2014 IEEE International Symposium on Software Reliability Engineering Workshops

Software Maintenance Optimization based on Stackelberg game methods

Jing Zhao[1], Yan-bin Wang [2], Gao-Rong Ning [3], Cheng-Hong Wang [1], Kishor S. Trivedi [4], K-Y Cai [3], Zhen-Yu Zhang [5]
1 Department of Computer Science and Technology, Harbin Engineering University, China, zhaoj@hrbeu.edu.cn
2 Department of Industrial engineering, Harbin Institute of Technology, China, wangyb@hit.edu.cn
3 Department of Automatic Control, Beihang University, China, ninggaorong@asee.buaa.edu.cn
4 Department of Electrical and Computer Engineering, Duke University, USA, kst@duke.edu
5 The State Key Laboratory of Computer Science, Institute of Software, Chines Academy of Sciences, zhangzy@ios.ac.cn, China

***Abstract*—Application servers (AS) of virtualized platform may suffer from software aging problem. In this paper, we first formulate the system model including three virtual machines. Two of them act as the main servers, and the third machine acts as the backup node. The motivation of our formulated model is that the relationship between the service provider and the service maintainer is collaborative as well as having different goals between them, the service provider as a leader wants to maximize his system availability, while the service maintainer wants to minimize his maintenance cost. Thus, the problem of maximizing availability and minimizing cost between the service provider and service maintainer is Stackelberg game based. Next, we assume that the AS degradation is caused by resource consumption due to memory leaks for the AS on the active VMs, and we present the system degradation states based on Markov renewal processes. We give the analytical definitions of threshold levels for R_{alert} at each VM, which are used to determine the optimal rejuvenation schedules. In addition, we obtain the steady-state availability expressions for the system and the mean maintenance cost. Finally, we give the Stackelberg strategy with the open-loop information and the solutions for the game theory by a numerical illustration.**

***Index Terms*—proactive restart; reactive restart; Stackelberg game; availability;**

I. Introduction

The increasing complexity of the multiple service infrastructure with the growing adoption of cloud-ready virtualized data centers or platforms, makes the software one of the main causes of system failures. A non-negligible fraction of these software failures is the consequence of software aging phenomena, which shows a degraded performance and/or an occurrence of hang/crash failures when it has been executing continuously for a long period[1]. The causes of software aging include memory leaks, unreleased file-locks, and round-off errors[2]. And it has been observed severe software failures in industrial environments leading to economic and reputational losses. For example, Google Doc service failed on 2011 due to memory leak with a high workload[3]. Amazon web services were off-line in October 2012 as a consequence of a memory leak and the failure of the monitoring system to capture it promptly[4]. So, even these companies which are able to hire some of the most skilled engineers, will eventually suffer software aging-related failures due to the increasing complexity of the software.

The aging effects and their dynamics can be captured by one or more indicators. Two kinds of metrics have been traditionally used to detect and measure software aging: system resources and user perceived attributes. System resources refer to available memory, CPU utilization and used swap space etc. , while user perceived attributes include response time, throughput and loss probability etc.. In [5] [6], the authors use metrics of available memory and used swap space to estimate the times-to-failure for systems that fail due to software aging. And in papers [7] [8], the authors employ accelerated degradation tests(ADT) technique or accelerated life tests(ALT) to accelerate system resources usage to reduce experimental time.

Memory leak is recognized to be one of the causes of resource exhaustion problems, which is one of the most serious reasons for software aging. In papers[9],[10], and [11], the authors study software aging due to memory leaks and also the software rejuvenation strategies by means of experimental and/or analytical approaches. Bao et al. [12] proposed a hierarchical model for the proactive fault management, in which the higher level is a semi-Markov process, and the low level of the model hierarchy is a degradation model in the presence of system resource leaks. As opposed to software aging, software rejuvenation has been proposed by Huang et al.[13]. This approach involves resetting the software system and ”clearing” the internal state or environment of the software before severe aging occurs so as to avoid system crash, which can maintain the robustness of software systems and avoid unexpected system outrages. So it is an efficient proactive control mechanism to counteract the effects of the aging.

