualized clustering is inherently designed for maximum data integrity and minimal recovery time in the case of site failure. Virtualization allows multiple operating system instances to run concurrently on a single physical machine. In this paper, we present a technique that can increase availability of application servers through the use of virtualization, clustering and software rejuvenation. Using analytical modeling, we analyze multiple design choices when a single physical server and dual physical servers are used to host multiple virtual machines. Analysis results are included to show the performance of the proposed method. By integrating virtualization, clustering and software rejuvenation, it is possible to benefit from increased availability, manageability and savings from server consolidation through virtualization without decreasing uptime of critical services.

Keywords:
Availability, clustering, software rejuvenation, virtualization.

1. Introduction

With today's more stringent compliance regulations, businesses of all sizes are required to implement security measures to ensure their-systems and data are readily available and quickly recoverable. This, combined with the extreme competitiveness of today's business environment creates a critical need for companies to protect their systems from downtime. As business becomes increasingly dependent on information and computing technology, continuous availability is a universal concern.

To improve the availability of application servers, previous approaches have adopted several clustering techniques [9]. Today most business-critical servers apply some sort of server-redundancy, load-balancers and fail-over techniques. This works quite fine to tolerate application crashes.

System Virtualization techniques are getting popular and gaining significant interest in the enterprise and personal computing spaces. 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. Modern computers are sufficiently powerful to use virtual machines (VMs). Many fields, such as autonomic computing, server consolidation, security and education publish results that praise the benefits of virtualization.

System availability can be further enhanced by taking a proactive approach to detect and predict an impending outage of a specific server in order to initiate planned failover in a more orderly fashion. This approach not only improves the end user's perception of service provided by the system, but also gives the system administrator additional time to work around any system capacity issues that may arise. Software rejuvenation is a proactive management technique and naturally fit with a cluster environment [2], [13]. Within a clustered environment, rejuvenation can be performed by invoking the cluster's failover mechanisms, either on a periodic basis based on prior experience of the time to resource exhaustion, or extemporaneously, upon prediction of an impending resource exhaustion.

In this paper we describe our proposal to offer high availability mechanism for application servers. This approach has been designed to use over any server or service. Our approach makes use of several concepts: virtualization, clustering and software rejuvenation.

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 construct the state transition models to describe the behaviour of the virtualized cluster systems with software rejuvenation. We use a Markov Chains based approach to build models and evaluate the models through both analytic analysis and SHARPE tool simulation. Analysis results are included to show the performance of the proposed method.

The structure of the paper is as follows: Section 1 discusses problem issue and describes the methods to counteract the problem. Section 2 addresses background. Section 3 presents our proposed approach. In Section 4 presents a state transition models to describe the behaviors of virtualized cluster system and in the following section, the models are analyzed and experimental results are given

Manuscript received September 5, 2008.
Manuscript revised September 20, 2008.

to validate the model solutions. Finally, we conclude with Section 5.

2. Background

In this section, we introduce the concepts of clustering, virtualization and software rejuvenation.

2.1 Clustering

A cluster is a collection of independent, self-contained computer systems working together to provide a more reliable and powerful system than a single node by itself [9]. Some clusters aim to provide zero downtime via redundancy, some aim to provide high availability by load balancing tasks across the cluster and others aim to provide high performance computation via parallelism.

Clustering has proven to be an effective method of scaling to larger systems for added performance, more users, or other attributes [6], as well as providing higher levels of availability and lower management costs. As part of a clustered system, a failover process is usually employed, which transfers workload to another portion of the clustered system when a hardware or software failure occurs.

An objective of the failover process is for it to occur gracefully, without an end user knowing that a failure has occurred. In order to ensure the ability to perform a failover, sufficient spare resources must be available to accommodate the migrated workload. Another advantage of a clustered system is its ability to improve system maintenance.

2.2 Virtualization

Virtualization [1], [8], [10] is a technology that combines or divides computing resources to present one or many operating environments using methodologies like hardware and software partitioning or aggregation, partial or complete machine simulation, emulation, time-sharing, and others. Virtualization essentially lets one computer do the job of multiple computers, by sharing the resources of a single computer across multiple environments.

A virtualization layer is a software layer that abstracts the physical resources for use by the Virtual Machines (VMs). The combination of better resource use, reduced power and cooling in the data center, and more manageable applications delivery has made virtualization a very popular solution.

Virtual servers and virtual desktops host multiple operating systems and multiple applications locally and in remote locations, freeing from physical and geographical limitations. In addition to energy savings and lower capital expenses due to more efficient use of hardware resources, get high availability of resources, better desktop management, increased security, and improved disaster recovery processes when we build a virtual infrastructure.

