A Self-Optimization Method for System Service Dependability based on Autonomic Computing

Qingtao Wu, Lina Zhu, Ying Lou and Ruijuan Zheng

Electronic & Information Engineering College, Henan University of Science and Technology
Luoyang, Henan Province 471023, China

Abstract

Under the intrusion or abnormal attacks, how to supply system service dependability autonomously, without being degraded, is the essential requirement to network system service. Autonomic Computing can overcome the heterogeneity and complexity of computing system, has been regarded as a novel and effective approach to implementing autonomous systems to address system security issues. To cope with the problem of declining network service dependability caused by safety threats, we proposed an autonomic method for optimizing system service performance based on Q-learning from the perspective of autonomic computing. First, we get the operations by utilizing the nonlinear mapping relations of the feedforward neural network. Then, we obtain the executive action by perceiving the state parameter changes of the network system in the service performance. Thirdly, we calculate the environment-rewarded function value integrated the changes of the system service performance and the service availability. Finally, we use the self-learning characteristics and prediction ability of the Q-learning to make the system service to achieve optimal performance. Simulation results show that this method is effective for optimizing the overall dependability and service utility of a system.

Keywords: *Autonomic Computing, Service Dependability, Feedforward Neural Network, Q-learning Algorithm*

1. Introduction

Heterogeneous and complex distributed systems currently face many safety threats such as intrusion attacks, random system failures, and everyday abnormal human operation. These threats make network service dependability a cause of concern for their users. In particular, dynamic changes in system service performance caused by the expansion and increased complexity of the system, make maintaining the system state for a system with unknown parameters more difficultly. Traditional methods for complex structures, such as firewalls, intrusion detection, fault-tolerant techniques, and user identification, can satisfy system needs no longer. Autonomic computing is inspired by the autonomic nervous system of the human body, has properties such as self-configuration, self-healing, self-optimization, and self-protection, regarded as a novel and effective approach to implementing autonomous systems to address system security issues. An autonomic system makes decisions on its own, using high-level policies; it will constantly check and optimize its status and automatically adapt itself to changing conditions.

Applying autonomic computing technology to achieve dependability has been the focus of recent studies by many institutes and scholars. References [1] and [2] discuss achieving dependability through the elimination and avoidance of system mistakes by using autonomic computing technology. References [3] and [4] discuss equipping a system with an autonomic optimizing capability through the design of an autonomic management framework. References [5] and [6] both studied an autonomic dependability model. Reference [5] focused on adding an autonomic recovery property to improve system dependability, whereas Reference [6] established a dependable autonomic management system with fault-tolerant technology, which realized to some extent the active management principle of "managing technology by technology." However, studies on applying autonomic computing to dependability, aiming to achieve autonomic optimization for systems with unknown parameters, have not yet made significant achievements.

Based on "managing technology by technology", the active management method that its structures could weaken system complexity and at the same time to provide users with reliable service, mainly faces several problems. Firstly, how to monitor the uncertain parameters which can influence the performance of service when required service credible performance declines? Secondly, how to construct adaptive adjustment process during the decline of service credible performance in order to look for the strategy to support long-term effective reliable operation of the system service? To solve these problems, this paper proposes a real-time autonomic method for optimizing service systems based on Q-learning from the perspective of autonomic computing, which can realize the independent optimization of the system service credibility from the perception of the system service state parameter, the selection of "state-action" pair, the calculation of

environmental prize value and independent executive of Q-learning algorithm, etc.

2. Autonomic model for optimizing system service dependability based on Q-learning

The Q-learning algorithm [7] is a type of reinforcement learning that can be used with any model. Instead of estimating the environment model, the algorithm can directly optimize the value of the learned state-action pair. This algorithm is therefore considered as a relatively effective reinforcement learning algorithm. By these characteristics，we adopted the algorithm in this paper. That is, the performance parameter of dependability that influences the system is adjusted, and a real-time autonomic optimization model that is oriented to the service performance of the network system is established via the Q-learning algorithm. The structure is presented in Fig. 1.