Stackelberg games called leader-follower games, are intially proposed by Stackelberg in 1952 based on some economic monopolization phenomena. In a Stackelberg game, one player acts as a leader and the rest as followers. The problem is then to find an optimal strategy for the leader, assuming that the followers react in a rational way that they optimize their objective functions given the leader’s actions[14]. Cloud computing generally involves virtualization of computing resources, such as Software as a Service(SaaS). In this paper, a virtual signed license is built between the service provider and the service maintainer. The service provider may provide the SaaS to an

978-1-4799-7377-4/14 $31.00 © 2014 IEEE
DOI 10.1109/ISSREW.2014.38

end user at the high availability, while the third-part service maintainer need to make the system maintenance cost least. This virtual agreement naturally formulates a Stackelberg game, where the leader refers to the service provider and the follower refers to the service maintainer. We conduct a case study for the virtualized network system, two of the running operating systems work as main service and the other one works as the backup service. The aging due to memory leak is assumed at the main application server, and proactive rejuvenation as well as reactive restart action would be utilized by the service maintainer. And we also present and obtain the definitions of threshold levels of virtual machines exactly by analytical approach, which could be used to decide when and whether to employ the maintenance operations on each virtual machine to make the cost least.

This paper is organized as follows. In Section II, an system model for a virtualized platform is given to discuss the processes of AS degradations running on different VMs. In Section III, we analyze the system degradation states based on renewal processes, and present the definitions of threshold levels for R_{alert} for each VM, that are used to determine the optimal rejuvenation schedules. We also give analytic expressions for these threshold levels. Accordingly, software rejuvenation policy and reactive maintenance techniques are adopted based on different states for the system. In section 4, we obtain the steady-state availability expressions for the system by using renewal theory, and also the mean maintenance cost. Furthermore, we give the Stackelberg strategy with the open-loop information. In Section V, we obtain the solutions for the game theory by numerical computation. In Section VI, we give the conclusions of this paper.

II. System model and Game theoretic approach in signed license

A. System model

We present a system model of virtualized platform as shown in Fig. 1, in which the two servers act as the main service, and the third server act as the backup server, and the ASs run on the virtual machines. The maintenance actions for AS are classified as proactive rejuvenation and the reactive restart explained as the follows.

- AS proactive restart (AS Rejuv.). Application proactive restart is a light-weight plan of rejuvenation to proactively restart program before the application crashes. This approach is useful to preserve volatile in-memory application data required to be treated appropriately during the shutdown process. Alternatively, the application crash will eventually cause the lost of the in-memory data.
- AS reactive restart (AS Rea. Res). When the application crashes, it will be reactively restarted. The main overhead cost of this technique is the longer service downtime compared with application proactive restart approach. Moreover, this approach has also other negative impacts like the lost of in-memory application data.

We assume that the degradation processes for the ASs of VM_1, VM_2 can be independently determined by monitoring

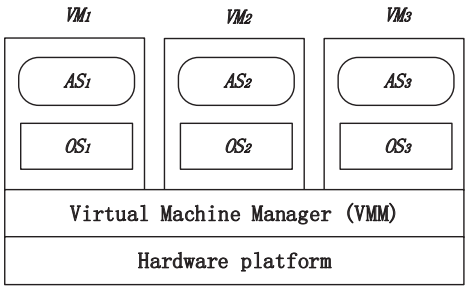

Fig. 1: The structure of the virtualized platform

its some appropriate parameters. In the following, the observed resource available are referred to as the degradation indexes. Let $R_1(t)$, and $R_2(t)$ be the values of the resource available of the AS measured at time t, respectively. Hence, $(R_1(t)$, and $R_2(t))$ $(t > 0)$ are two stochastic renewal processes modeling the two AS degradation of VM_1, VM_2 versus time. Let $X_i^k (i = 1, 2; k = 1, 2, \cdots)$ represents the amount of memory allocated to the application minus the memory released in the time interval $(\tau \cdot (k-1), \tau \cdot k]$, respectively. Assume that X_1^k $(k = 1, 2, \cdots)$ are independent and identically distributed, and the same with X_2^k $(k = 1, 2, \cdots)$. Then

$$R_i(k \cdot \tau) = R_i^{total} - \sum_{l=1}^{k} X_i^l, i = 1, 2$$

where R_i^{total} is the values of R_i resources when they are initial fully available. Let $f_{X_i^k}(x)$ be the pdf of X_i^k, then $R_i^{total} - R_i(k \cdot \tau)$ with the pdfs $f_{X_i^k}^{*k}(x)$ are the memory usage of the two levels in time interval $(0, k \cdot \tau]$, where $f_{X_1}^{*k}(x)$ and $f_{X_2}^{*k}(x)$ are the convolutions of $f_{X_1^k}(x)$, $(1 \le k \le k)$ and $f_{X_2^k}(x)$, $(1 \le i \le k)$, respectively. $R(t) = 0$ means that AS run out of its total memory available, and is down at time t. A possible artistic realization of related renewal processes modeling the two-granularity software aging is shown in Fig. 2.