2.3 Virtualized Clustering

Cluster Virtualization is the vastly simplified deployment and management of large pools of servers, accomplished by making very large groups of servers appear and act like a single system, as easy to manage as a single workstation.

The financial and efficiency benefits of this approach are extremely compelling, making Cluster Virtualization the most practical and cost-effective methodologies for reducing the complexity, cost and overall administrative burden of large scale computing, enabling to get the most out of computing resources. Virtualized clustering is also inherently designed for maximum data integrity and minimal recovery time in the case of site failures.

2.4 Software Rejuvenation

Software rejuvenation is a proactive fault management technique aimed at cleaning up the system's internal state to prevent occurrence of more severe crash failure in the future [11]. It involves occasionally stopping the software application, cleaning its internal state and/or its environment, and then restarting it [5].

By removing the accrued error conditions and freeing up or decrementing operating system resources, this technique proactively prevents unexpected future system outages. Unlike downtime caused by sudden failure occurrences, the downtime related to software rejuvenation can be scheduled at the discretion of the user/administrator.

According to the control mechanism, software rejuvenation can be categorized into two approaches, open-loop control and closed-loop control. Open-loop approach is characterized by no feedback information from the system after the integration of software rejuvenation functionality. Time-based rejuvenation and its variants fall into this category.

On the other hand, in closed-loop approach, rejuvenation trigger is dependent on some form of feedback from the system. The rejuvenation decision is made based on current system state and/or previous system behavior, which include workload, resource usage and failure logs [3]. Measurement-based rejuvenation belongs to this category [14].

Software rejuvenation is a cost effective technique for dealing with software faults that include protection not only against hard failures, but also against performance degradation as well. The software rejuvenation technique

works with the planned restart mechanism, which can lower the system recovery overhead by a great extent.

3. System Architecture

In this section we describe our proposal to offer high availability mechanism. Our approach makes use of several concepts: a virtualization, clustering and software rejuvenation. 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.

If failures are generated in the active node, the backup node takes over the resources of the active node and continuously provides a corresponding service based on the resources. In a cluster environment, multiple hosts provide the same service and a load balancer dispatches requests to one of these hosts. The seamless service can be provided to a target user through such an active/backup node structure. Virtualization technologies allow to run multiple virtual servers on a single physical machine. Cluster virtualization enables large pools of servers to act and appear like a single, consistent virtual machine.

Software rejuvenation technology is a natural fit with clustered systems. Within a virtualized clustering environment, rejuvenation can be performed by invoking the cluster's failover mechanisms. Using the node failover mechanisms in a high-availability virtualized cluster, one can maintain operation while rejuvenating one node at a time. The following scenarios are studied in this paper.

- Virtual/virtual, single physical machine
- Virtual/virtual, dual physical machine

This approach has been designed to use over any server or service. It is just necessary to install a virtualization layer and install some software modules. On top of the virtualization layer, we create 2 VMs per single physical machine. The main application server will be running on active VM. The remaining VM will work as standby server, where we instantiate a replica of the application server. The cluster manager is running directly in the VM as shown in Figure 1 and 2.

The cluster manager is used for software, handling all the things that are needed such as supervision of resources (processes, IP addresses), restarting of resources, and handling of failover. The cluster manager is also used to monitor applications and do local recoveries.

Live migration is currently getting very popular for virtualization technologies. Live migration allows an operating system and its applications to be migrated to a new server to balance the load over the available hardware. Virtualization allows to migrate a virtual machine to another server while preserving memory processes and network connections.

3.1 Virtual/Virtual, Single Physical Machine

This setup builds an HA cluster between two or more virtual machines on a single physical machine as shown in Figure 1. Physical machine hosts the virtual machines. The cluster manager itself is running directly in the virtual machines. This setup does not protect against hardware failures. So the physical machine itself is a SPOF (single point of failure).

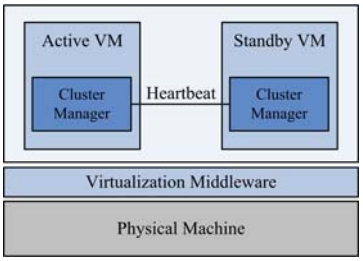

Fig. 1 Virtual/virtual, single physical machine architecture.

3.2 Virtual/Virtual, Dual Physical Machines

In this scenario an HA cluster is built between two or more virtual machines, each of them running on different physical machines. The SPOF of a single physical machine is eliminated. Since the cluster manager is running in a virtual machine it can be easily moved to other physical machines. Moreover backup and recovery procedures are traditionally simplified through the use of virtualization.