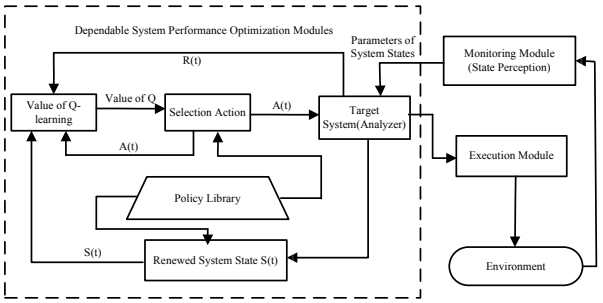

Fig. 1 Autonomic optimization model for system service dependability based on Q-learning.

This system includes the following three main modules:
(1) Monitoring module: In this module, the state perception deployed in each part of the system perceives changes in the system environment. The information collected about the state of features of the system service performance is submitted to the analyzer of the target system after being standardized.

The operating state of the network service is presented through a set of observational variables. Given time t and state observation sequence S_t, then $S_t=\{s_1,s_2,\ldots,s_m\}$, where m is the number of services that the system provides. S_i $(i=1,2,\ldots m)$ refers to the ith observation value of the operating state, and $s_i=\{v_i|v_i\in\{v_1,v_2,\ldots,v_n\}\}$, where $\{v_1,v_2,\ldots,v_n\}$ is the set of observation values for one specific service.

(2) Autonomic optimizing module: This module is the core of the overall optimizing mechanism. The analyzer subtracts clusters and fuses the collected information to filter property parameters that reduce system service performance. The performance can be improved by realizing the nonlinear mapping relation of the system between "state" and "action", using a feedforward neural network and according to credible policies stored in a policy library. The system can then renew the value of Q by executing the learning algorithm on its own, after obtaining an environment reward-function value based on service availability and changes in system performance.

(3) Execution module: This module selects the optimal process in the system for retaining and improving dependability.

The three modules constitute an autonomic feedback control structure, in which a terminal network system is the protection target, and an autonomic optimizing model selects the optimal policy. Optimizing the service automatically according to the system performance state in every decision cycle reduces human interference and achieves dependability simultaneously.

3. Real-time optimization algorithm for network service performance

The autonomic optimization module determines the executive action that has the maximum probability value according to the mapping of environment-state property parameters to corresponding actions. System dependability can be improved in the long term with the maximum positive environment reward-function value achieved via improvements in service performance and availability, and via the executive learning algorithm that renews the value function Q by working out the optimal policy and implementing it in the system.

3.1 Method for selecting the "state–action" pair

The basic concept in selecting the "state–action" pair is that a three-layer feedforward neural network [8] can conduct any nonlinear mapping. This three-layer structure can be used to represent the probability distribution map of the system state S_t to the action $A(_t)$, which the state perception module monitors. If the first layer is assumed to be the input layer, where the number of neurons is the sum of the number of system states that the state perception and action variables perceive, and the input vector is *M-dimensional*, then $x=(s_1,s_2,\ldots,s_t,a_1,a_2,..,a_t)$ will be obtained. The number of neurons in the hidden layer is given by K, the value of which can be calculated via the empirical formula in Equation (1) and is proposed in Reference [9].

$$K=0.51+\sqrt{0.43MZ+0.12Z^2+2.54M+0.77Z+0.35} \quad (1)$$

Here, M is the number of neurons in the input layer, and Z is the number of neurons in the output layer (the number of actions). In using this formula, the values obtained

should be rounded up. The initial number of hidden-layer neurons can be determined via this formula and then adjusted slightly according to the actual situation.

The output value for a neuron in the hidden layer i can be expressed by Equation (2):

$$\rho_i=\vartheta(\varpi_j x-\varsigma_i^l)=\vartheta(\sum_{j=1}^{N}\varpi_{ij}x_j-\varsigma_i^l),i=1,2,...,K \quad (2)$$

Where ϑ is the excitation function, ϖ_{ij} is the link weight between the hidden neuron i and the input neuron j, and ς_k^i is the threshold value of hidden neuron i. The output value for output neuron k is as follows.

$$\Delta k(x)=\vartheta''(\omega_k\rho-\zeta_k^Z) \quad (3)$$

In this equation, ϑ'' is the excitation function, $\rho=(\rho_1,\rho_2,\ldots,\rho_N)$ is the output vector from the hidden layer, ω_{ki} is the link weight, and ζ_k^i is the threshold value. $\Delta_k(x)$ is the threshold value for output node k ($k=1,2,\ldots$), and Z is the probability value of action k. In this way, the action with the maximum probability value $\Delta_k(x)$ can be chosen.

