A Method of Autonomous Recoverment for Survivable System

Sun Tao[1, a], Zhao Guo-Sheng[1,b],Wang Jian[2] , Su Yan[1] and Liu Hai-Long[1]

[1]Center of Computer Network and Information, Harbin Normal University, Harbin, China

[2]Institute of Computer Science, Harbin University of Science and Technology, Harbin, China

[a]suntao@hrbnu.edu.cn, [b]zgswj@163.com

Keywords: Survivability; Autonomous Recoverment; Petri Net

Abstract. A method of autonomous recoverment for survivable system was proposed. Firstly, failure models of survivable system were formally described based on stochastic Petri nets, in which a number of measurable index parameters for recoverment were also given out. And then a recursive autonomous recoverment strategy was presented,Compared with the traditional cyclical recovery strategy, recoverment strategies reduced the time and costs of emergency recovery.

Introduction

Many studies show that the reboot[1] technology can effectively eliminate some errors, failure states that were accumulated during the system's operation, including those errors, failures that were caused by human attacks. Therefore, reboot can effectively recover the system or application service to the initial good condition. But in current, the research on the recoverment strategy of survivable system is not deep enough; the latest literature is performed to the service-level. Castelli[2] simply divided reboot into two levels: service-level and system-level, but did not give the specific methods of determining the reboot interval and reboot priority. Hong [3] defined the recovery granularity according to the actual measured value of system resources, however, when the current values of resource losses can not accurately reflect the degree of system resource losses, this method is no longer applicable. Although Xie[4] adopt the semi-Markov process to optimize the above methods, but the basic ideas did not change. The studies in above all have their limitations, and still appear rough. Relatively, the research on software fault tolerance, the reboot technology research after software failure has entered to the mature stage. After the recursive restorability [5,6] technology was presented in the research of recovery-oriented computing project that was undertook by the UC Berkeley and the Stanford university in cooperation, Candea proposed the micro-reboot technology [7,8], whose idea is to build a reboot tree in advance, each node on the reboot tree is an application or process that can reboot independently.

Failure Model and Recoverable Index

We think that the system behavior with a high survivability and its result can be expected through the solving of its model, it is able to make the behavior states can be monitored, behavior results can be assessed, and abnormal behaviors can be controlled. The recovery-oriented survivable system exists some measurable indexes: MTTF (Mean Time to Failure), MTBF (Mean Time between Faults), and MTTR (Mean Time to Repair).

Failure model

Most failure models assume that the system behavior is a Poison process; these models generally assume that each failure event associates with a restoration-recovery process, and think that the relationship between recovery and restoration can change in a wider range. First of all we should make limitations on the two extreme conditions: (1) after each failure occurred, only to restore the system, and then the restored system could return to the normal working ability; (2) the relationship between restoration and failure is uncertain, but decided by a specific stochastic process [9].

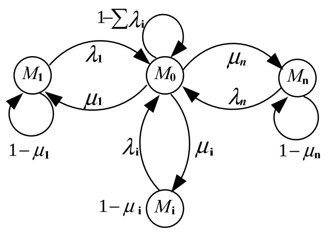

Fig 1. MC of series restorable system

Failure model of series system

Assume that the series system has n components, the failure rate and recovery rate of each component are *failure$_i$*$=\lambda_i$, *repaire$_i$*$=\mu_i$ ((i=1,2,$\cdots$,n) respectively. Assume that $X(t)$ is the system state space, then using the literature[10] can get the stochastic Petri model and by analyzing the state transition can obtain the corresponding Markov chain as shown in figure 1.

Failure model of parallel system

Assume that no two or more than two components are failure at the same time, and at any time no two or more than two components can be recovered. Parallel recoverable system consists of n parts, assume that the failure rate and recovery rate of each part are the same, and *failure*$_i$= λ ,*repaire*$_i$= μ . Assume that *X(t)* is the system state space, then we can get the stochastic Petri model and by analyzing the state transition can get the corresponding Markov chain as shown in figure 2.