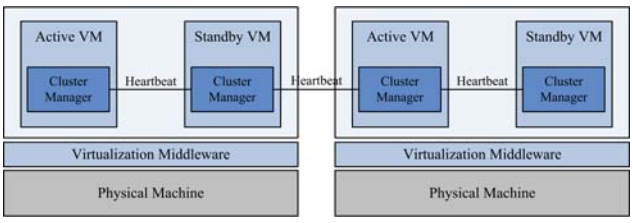

Fig. 2 Virtual/virtual, dual physical machines architecture.

3.3 Proposed System Flow

Our proposed system flow is shown in Figure 3. When some potential anomaly happens, a rejuvenation operation will be triggered. In order not to lose any in-flight request or session data at the time of rejuvenation, first standby

VM will be started and then all the new requests and sessions are migrated from the active VM to standby VM.

When all the ongoing requests are finished in active VM, then active VM will be rejuvenated. In case of a fail/switchover the whole virtual machine is failed over to the standby physical machine.

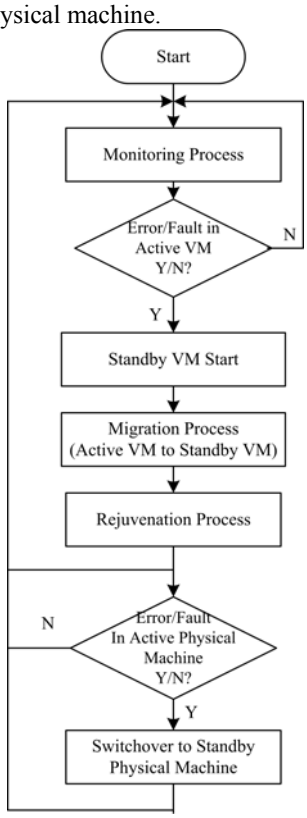

Fig. 3 Proposed system flow.

4. Modeling and Analysis

Using analytical modeling, we analyze multiple design choices when a single physical server and dual physical servers are used to host multiple virtual machines. In this section, we are interested in determining how our proposed approach can enhance the availability of the system. We construct the state transition models to describe the behavior of virtualized cluster systems as shown in Figure 4 and 7. By mapping through actions to these transition models with stochastic process, we get mathematical steady-state solution of the chain. In our models there are 4 states: Healthy state, Unstable 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:

- Failure rate (λ) and repair rate (μ) of the VM are identical at all states.
- Unstable rate (λ_u) and rejuvenation rate (λ_r) of VM are identical at all states.

As time progresses, active VM eventually transit to unstable state with rate ($i * \lambda_u$). The active VM is still operational in this state but can fail with ($i * \lambda_{sv}$). If a VM has encountered the error or fault, the VM has to be rejuvenated with a rate ($i * \lambda_r$) and it can recover with a rate (μ_r). In this case, the VM goes back to healthy state which represents the clean state of the VM. In the failure state (F, F_j), all VMs stop running and no available VM remains. Our state transition diagram in Figure 4 and Figure 7 can be described as Markov process class and we can perform steady state analysis of the diagram. To acquire system availability, we perform numerical analysis and SHARPE tool evaluation. The exact model parameter values for the model are not known, however, a good estimate value for a range of model parameter is assumed. For this purpose, the parameters are chosen as shown in Table 1.

Table 1 System-operating parameters

Parameters	Values
λ	1 time/year
λ_u	2 times/month
λ_r	1 time/month
λ_{sv}	4 times/month
$1/\lambda_s$	10 min
μ	2 times/day
$1/\mu_r$	5 min

λ is the rate at which a state transitions from unstable state to the failure state and λ_u is the rate at which a state transitions from healthy state to unstable state. λ_r is the rate at which a VM is rejuvenated once the need to rejuvenate has been established. $1/\lambda_{sv}$ is the mean switchover time; the time needed to transfer from the active VM to the standby VM and $1/\lambda_s$ is the mean switchover time; the time needed to transfer from the active physical machine to the standby physical machine. $1/\mu_r$ is mean time spent during the rejuvenation process. μ is the rate at which a VM is repaired when it has suffered an unplanned VM failure.

4.1 Availability Analysis on Virtual/Virtual Single Physical Machine (2VMs1P)

Now we consider the state transition diagram of (2VMs1P) as shown in Figure 4.

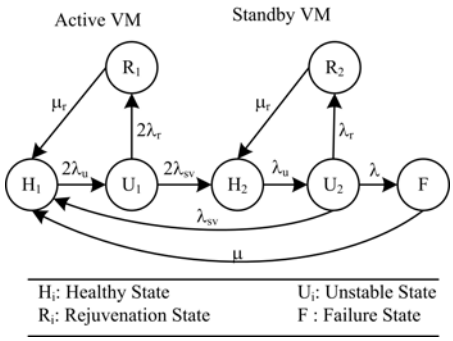

Fig. 4 State transition diagram for 2VMs1P .