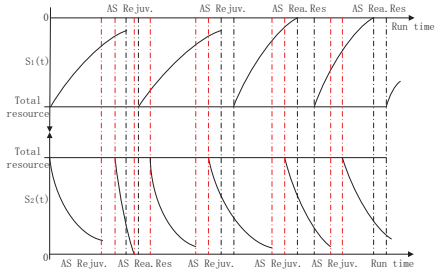

Fig. 2: ASs aging and rejuvenation of the two VMs

Suppose that when a request arrives, it is assigned to VM_1 with probability p, so it is assigned to VM_2 with probability $1-p$. And we suppose that the aging rates of the ASs for two virtual machines have a relation with the number of requests.

The expectation $\mathbb{E}(X_1^k)$ of X_1^k is proportional to the number of request, that is the probability p contributes $\mathbb{E}(X_1^k)$. Fig.3 shows the transition states of the system, where state $(1,1)$ represents that the two active ASs are robust, while the state $(D,1)$ means that one of the AS is at the degradation state, the other is in the robust. The item A, or B in the state (A,B) does not differ the state of which AS of VM, and it represents the state of one of the AS for VM is A or B, so we can conclude the meanings of the other states transitions in the Fig. 3.

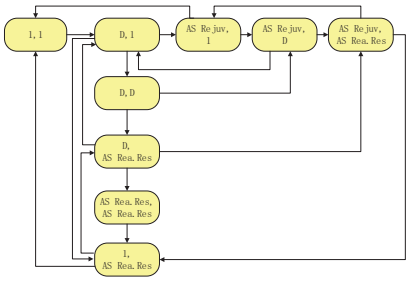

Fig. 3: The state transient diagram of the system

B. Game theoretic in signed license

The Stackelberg leadership game model is a strategic game in which there are two types of players: leaders and followers. The 'leader' player moves first and then the 'follower' player moves sequentially. Thus, the follower chooses a strategy to optimize its own objective function, knowing the leader's move. The leader, has to optimize its own objective function by predicting the optimal response of the follower. As in our signed license between service provider and service maintainer, in the next following sections, we should first give the defination there exists a Stackelberg equilibrium in this game, and then calculate the Stackelberg equilibrium of the game.

III. Rejuvenation policy based on risk-level

Whether a AS needed to be proactive restarted or reactive restarted is decided by the current value of the measure index $R_1(t)$ and $R_2(t)$. In this case, an alert threshold R_{alert} is established and the rejuvenation of AS_i is performed as soon as $R_i(t) \leq R_i^{alert}$. Hence the probability that the system crashes before rejuvenation is determined by R_i^{alert}. Suppose that the conditional probability of the AS_i crash is

$$\mathbb{P}(X_i^{k+1} > R_i^{alert} \mid R_i(k\cdot\tau) = R_i^{alert}) = \alpha_i. \tag{1}$$

Accordingly, α_i denotes that if a AS crashes, then reactive restart of the AS is needed.

Suppose that the maintenance cost for proactive or reactive restart of a AS is m_{Rej} and m_{Res} respectively. And suppose that the time needed for proactive or reactive restart of a AS is T_{Rej} and T_{Res}, respectively.