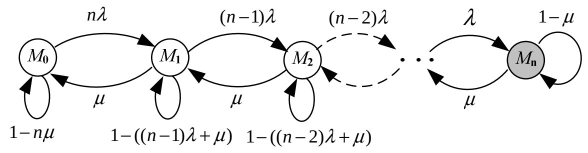

Figure 2. MC of parallel restorable system

Failure model of redundancy backup system

According to the different states which the backup components exist in, redundancy backup system mainly has three forms: cold backup, warm backup and hot backup. Cold backup means that the spare components are in not working state. Warm backup means that the spare components are in the exact same working state, but relative to the main components, the spare components are in light load state, and its failure rate is smaller than the main components', but in the hot backup system, their failure rates are the same[11]. Below we make use of stochastic Petri nets to describe the general k/n redundant backup system model, figure 3 is a cold backup system, and figure 4 is a warm backup and hot backup system.

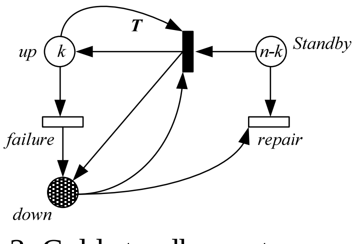

Figure 3. Cold standby system

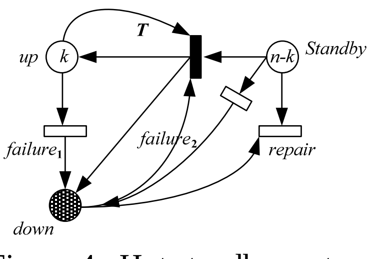

Figure 4. Hot standby system

The solving of recoverable index parameter value

After the stochastic Petri nets have modeled for the failure models, then MC that is isomorphic with the SPN model could be got according to the reachable graph. Suppose that the reachable marking set in MC, [M0] has n elements, means that MC has n states. Defining the state transition matrix Q= [q_{ij}], in which the element q_{ij} on Q's non diagonal can be acquired like this: when there is an arc which is connected from state M_i to state M_j, then the speed which is marked on the arc is the value of q_{ij}, if there is no arc connected from state M_i to state M_j, then q_{ij} =0. The element q_{ij} on Q's non diagonal equals to the negative value of the summation of the rate which is marked on each arc that is outputted from state M_i. Assume that the steady state concept of MC having n states is a row vector $\Pi = (\pi_1, \pi_2, ..., \pi_n)$, according to the Markov process, having the following linear equations:

$$\begin{cases} \Pi \times Q = 0 \\ \sum \pi_i = 1 \end{cases} \tag{1}$$

Solving the linear equations, then can get the stable probability of each reachable marking $P_i(t = \infty) = \pi_i (1 \le i \le n)$.

By MC and the state transition matrix Q, We can get the solving of recoverable index parameters *MTTR, MTTF, MTBF* and so on. In which $\Lambda = -\sum_{i=1}^{n} \lambda_i$, and we can get the steady state probability when $t \to \infty$

$$\begin{cases} \pi_0 = [1 + \sum_{i=1}^{n} \frac{\lambda_i}{\mu_i}]^{-1} \\ \pi_i = \pi_0 \frac{\lambda_i}{\mu_i}, i = 1, ..., n \end{cases} \tag{2}$$

(1) *MTTF*-(Mean Time to Failure)

When the failure probability obeys to the exponential distribution, $\lambda = \sum_{i=1}^{n} \lambda_i$, then

$$MTTF = \int_0^\infty R(t)dt = \int_0^\infty \exp(-\sum_{i=1}^{n} \lambda_i t)dt = \lambda^{-1} \tag{3}$$

(2) *RECOVERBILITY*

Recoverability: after the system service failure, the probability of being restored in time interval *t* is *M* (*t*). First of all defining the system's service recovery rate $\mu(t)$:

$$\mu(t) = \frac{1}{1 - M(t)} \frac{dM(t)}{dt} \tag{4}$$

Then *M* (*t*) can be got from formula [4]:

$$M(t) = 1 - \exp(-\int_0^1 \mu(t)dt) \tag{5}$$

(3) *MTTR*-(Mean Time to Repair)