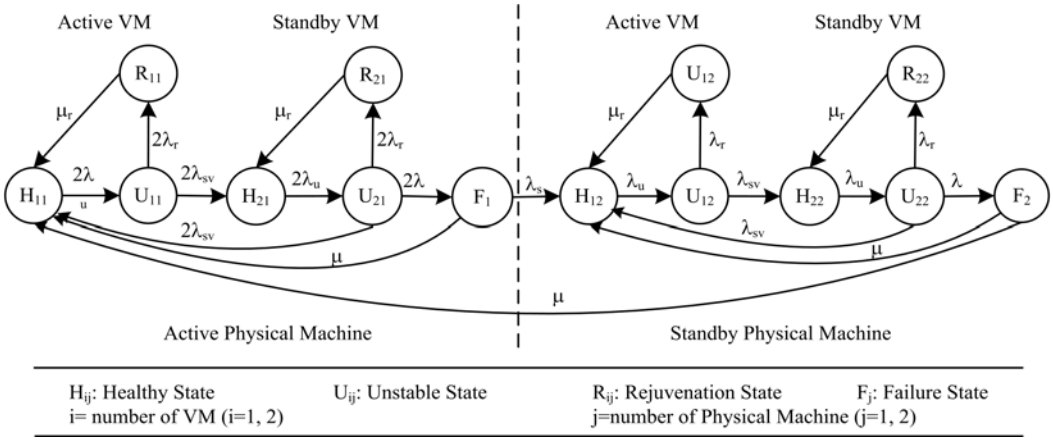

Fig. 7 State transition diagram for 2VMs2P.

We may compute the steady-state probabilities by first writing down the steady-state balance equations of Figure 4 are as follows:

$$\mu P_F + \lambda_{sv} P_{U_2} + \mu_r P_{R_1} = 2\lambda_u P_{H_1} \tag{1}$$

$$\lambda_u P_{H_1} = (\lambda_{sv} + \lambda_r) P_{U_1} \tag{2}$$

$$2\lambda_r P_{U_1} = \mu_r P_{R_1} \tag{3}$$

$$2\lambda_{sv} P_{U_1} + \mu_r P_{R_2} = \lambda_u P_{H_2} \tag{4}$$

$$\lambda_u P_{H_2} = (\lambda_{sv} + \lambda_r + \lambda) P_{U_2} \tag{5}$$

$$\lambda_r P_{U_2} = \mu_r P_{R_2} \tag{6}$$

$$\lambda P_{U_2} = \mu P_F \tag{7}$$

The conservation equation of Figure 4 is obtained by summing the probabilities of all states in the system and the sum of the equation is 1.

$$\sum_{i=1}^{2} P_{H_i} + \sum_{i=1}^{2} P_{u_i} + \sum_{i=1}^{2} P_{R_i} + P_F = 1 \tag{8}$$

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

$$P_{H_1} = \left[1 + \frac{\lambda_u}{\lambda_{sv} + \lambda_r} \left\{ 1 + \frac{2\lambda_r}{\mu_r} + \frac{2\lambda_{sv}}{\lambda_{sv} + \lambda} \left(1 + \frac{\lambda_{sv} + \lambda_r + \lambda}{\lambda_u} + \frac{\lambda_r}{\mu_r} + \frac{\lambda}{\mu} \right) \right\} \right]^{-1} \tag{9}$$

$$P_{U_1} = \frac{\lambda_u}{\lambda_{sv} + \lambda_r} P_{H_1} \tag{10}$$

$$P_{R_1} = \frac{2\lambda_u}{\lambda_{sv} + \lambda_r} \frac{\lambda_r}{\mu_r} P_{H_1} \tag{11}$$

$$P_{H_2} = \frac{2\lambda_u}{\lambda_{sv} + \lambda_r} \frac{\lambda_{sv}}{\lambda_{sv} + \lambda} \frac{\lambda_{sv} + \lambda_r + \lambda}{\lambda_u} P_{H_1} \tag{12}$$

$$P_{U_2} = \frac{2\lambda_u}{\lambda_{sv}+\lambda_r}\frac{\lambda_{sv}}{\lambda_{sv}+\lambda}P_{H_1} \quad (13)$$

$$P_{R_2} = \frac{2\lambda_u}{\lambda_{sv}+\lambda_r}\frac{\lambda_{sv}}{\lambda_{sv}+\lambda}\frac{\lambda_r}{\mu_r}P_{H_1} \quad (14)$$

$$P_F = \frac{2\lambda_u}{\lambda_{sv}+\lambda_r}\frac{\lambda_{sv}}{\lambda_{sv}+\lambda}\frac{\lambda}{\mu}P_{H_1} \quad (15)$$