If AS_i rejuvenation takes place at time $k\cdot\tau$ (the k^{th} monitoring), then the mean cost is $\frac{m_{Rej}}{k\cdot\tau+T_{Rej}}$. If AS will not be rejuvenated at time $k\cdot\tau$ but at time $(k+1)\cdot\tau$, then the mean cost is $\frac{(1-\alpha_i)m_{Rej}}{(k+1)\cdot\tau+T_{Rej}} + \frac{\alpha\cdot m_{Res}}{(k+1)\cdot\tau+T_{Res}}$. Then probability α_i satisfies

$$\frac{m_{Rej}}{k\cdot\tau+T_{Rej}} = \frac{(1-\alpha_i)m_{Rej}}{(k+1)\cdot\tau+T_{Rej}} + \frac{\alpha_i\cdot m_{Res}}{(k+1)\cdot\tau+T_{Res}}. \tag{2}$$

So we have

$$\alpha_i = \frac{\frac{m_{Rej}\cdot\tau}{(k\cdot\tau+T_{Rej})((k+1)\cdot\tau+T_{Rej})}}{\frac{m_{Res}}{(k+1)\cdot\tau+T_{Res}} - \frac{m_{Rej}}{(k+1)\cdot\tau+T_{Rej}}}. \tag{3}$$

Consider that the variables k, T_{Rej}, T_{Res} in (3) can be replaced by their expectation $\frac{R_i^{total}-R_i^{alert}}{\mathbb{E}(X_i)}$, $\mathbb{E}(T_{Rej})$, $\mathbb{E}(T_{Res})$ respectively. Then one of the optimal R_i^{alert} is given by

$$\hat{R}_i^{alert} = \left\{ R_i^{alert} \ : \ \mathbb{P}[X_i^{k+1} \geq R_i^{alert} \mid R_i(k\cdot\tau) = R_i^{alert}] \right.$$
$$\left. = \frac{\frac{m_{Rej}\cdot\tau}{(\frac{R_i^{total}-R_i^{alert}}{\mathbb{E}(X_i)}\cdot\tau+\mathbb{E}(T_{Rej}))((\frac{R_i^{total}-R_i^{alert}}{\mathbb{E}(X_i)}+1)\cdot\tau+\mathbb{E}(T_{Rej}))}}{\frac{m_{Res}}{(\frac{R_i^{total}-R_i^{alert}}{\mathbb{E}(X_i)}+1)\cdot\tau+\mathbb{E}(T_{Res})} - \frac{m_{rej}}{(\frac{R_i^{total}-R_i^{alert}}{\mathbb{E}(X_i)}+1)\cdot\tau+\mathbb{E}(T_{Rej})}} \right\}. \tag{4}$$

IV. Availability analysis

With the alert threshold being given, for any time t, the p_2^{mig} denotes the probability that the new AS_2 service request is migrated to the back up AS of VM_3, which can be obtained using the renewal theory as

$$p_2^{mig} = (1-\alpha_1)\frac{\mathbb{E}(T_1)}{\mathbb{E}(T_{Rej})+\mathbb{E}(T_1)} + \alpha_1\frac{\mathbb{E}(T_1)}{\mathbb{E}(T_{Res})+\mathbb{E}(T_1)}, \tag{5}$$

where $\mathbb{E}(T_1)$ is the average active life of AS_1.

$$p_1^{mig} = (1-\alpha_2)\frac{\mathbb{E}(T_2)}{\mathbb{E}(T_{Rej})+\mathbb{E}(T_2)} + \alpha_2\frac{\mathbb{E}(T_2)}{\mathbb{E}(T_{Res})+\mathbb{E}(T_2)}, \tag{6}$$

where $\mathbb{E}(T_2)$ is the active life of AS_2. The P_1^{mig} and P_2^{mig}, are the probabilities of AS_1's or AS_2's migrating service to AS_3. when the AS_1 is in the normal operation. If the AS_2 is at aging or at the resource exhaustion state, the AS_2's migrating its service to the third back up AS_3 at VM_3 depends on that the AS_3 is not being utilized by AS_1, i.e., at this case the non-utilized of the AS_3 equals to the probability of usage of AS_1. Thus we can obtain that P_2^{mig} as the probability of the usage probability of the VM_1. And by the same token, P_1^{mig} can be obtained as the probability of the usage probability of the VM_2. Therefore, the mean running maintain cost m for