3.2 Evaluation function constrained by dependable system performance parameters

The primary emphasis is given to network availability. Therefore, many parameters should be considered in ensuring service dependability for the target system when opening a network. In this paper, the information about the network state contained in the evaluation feedback function is more comprehensive because of the constraints of service availability and network performance parameters. To meet the required performance parameters of the target system, an environment reward R_i is defined as an evaluation function constrained by the performance of multiple data sources, with $R=\{R_1,R_2,\ldots,R_t\}$ being the set of the R_i that correspond to each executed action.

In this paper, the long-term and maximum environment reward can be obtained after a number of actions are executed to ensure system dependability. If a positive environment reward is obtained after executing an action, system dependability will be improved, or this trend will be declined. The function value of the environment reward is presented in Equation (4).

$$R_t=\sum_{j=1}^{m}\omega(t_j)A_s\bullet\Delta E_s \quad (4)$$

In this equation, $w(t_j)$ is the proportion of action j or policy in the target system, A_s is the availability of the system service, and ΔE_s shows the change in system service performance. Many factors determine system availability [10], including service computing resources such as the server CPU, state connection, memory occupancy rate, and load. Therefore, the calculation of system availability is as presented in Equation (5):

$$A_s=1-\prod_{i=1}^{m}(1-\omega_i\bullet A_i) \quad (5)$$

Where m is the number of computing resources that determine system service availability (four in most cases), A_i is the availability of computing resource i, and ω_i is the weight of computing resource i in the service availability of the system.

In most cases, parameters that reflect network service performance cover the response time for network service, mean transmission rate, and average delay. The computational formula is given by Equation (6).

$$\Delta E_s=\sum_{i=1}^{n}\omega_i\bullet\Delta E_i \quad (6)$$

In this formula, n is the number of parameters that reflect the changes in network service performance (three in most cases), ω_i is the weight of parameter i in service performance, and ΔE_i is the measure of change in the service performance before and after executing an action.

3.3 Algorithm description

The system can be optimized dynamically in real time based on the Q-learning control algorithm, after the method for selecting the state–action pair and reward function value has been determined. The main process of this algorithm is that the initial estimated value of $Q_t(s_t,a_t) - Q_{t+1}(s_{t+1},a_{t+1})$ is set to 0. The environment state s_t at each moment is then observed repeatedly. Action a_t is chosen and executed. Finally, according to the function value of the environment reward R, the system executes the learning algorithm autonomically, renewing the estimated value $Q_{t+1}(s_{t+1}, a_{t+1})$ and the system state $s_{t+1}\leftarrow s_t$. The formula for renewing the value function Q is as follows.

$$Q_{t+1}(s_{t+1},a_{t+1})=(1-\alpha)Q_t(s_t,a_t)+\alpha[R_t+\gamma\max_A Q_t(s_t,a_t)] \quad (7)$$

In this formula, $\gamma\in(0,1)$ is the time discount factor, $\alpha\in(0,1)$ is the learning rate, and a_{t+1} and s_{t+1} are the iteration action and state at the next moment, respectively. A higher value of α indicates faster convergence. However, a higher value is not necessarily better because an excessively high value of α may prevent convergence. Reference [11] proved that the maximum value function for each state will converge to a constant value after iteration, and has identified when α can meet a number of requirements.

$$\max_{a\in A}Q(s_t,a_t)\rightarrow E\{\sum_{n}^{\infty}\gamma^{n-1}R_{n+1}\} \quad (8)$$

The steps in executing the algorithm are as follows.

Step 1: State set $S_t=\{s_1,s_2,\ldots,s_m\}$ and action set $A_t=\{a_1,a_2,\ldots,a_t\}$ are inputted. $Q_{t+1}(s_{t+1},a_{t+1})\leftarrow 0$ is initialized.

Step 2: Schedule action a_t, with the method-of-action probability distribution being selected and executed.

Step 3: The reward function value R can be obtained by using Equation (4).

Step 4: The Q function value is renewed using Equations (3) and (4).

These four steps are executed repeatedly. When $Q_t(s_t,a_t)$ reaches the maximum value, Step 5 is executed.

Step 5: $\max_{a\in A} Q_{t+1}(s_{t+1},a_{t+1})$ is determined and $s_{t+1}\leftarrow s_t$ is renewed.