The average recovery time of survivable system can be expressed as

$$MTTR = \lambda / \rho = \sum_{i=1}^{n} \lambda_i \Big/ \sum_{i=1}^{n} (\lambda_i / \mu_i) \tag{6}$$

In which, $\rho = \sum_{i=1}^{n} \rho_i$ is recovery coefficient, $\rho_i = \lambda_i / \mu_i$ is the recovery coefficient of the ith component.

(4) *MTBF*-(Mean Time between Faults)

Mean time between faults, MTBF is approximately equal to the ratio between the normal service time and the number of failures, that is

$$MTBF = t / N_f(t) \tag{7}$$

Recoverment Strategy for Survivable System

Emergency recoverment is a technology of preventive and proactive survivability enhancement. When the survivability of a system degenerated to a certain degree, by terminating the application, or rebooting the system, application service or process so as to clean up their internal states, one can release the operating system resources, thus recovering the survivability of a system.

System-level recoverment strategy

The implementation of system-level recoverment strategy will terminate the operation of a system, the several key services offered by the system are not available now, which should be paid a certain expenses. Our method, which is relied on the existing failure distribution of a system and based on the failure model, defines the system-level recoverment interval according to the important recoverable index parameters such as MTTF, MTBF, MTTR or others, which were obtained by the survivability decline law of the failure model.

Definition 1 Emergency recoverment interval of a system is η.

$\eta_i = MTBF_i - MTTR_i$, $i=0,1,\ldots,$.

System is beginning to run with the best survivability H_{max}, subsequently, various of failures, errors, attacks and other unpredictable events occur, which result in a gradually decline of the survivability, assume that the survivability reduces to H_{min} at the moment η_i, then, the system-level recoverment strategy is implemented to make the system survivability quickly restore to the initial state, as is shown in Figure 5. Recoverment interval ($\eta_0, \eta_1, \ldots$) can be obtained from the operation law of the failure model.

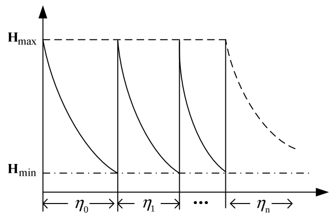

Fig 5. The system-level recoverment strategy

Service-level recoverment strategy

Whenever the survivability of a key service is reduced to a predetermined threshold P_{min}, a service-level recoverment is implemented; M times process-level recoverment are implemented when the survivability of a key service meets the condition $P_{max}>P>P_{min}$, thus getting a series of different recoverment values of system survivability ($P_{\max}^{(1)}$, $P_{\max}^{(2)}$,…, $P_{\max}^{(n)}$) and the recoverment interval (η_0, η_1, ⋯, η_n), therein, the recoverment interval value is the same as above section. Followed by recycling, as we can see from Figure 6.

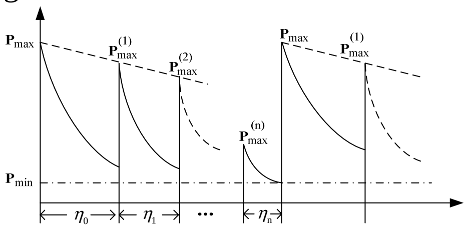

Fig 6. The service-level recoverment strategy

Simulation of Cases and Analysis

In order to track and analyse the influence on system survivability that caused by the nested recursive emergency recoverment strategy in a larger time sample space, PerfMon[13] tools were used to carry out simulation tests. As you can see from Figure 7, Compared with the traditional cyclical recovery strategy, especially when the system survivability declined to [0.35-0.45] interval, the emergency recoverment strategies have obviously improvement the system survivability, after fifty times tests in the same condition, it can effectively improve 25% of the system survivability, therefore, it achieved the purpose of enhancing the system survivability.