the system is

$$
\begin{aligned}
m = & \frac{m_{Rej} \cdot (1-\alpha_1)(1-p_1^{mig})}{\mathbb{E}(T_1)+\mathbb{E}(T_{Rej})} \\
& + \frac{m_{Rej} \cdot (1-\alpha_2)(1-p_2^{mig})}{\mathbb{E}(T_2)+\mathbb{E}(T_{Rej})} \\
& + \frac{m_{Rej} \cdot (1-\alpha_1)p_1^{mig}}{\mathbb{E}(T_1)+\mathbb{E}(T_{Rej})} \\
& + \frac{m_{Rej} \cdot (1-\alpha_2)p_2^{mig}}{\mathbb{E}(T_2)+\mathbb{E}(T_{Rej})} \\
& + \frac{m_{Res} \cdot \alpha_1}{\mathbb{E}(T_1)+\mathbb{E}(T_{Res})} + \frac{m_{Res} \cdot \alpha_2}{\mathbb{E}(T_2)+\mathbb{E}(T_{Res})} \\
& + \frac{m_{insp}}{\tau},
\end{aligned} \tag{7}
$$

where m_{insp} is the maintenance cost for each inspection. The availability of the system equals the availability for the actions: Rejuvenation and migration, Rejuvenation and no-migration, Dead and migration, Dead and no-migration of AS_1 and AS_2. The availability for AS_1 is A_{AS_1},

$$
\begin{aligned}
A_{AS_1} = & \frac{p \cdot \mathbb{E}(T_1) \cdot (1-\alpha_1)(1-p_1^{mig})}{\mathbb{E}(T_1)+\mathbb{E}(T_{Rej})} \\
& + \frac{p \cdot (\mathbb{E}(T_1)+\mathbb{E}(T_{Rej})) \cdot (1-\alpha_1)p_1^{mig}}{\mathbb{E}(T_1)+\mathbb{E}(T_{Rej})} \\
& + \frac{p \cdot (\mathbb{E}(T_1)+\mathbb{E}(T_{Rej})) \cdot \alpha_1 \cdot (p_1^{mig})}{\mathbb{E}(T_1)+\mathbb{E}(T_{Res})} \\
& + \frac{p \cdot \mathbb{E}(T_1) \cdot \alpha_1 \cdot (1-p_1^{mig})}{\mathbb{E}(T_1)+\mathbb{E}(T_{Res})}.
\end{aligned} \tag{8}
$$

The availability for AS_2 is A_{AS_2},

$$
\begin{aligned}
A_{AS_2} = & \frac{(1-p) \cdot \mathbb{E}(T_2) \cdot (1-\alpha_2)(1-p_2^{mig})}{\mathbb{E}(T_2)+\mathbb{E}(T_{Rej})} \\
& + \frac{(1-p) \cdot (\mathbb{E}(T_2)+\mathbb{E}(T_{Rej})) \cdot (1-\alpha_2)p_2^{mig}}{\mathbb{E}(T_2)+\mathbb{E}(T_{Rej})} \\
& + \frac{(1-p) \cdot (\mathbb{E}(T_2)+\mathbb{E}(T_{Rej})) \cdot \alpha_2 p_2^{mig})}{\mathbb{E}(T_2)+\mathbb{E}(T_{Res})} \\
& + \frac{(1-p) \cdot \mathbb{E}(T_2) \cdot \alpha_2 \cdot (1-p_2^{mig})}{\mathbb{E}(T_2)+\mathbb{E}(T_{Res})}.
\end{aligned} \tag{9}
$$

Therefore, the total availability of the system, A, equals $A_{AS_1} + A_{AS_2}$.

In this environment, the owner of the system wants to improve the availability, his strategy is to change p. The maintainer is to minimize the maintenance cost, his strategy is to change τ. Stackelberg strategy with an open-loop information structure is used for the game. The owner is assumed to be the leader and select his strategy first, while maintainer is the follower. The follower does not know the rational reaction of the leader and must optimize his criterion m for a given control p of the leader. The leader foresees this and controls the entire system.