4. Simulation and analysis of results

By using the simulation tool NS2, the maintenance effects of overall system dependability and of the service utility during the operation of the network were simulated and compared with and without the optimization algorithm.

The system structure is presented in Fig. 2. A client sends a service request to the server through a local area network (LAN). Two groups of experiments were designed, using a traditional server with and without the addition of an optimizing model. In the simulation, 2,000 servers were deployed randomly in the LAN to act as service resources. The daily service-failure rate was set at 2.3%, following the observations in Reference [12].

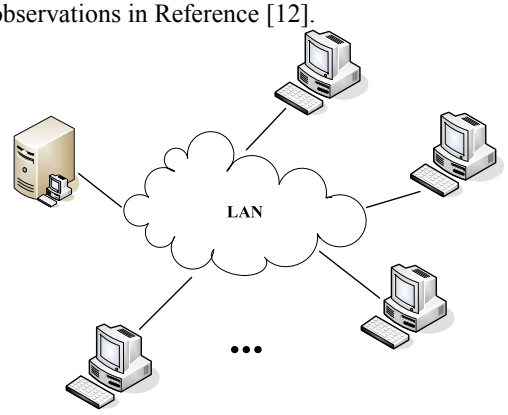

Fig. 2 Experiment environment.

The first group of simulations aimed to test the effects of the autonomic optimization model on overall system dependability. Figure 3 presents the average dependability of the system service after 90 days of operation. Figure 3(a) shows the maintenance effect of a traditional server on performance dependability while the system was operational. Figure 3(b) presents the maintenance effect after adding the autonomic optimizing model.

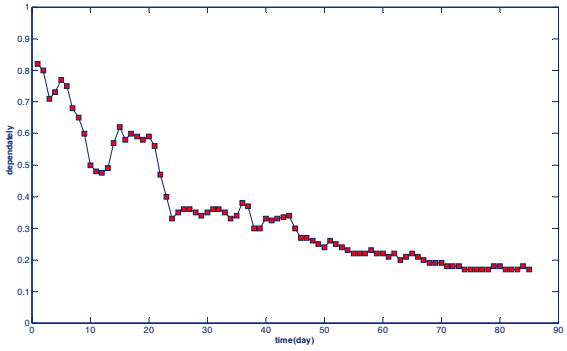

Fig. 3(a) Maintenance effect of a traditional server on system service dependability.

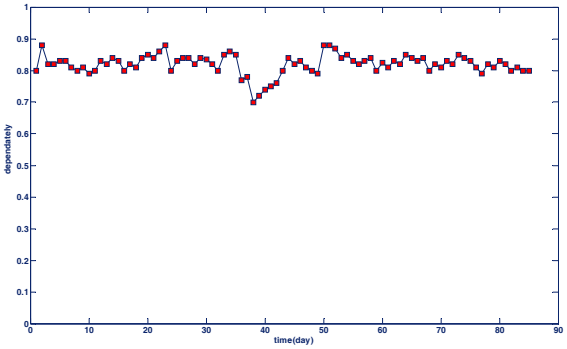

Fig. 3(b) Maintenance effect of a traditional server with an autonomic optimizing model on system service dependability.

A comparison of Figs. 3(a) and 3(b) reveals that the dependability of the traditional server declines noticeably. On the other hand, that of the server with the autonomic optimizing model is better overall, fluctuating less and remaining at a high level.

The second group of simulations aimed to analyze the effects of the autonomic optimizing model on system service dependability before and after adding the model. Figure 4 presents the average utility of a LAN composed of 2,000 servers and 10,000 requests after 90 days of operation. Figure 4(a) presents the effect of a traditional server on utility, and Fig. 4(b) presents the effect on the utility of a server with the autonomic optimizing model.

A comparison of Figs. 4(a) and 4(b) demonstrates that, together with a decline in dependability, the utility also declines when using a traditional server. The addition of the autonomic optimizing model preserves the system dependability because it optimizes the environment via the autonomic optimizing control process, thereby stabilizing its utility.