Availability models capture failure and repair behavior of systems and their components. States of the underlying Markov chain will be classified as up states or down states. The system is not available in the rejuvenation process in rejuvenation state (R_2) and the failure state (F). The system availability in the steady-state is defined as follows:

Availability = 1- Unavailability

$$Availabili\,ty_{(2VMs1P)} = 1-(P_{R_2}+P_F) \quad (16)$$

The meaning of the probabilities is as follows:

P_{H_i} The probability of the VM is in healthy state

P_{U_i} The probability of the VM is in unstable state

P_{R_i} The probability of the VM is in rejuvenation state

P_F The probability of the VM is in failure state

i = 1, 2 (number of virtual machines)

The variation of the steady-state availability of system with different number of rejuvenation rates and rejuvenation time is plotted in Figure 5. We perform software rejuvenation with the interval from (rate=0: no rejuvenation) to (rate=3). We see from Figure 5 that the amount of availability increment from not performing rejuvenation to performing rejuvenation is significant. As unstable states are removed frequently with high rejuvenation rates, the availability of the virtualized cluster system is increases.

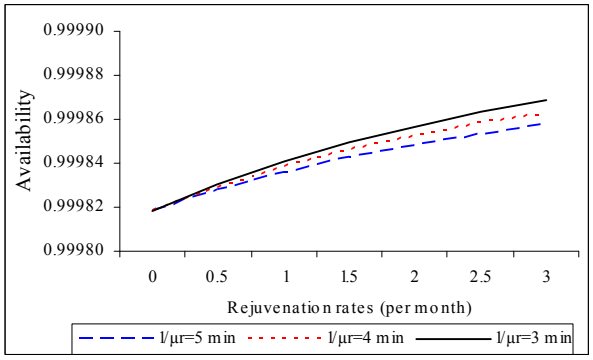

Fig. 5 Availability vs different rejuvenation time and rejuvenation rates (2VMs1P).

The influence of failure rates along with different rejuvenation rates on availability is shown in Figure 6. We assume the failure rates are one time per year and two times per year.

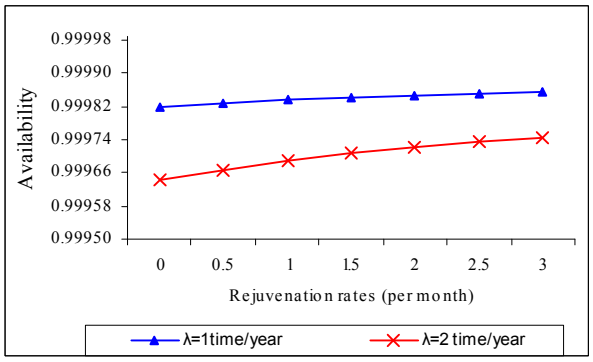

Figure 6. Availability Vs different failure rates and rejuvenation rates (2VMs1P).

High availability systems require fewer failures and faster repair. We observe from Figure 6 that the failure rate of the system acts as an important factor in the availability of the virtualized cluster system.

4.2 Analysis on Virtual/Virtual, Dual Physical Machines (2VMs2P)

Now we consider the state transition diagram of virtual/virtual dual physical machines as shown in Figure 7. The steady-state balance equations of the diagram are as follow:

$$\mu P_{F_1} + \mu P_{F_2} + 2\lambda_{sv}P_{U_{21}} + \mu_r P_{R_{11}} = 2\lambda_u P_{H_{11}} \quad (17)$$

$$\lambda_u P_{H_{11}} = (\lambda_{sv}+\lambda_r)P_{U_{11}} \quad (18)$$

$$2\lambda_r P_{U_{11}} = \mu_r P_{R_{11}} \quad (19)$$

$$2\lambda_{sv}P_{U_{11}} + \mu_r P_{R_{21}} = 2\lambda_U P_{H_{21}} \tag{20}$$

$$\lambda_u P_{H_{21}} = (\lambda_{sv} + \lambda_r + \lambda)P_{U_{21}} \tag{21}$$

$$2\lambda_r P_{U_{21}} = \mu_r P_{R_{21}} \tag{22}$$

$$2\lambda P_{U_{21}} = (\mu + \lambda_s)P_{F_1} \tag{23}$$

$$\mu P_{F_2} + \lambda_{sv}P_{U_{22}} + \lambda_S P_{F_1} + \mu_r P_{R_{12}} = 2\lambda_u P_{H_{12}} \tag{24}$$