Definition 1: If there exists a mapping $f : \mathcal{P} \to \mathcal{T}$ such that for any fixed $p \in \mathcal{P}$,

$$
m(p, f(p)) \leq m(p, \tau), \quad \forall \tau \in \mathcal{T},
$$

TABLE I: Parameters for numerical computation

m_{Rej}	m_{Res}	K_1	K_2	$\mathbb{E}(T_{Rej})$	$\mathbb{E}(T_{Res})$
$1K$	$4K$	$200/mu1$	$180/mu2$	0.02minutes	0.5minutes

TABLE II: The optimized value for the availability as well as the maintenance cost

$p = 0.1 * 5$	$\tau = 0.059 * 7 = 0.413$
$p_1^{mig} = 0.99993$	$p_2^{mig} = 0.99927$
$R_1^{alert} = 18.2909$	$R_2 alert = 3.9853$
$minM = 0.0344k$	$maxA = 0.99998$
$\alpha_1 = 0.00631$	$\alpha_2 = 0.000605$

and if there exists a $\bar{p} \in \mathcal{P}$ such that

$$
A(\bar{p}, f(\bar{p})) \geq A(p, f(p)), \quad \forall p \in \mathcal{P},
$$

then the pair $(\bar{p}, \bar{\tau}) := (\bar{p}, f(\bar{p})) \in \mathcal{P} \times \mathcal{T}$ is called a stackelberg strategy pair with player 1 as the leader.
The Stackelberg strategy is the optimal strategy for the leader when the follower reacts by playing optimally.

V. Numerical Results

We present numerical results to illustrate the Stackelberg strategy. The total amount of the system resource for $AS_i, (i = 1, 2)$ is M_1 and M_2, which equals to $200M$bytes, $180M$bytes, respectively. The memory consumption rate in each hour of the $AS_i, (i = 1, 2)$ follows the normal distribution, and the maintenance cost for proactive or reactive restart of a AS is m_{Rej} and m_{Res}, which equals $1k$ dollars, and $4k$ dollars, respectively. The time needed for proactive or reactive restart of a AS is T_{Rej} and T_{Res}, which equals 0.02 minute, and 0.5 minute, respectively. K_1 and K_2 represent the monitored times, the parameters are shown in table I. The assumed probability of service entering VM_1 is p, and then the probability of service entering VM_2 is $1 - p$. The probability p changes from the value 0.1 to 0.9 increasing 0.1, and the τ changes from value 0.059 hour to 0.59 hour increasing 0.059. Matlab is used to conduct numerical results. The normal rand numbers for memory consumption in each hour(kB/hour) are generated, i.e., the mean memory consumption in each hour for AS_1 is μ_1, which equals $7 \cdot p \cdot \tau$, and the squared error is σ_1, which equals $8 \cdot \sqrt{p \cdot \tau}$. And also, the mean memory consumption in each hour for AS_2 is μ_2 equals $6 \cdot (1-p) \cdot \tau$, and the squared error is $5 \cdot \sqrt{(1-p) \cdot \tau}$.

The Stackelberg game optimized value for the maximized availability $maxA$ equals 0.99998, and the optimized maintenance cost $minM$ equals $0.0344K$, and the optimized $p = 01 * 5$, while the $\tau = 0.059 * 7 = 0.413$. The other correspondingly optimized parameters are shown in table II. We also show the availability vs allocation probability trends as shown in Fig.4, the Stackelberg equilibrium solution for the service provider is when $p = 0.5$, the system reach the maximized availability $maxA = 0.99998$. At this case, the follower as service maintainer react in a rational way to optimize his objective cost function m to minimize his cost, so the obtained solution for service maintainer is $p = 0.5, \tau = 0.0597 * 7 = 0.413$, the

correspondingly minimized cost equals $minM = 0.0344K$, as shown in Fig.5.

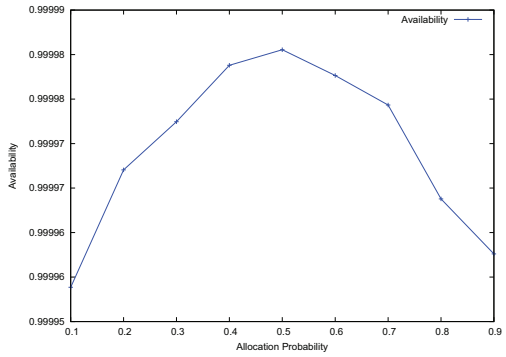

Fig. 4: Availability vs allocation probability

The optimized stackelberg game solution for cost is shown in Fig.5.