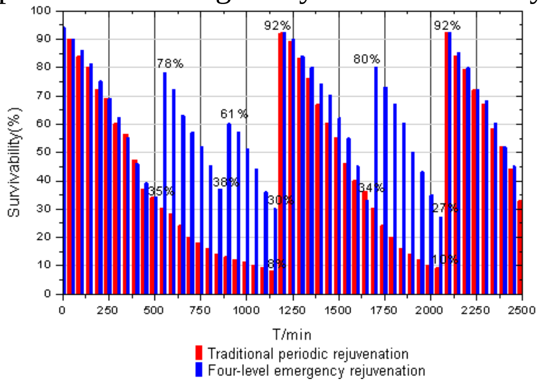

Fig 7. The survivability evolvement process

Conclusions

In order to make a reasonable fine-grained emergency recovery strategy of the survivable system, this paper presented a autonomous recoverment strategy of the survivable system, which made use of SPNs to formally describe the implementation process of the recursive reboot recovery. The simulation results showed that the proposed emergency recovery strategy reduced the time and costs of emergency recovery.

Acknowledgment

This work is supported by the National Natural Science Foundation of China(Grant No. 61202458),the Specialized Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20112303120007) and the Foundation of Heilongjiang Educational Committee (Grant No.12521146,12511099).We thank Professor J. wang and G.S. Zhao for enlightening discussions.

References

[1] G. Candea, J. Cutler, A. Fox, et al. Reducing Recovery Time in a Small Recursive Restartable System. International Conference on Dependable System and Network(DSN'02) 2002,pp.605-61.

[2] V. Castelli, R. E. Harper, P. Heidelberger, et al. Proactive Management of Software Aging. IBM Journal of Research and Development, 2001, vol.45,pp.311-332.

[3] Y. Hong, D. Chen, L. Li, et al. Closed loop design for software recoverment. Proceedings of the Workshop on Self-Healing, Adaptive and Self-Managed Systems. New York, USA, ACM,2002

[4] W. Xie, Y. Hong, K. Trivedi. Analysis of a Two-Level Software Recoverment Policy. Reliability Engineering and System Safety, 2005,vol.87,pp.13-22.

[5] D. Patterson, A. Brown, P. Broadwell, et al. Recovery Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies. Berkeley, 2002,http://roc.cs.berkeley.edu/papers /ROC_TR02-1175.pdf

[6] G. Candea, A. Fox. Recursive Restartability: Turning the Reboot Sledgehammer into a Scalpel. 8th workshop on Hot Topics in Operating Systems. Los Alamitos, USA: IEEE Computer Society, 2001,pp.125- 130.

[7] G. Candea, S. Kawamoto, Y. Fujiki, et al. Microreboot - a Technique for Cheap Recovery. 6th Symposium on Operating Systems Design and Implementation. San Francisco, CA, USA: USENIX, 2004,pp.31-44.

[8] G. Candea, J. Cutler, A. Fox. Improving Availability with Recusive Microreboots: a Soft-State System Case Study. Performance Evaluation Journal, 2004, vol.56,pp.213-248.

[9] J. C. Laprie, M. Kaaniche,K. Kanoun. Modeling Computer Systems Evolutions: Non-Stationary Processes and Stochastic Petri Nets-Application to Dependability Growth. Proceedings of the Sixth International Workshop on Petri Nets and Performance Models,1995,pp.221-230.

[10] L. Chuang. Stochastic Petri nets and system performance evaluation. Beijing: Qinghua University Press,2000

[11] L. T. Jiang, G. Z. Xue,R. D. Ying, et al. Application of stochastic petri net to system availability analysis. Journal of System Simulation, 2002, vol.14,pp.796-799.

[12] J. Xu, K. Zhang, F. Y. Liu. Linux-based computing system performance monitoring. The academic journal of Nanjing University of Technology(Natural Science), 2007,vol.31,pp.622-627.