$$\lambda_u P_{H_{12}} = (\lambda_{sv} + \lambda_r)P_{U_{12}} \tag{25}$$

$$\lambda_r P_{U_{12}} = \mu_r P_{R_{12}} \tag{26}$$

$$\lambda_{sv}P_{U_{12}} + \mu_r P_{R_{22}} = \lambda_U P_{H_{22}} \tag{27}$$

$$\lambda_u P_{H_{22}} = (\lambda_{sv} + \lambda_r + \lambda)P_{U_{22}} \tag{28}$$

$$\lambda_r P_{U_{22}} = \mu_r P_{R_{22}} \tag{29}$$

$$\lambda P_{U_{22}} = 2\mu P_{F_2} \tag{30}$$

The conservation equation of Figure 7 is obtained by summing the probabilities of all states in the system and the sum of the equation is 1.

$$\sum_{i,j=1}^{2} P_{H_{iJ}} + \sum_{i,j=1}^{2} P_{u_{ij}} + \sum_{i,j=1}^{2} P_{R_{ij}} + \sum_{j=1}^{2} P_{F_j} = 1 \tag{31}$$

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

$$P_{H_{11}} = \left[\begin{array}{l} 1+2Z+4\dfrac{\mu}{\lambda}Z\left\{ \begin{array}{l} \dfrac{\lambda_{sv}+\lambda}{\lambda_{sv}}\left(1+\dfrac{\lambda_u}{\lambda_{sv}+\lambda_r}\dfrac{\lambda_r}{\mu_r}\right)+ \\ 1+\dfrac{\lambda_{sv}+\lambda_r+\lambda}{\lambda_u}+\dfrac{\lambda_r}{\mu_r} \end{array} \right\}+ \\ \dfrac{\lambda_u}{\lambda_{sv}+\lambda_r}\left\{ \begin{array}{l} 1+\dfrac{2\lambda_r}{\mu_r}(1+\dfrac{\lambda_{sv}}{\lambda_{sv}+\lambda})+ \\ \dfrac{\lambda_{sv}}{\lambda_{sv}+\lambda}(1+\dfrac{\lambda_{sv}+\lambda_r+\lambda}{\lambda_u}+\dfrac{2\lambda}{\mu+\lambda_s}) \end{array} \right\} \end{array} \right]^{-1} \tag{32}$$

$$P_{U_{11}} = \frac{\lambda_u}{\lambda_{sv}+\lambda_r}P_{H_{11}} \tag{33}$$

$$P_{R_{11}} = \frac{2\lambda_u}{\lambda_{sv}+\lambda_r}\frac{\lambda_r}{\mu_r}P_{H_{11}} \tag{34}$$

$$P_{H21} = \frac{\lambda_U}{\lambda_{sv}+\lambda_r}\frac{\lambda_{sv}}{\lambda_{sv}+\lambda}\frac{\lambda_{sv}+\lambda_r+\lambda}{\lambda_u}P_{H_{11}} \tag{35}$$

$$P_{U_{21}} = \frac{\lambda_u}{\lambda_{sv}+\lambda_r}\frac{\lambda_{sv}}{\lambda_{sv}+\lambda}P_{H_{11}} \tag{36}$$

$$P_{R21} = \frac{2\lambda_u}{\lambda_{sv}+\lambda_r}\frac{\lambda_{sv}}{\lambda_{sv}+\lambda}\frac{\lambda_r}{\mu_r}P_{H_{11}} \tag{37}$$

$$P_{F_1} = \frac{2\lambda_u}{\lambda_{sv}+\lambda_r}\frac{\lambda_{sv}}{\lambda_{sv}+\lambda}\frac{\lambda}{\mu+\lambda_s}P_{H_{11}} \tag{38}$$

$$P_{H_{12}} = 4Z\frac{\lambda_{sv}+\lambda_r}{\lambda_u}\frac{\lambda_{sv}+\lambda}{\lambda_{sv}}\frac{\mu}{\lambda}P_{H_{11}} \tag{39}$$

$$P_{U_{12}} = 4Z\frac{\lambda_{sv}+\lambda}{\lambda_{sv}}\frac{\mu}{\lambda}P_{H_{11}} \tag{40}$$

$$P_{R_{12}} = 4Z\frac{\lambda_{sv}+\lambda}{\lambda_{sv}}\frac{\mu}{\lambda}\frac{\lambda_r}{\mu_r}P_{H_{11}} \tag{41}$$

$$P_{H_{22}} = 4Z\frac{\lambda_{sv}+\lambda_r+\lambda}{\lambda_u}\frac{\mu}{\lambda}P_{H_{11}} \tag{42}$$