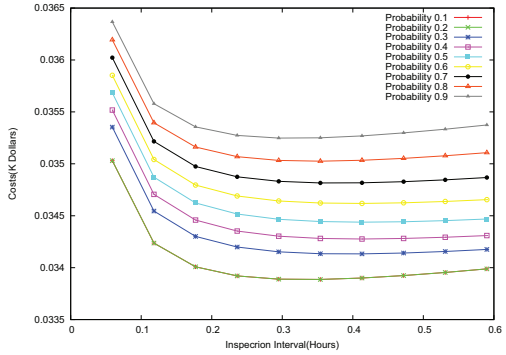

Fig. 5: Cost vs inspection interval

VI. CONCLUSION

We presented the AS degradation process model of the virtualized platform. We formulated the problem of signed license between service provider and service maintainer, aimed at maximizing availability and minimizing cost as Stackelberg game based. The service provider as a leader wants to maximize his system availability, while the service maintainer wants to minimize his maintenance cost. Next, We defined the proactive action as well as reactive restart action for the AS, and obtained the analytical definitions of threshold levels for A_{red} at each VM, which are used to determine the optimal maintenance schedules. In addition, we obtained the steady-state availability expressions for the system and the mean maintenance cost. Finally, we gave the Stackelberg strategy with the open-loop information and the solutions for the game theory by a numerical illustration.

ACKNOWLEDGEMENT

This research was supported in part by the Opening Fund of the State key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences (SYSKF1405)

REFERENCES

[1] M. Grottke, L. Li, K. Vaidyanathan, and K. Trivedi, "Analysis of software aging in a web server," *IEEE Trans. Rel.*, vol. 55, no. 3, pp. 411–420, 2006.

[2] K. Vaidyanathan and K. Trivedi, "A comprehensive model for software rejuvenation," *IEEE Trans. Dependable Secure Comput.*, vol. 2, no. 2, pp. 124–137, 2005.

[3] K. S. Trivedi, "Reducing system reboot time with kexec," [EB/OL], 2011, http://www.slideshare.net/PersistentInc/turing100persistent.

[4] Techcrunch.com, "Amazon web services outage caused by memory leak and failure in monitoring alarm," [EB/OL], 2012, http://techcrunch.com/2012/10/27/amazon-web-services-outage-caused-by-memory-leak-and-failure-in-monitoring-alarm/.

[5] S. Garg, A. Vanmoorsel, K. Vaidyanathan, and K. S. Trivedi, "A methodology for detection and estimation of software aging," in *Proc. the Ninth International Symposium of Software Reliability Engineering*, 1998, pp. 283–292.

[6] M. Grottke, L. Li, K. Vaidyanathan, and K. S. Trivedi, "Analysis of software aging in a web server," *IEEE Transactions on Reliability*, vol. 55, no. 3, pp. 415–420, 2006.

[7] R. J. Matias, P. A. Barbetta, and K. S. Trivedi, "Accelerated degradation tests applied to software aging experiments," *IEEE Transactions on Reliability*, vol. 59, no. 1, pp. 102–114, 2010.

[8] J. Zhao, Y. Wang, G. Ning, K. S. Trivedi, R. Matias, and K. Y. Cai, "A comprehensive approach to optimal software rejuvenation," *Performance Evaluation*, vol. 73, no. 11, pp. 917–933, 2013.

[9] J. Zhao, Y. Jin, K. S. Trivedi, and R. Matias, "Injecting memory leaks to accelerate software failures," in *Proc. 22th International Symposium on software reliability engineering*, 2011, pp. 260–269.

[10] J. Alonso, J. Berral, R. Gavald à , and J. Torres, "Adaptive on-line software aging prediction based on machine learning," in *Proc. Intl.Conf. Dependable Systems and Networks, DSN 2010*, 2010, pp. 507–516.

[11] A. Mac ê do, T. B. Ferreira, and R. Matias, "The mechanics of memory-related software aging," in *2010 IEEE Second International Workshop on Software Aging and Rejuvenation*, 2010, pp. 1–5.

[12] Y. Bao, X. Sun, and K. Trivedi, "A workload-based analysis of software aging and rejuvenation," *IEEE Trans. Rel.*, vol. 54, no. 3, pp. 541–548, 2005.

[13] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton, "Software rejuvenation:analysis, module, and applications," in *Proc. 25th International Symposium on Fault-Tolerance Computing*, vol. 255, no. 5050, 1995, pp. 381–390.

[14] von. Stackelberg H, "The theory of the market economy," in *Oxford University Press, UK*, 1952.