$$P_{U_{22}} = 4Z\frac{\mu}{\lambda}P_{H_{11}} \tag{43}$$

$$P_{R_{22}} = 4Z\frac{\mu}{\lambda}\frac{\lambda_r}{\mu_r}P_{H_{11}} \tag{44}$$

$$P_{F_2} = 2ZP_{H_{11}} \tag{45}$$

Where

$$Z = \frac{\lambda_u}{\mu} - \frac{\lambda\lambda_u\lambda_{sv}}{(\mu+\lambda_s)(\lambda_{sv}+\lambda_r)(\lambda_{sv}+\lambda)} - \frac{\lambda_u\lambda_{sv}^2}{\mu(\lambda_{sv}+\lambda_r)(\lambda_{sv}+\lambda)} - \frac{\lambda_r\lambda_u}{\mu(\lambda_{sv}+\lambda)} \tag{46}$$

The meaning of the probabilities is as follows:

$P_{H_{ij}}$ The probability of the VM is in healthy state

$P_{U_{ij}}$ The probability of the VM is in unstable state

$P_{R_{ij}}$ The probability of the VM is in rejuvenation state

P_{F_j} The probability of the VM is in failure state

i = 1, 2 (number of virtual machines)

j = 1, 2 (number of physical machines)

The system is not available in the rejuvenation processes in rejuvenation state (R_{22}) and the failure states (F_1, F_2). The system availability in the steady-state is defined as follows:

Availability = 1- Unavailability

$$Availabili\,ty_{(2VMs\,2P)} = 1 - (P_{R_{22}} + P_{F_1} + P_{F_2}) \qquad (47)$$

The change in the availability of system with the different rejuvenation time and rejuvenation rates is plotted in Figure 8. It can be observed from this Figure that the faster the rejuvenation time, the system can achieve the higher availability and the higher the rejuvenation rates, the system can also achieve higher availability.

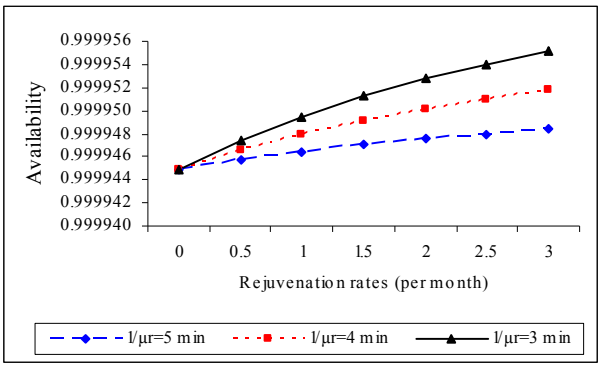

Fig. 8 Availability Vs rejuvenation time and rejuvenation rates (2VMs2P).

The influence of failure rate along with different rejuvenation rates on availability is shown in Figure 9. This result shows that the failure rate of application server acts as an important factor in the availability of 2VMs2P configuration system.

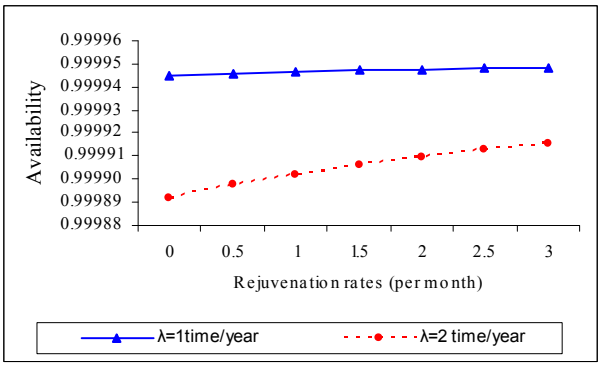

Fig. 9 Availability vs different failure rates and rejuvenation rates (2VMs2P).

4.3 Models Validation

SHARPE [4], [12] 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. And we have found that the evaluation results through SHARPE are same with our mathematical derivation results as shown in Figure 10.

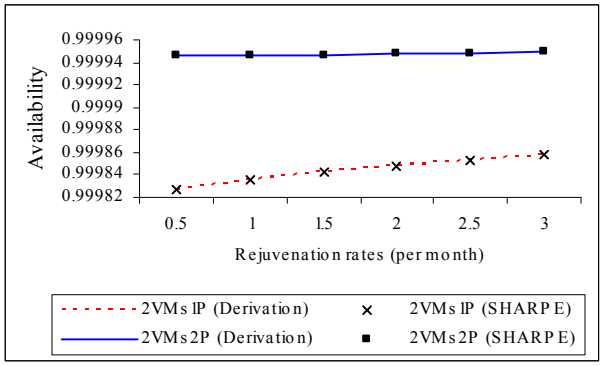

Fig. 10 Availability vs different number of physical machines and rejuvenation rates.

According to the required availability level, the decision making of a rejuvenation rate is possible under consideration of multiple design choices when a single physical server and dual physical servers are used to host multiple virtual machines.

5. Conclusions

Virtualized clustering results in optimal simplicity of provisioning, simplicity of management, and simplicity of operation. In this paper we presented possible combinations of virtualization, high availability cluster and software rejuvenation. This approach has been designed to use over any server or service. Virtual/virtual, single physical machine setup is a very comfortable way to test different HA cluster setups without requiring much hardware. Virtual/virtual, dual physical machine setup can eliminate the single point of failure. We also presented Markov models for analyzing availability of virtualized clustering systems with proactive rejuvenation. We validated our experiment results with the evaluation results through SHARPE. It is found that our derivation results and SHARPE results are same. Using virtualized clustering with rejuvenation for the purpose of high availability provides many benefits over traditional cluster. Future work will include experiments with an implementation

under real world conditions to verify the practical efficacy of our approach.

Acknowledgments

This research was supported by the advanced Broadcasting Media Technology Research Center (ABRC) in Korea Aerospace University, Korea, under the Gyeonggi Regional Research Center (GRRC) support program supervised by Gyeonggi Province.

References

[1] J. Alonso, L. Silva, A. Andrzejak, P. Silva and J. Torres. High-available grid services through the use of virtualized clustering. In Proc. 8th IEEE/ACM Int. Conference on Grid Computing, Sept 19-21, 2007, pp. 34-41.

[2] V. Castelli, R. Harper, P. Heidelberger, S. Hunter, K. Trivedi, K. Vaidyanathan, and W. Zeggert. Proactive management of software aging. IBM Journal of Research and Development, 2001, vol. 45(2), pp. 311–332.

[3] S. Garg, A. Moorsel, K. Vaidynathan and K. S. Trivedi. A methodology for detection and estimation of software aging. In Proc. 9th Int. Symp. On Software Reliability Engineering, Paderborn, Germany, November 1998, pp. 283.

[4] C. Hirel, A. Robin, Sahner, X. Zang, K.S. Trivedi. Reliability and performability modeling using SHARPE 2000. Computer Performance Evaluation/TOOLS 2000. In Lecture Notes In Computer Science; Vol. 1786, Springer-Verlag, 2000, pp. 345-349.

[5] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton. Software rejuvenation: analysis, module and application. In Proc. 25th Int. Symp., on Fault-Tolerant Computing, Pasadena, CA, June 27-30, 1995. pp. 381-390.

[6] K. Hwang. Advanced Computer Architecture: Parallelism, Scalability and Programmability, McGraw-Hill Book Co., Inc., New York, 1993.

[7] E. Marcus and H. Stern. “Blueprints for High Availability”, Wiley, 2003

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

[9] G. Pfister. In search of clusters: The Coming Battle in Lowly Parallel Computing, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1998.

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

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

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

[13] K. Vaidyanathan, R. Harper, S. Hunter, and K. Trivedi. Analysis and implementation of software rejuvenation in cluster systems. In Proc. ACM Int. Conference on Measurement and Modeling of Computer Systems, SIGMET-RICS, 2001, pp. 67-71.

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

Thandar Thein received M.Sc. (Computer Science) and Ph.D degrees in 1996 and 2004, respectively from University of Computer Studies, Yangon, Myanmar. Currently she is doing Post Doctorate Research in Korea Aerospace University. Her research interests include Ubiquitous Sensor Network Security, Security Engineering, and Network Security and Survivability.

Sung-Do Chi received BS and MS degrees in 1982 and 1984, respectively, in electrical engineering from Yonsei University, Seoul, Korea, and a PhD degree in electrical and computer engineering in 1991 from the University of Arizona, Tucson. From 1984 to 1986, he was a part-time instructor at the Yonsei University. He also worked as software engineer at the Digital Equipment Corporation, Seoul Branch. His research interests include traffic modeling, model-based reasoning, intelligent system design methodology, discrete-event system modeling and simulation; simulation based artificial life, computer security and Biotechnology.

Jong Sou Park received the M.S. degree in Electrical and Computer Engineering from North Carolina State University in 1986. And he received his Ph.D in Computer Engineering from The Pennsylvania State University in 1994. From 1994 - 1996, he worked as an assistant Professor at The Pennsylvania State University in Computer Engineering Department and he was president of the KSEA Central PA, Chapter. He is currently a full professor in Computer Engineering Department, Korea Aerospace University. His main research interests are information security, embedded system and hardware design. He is a member of IEEE and IEICE, and he is an executive board member of the Korea Institute of Information Security and Cryptology, and Korea Information Assurance Society.