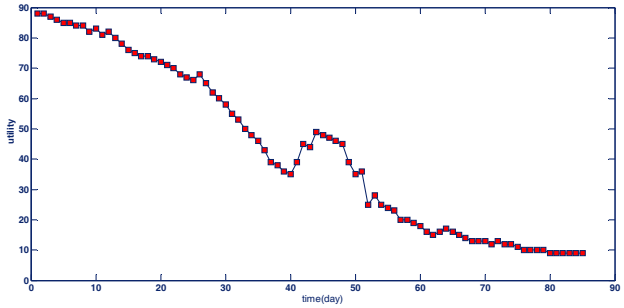

Fig. 4(a) Maintenance effect of a traditional server on the utility of the system service.

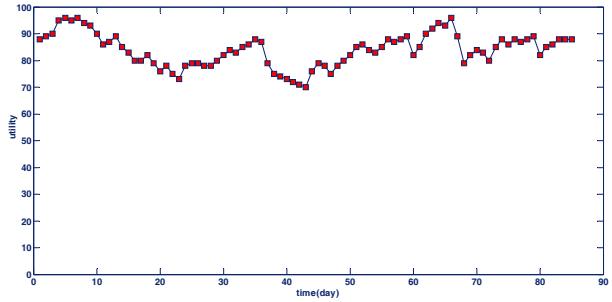

Fig. 4(b) Maintenance effect of a server with an autonomic optimizing model on the utility of the system service.

5. Conclusions

The complexity of computing management systems has increased dramatically. As a result, ensuring dependability in complicated structures is difficult when using traditional servers. In this paper, an autonomic method for optimizing network service dependability based on Q-learning is proposed. This method can help retain service dependability by periodically optimizing the environment state immediately after collecting relevant characteristics of a target system and calculating an approximately optimal function value *Q* according to the state–action pair and environment reward-function values. Simulation results show that the overall dependability and service utility of a system are better if an optimizing model is included in the system. The selection of key state parameters will be investigated further in future research.

Acknowledgments

The authors thank the anonymous reviewers for their valuable comments and suggestions. This work is sponsored partially by the National Natural Science Foundation of China (No. 61003035 and 61142002) and the Plan for Scientific Innovation Talent of Henan Province (No. 124100510006)

References

[1] J Banks, E Crow, “Embedded Diagnostics Enable Military Ground Vehicle Autonomic Logistics”, Reliability and Maintainability Symposium (RAMS’07), 2007, pp. 48-52.
[2] L Tosiyasu, Kunii, “Autonomic and Trusted Computing For Ubiquitous Intelligence”, International Conference on Autonomic and Trusted Computing, 2007, pp. 1-12.
[3] IA Troxel, AD George, “Adaptable and Autonomic Mission Manager for Dependable Aerospace Computing”, Symposium on Dependable, Autonomic And Secure Computing, 2006, pp. 11-18.
[4] M Hinchey, “Modeling for NASA Autonomous Nano-Technology Swarm Missions and Model-Driven Autonomic Computing”, The 21st International Conference on Advanced Information Networking and Applications (AINA’07), 2007, pp. 250-257.
[5] B Andrea, B Alfredo, P Paolo, “A Dependable Autonomic Computing Environment for Self-Testing of Complex Heterogeneous Systems”, Electronic Notes in Theoretical Computer Science, Vol. 116, 2005, pp. 45-57.
[6] S Perter, P Narasimhan, J Wilkes, “Prato: Databases on demand”, The 4th International Conference on Autonomic Computing (ICAC’07), 2007, pp. 11-15.
[7] YU Tao, ZHEN Wei-guo, “Optimal Control Method of PSS Based on Multi-Step Backtrack Q(λ) Learning”, Power System Protection and Control, Vol. 39, No. 3, 2011, pp. 18-23.
[8] LIU Hai-tao, HONG Bing-rong, PIAO Song-hao, “Evolutionary Algorithm Based Reinforcement Learning in the Uncertain Environments”, Acta Electronica Sinica, Vol. 34, No. 7, 2006, pp. 1356-1360.
[9] XU Ming-liang, XU Wen-bo, “Q-learning Control Based on Self-Organizing RBF Network”, Control and Decision, Vol. 25, No. 2, 2010, pp. 303-306.
[10] LIANG Ying, “Mission-Critical Network Systems Security Posture Formal Modeling and Analysis for Service-Oriented”, Ph.D. Thesis, Department of Computer Science, Harbin Engineering University, Harbin, China, 2009.
[11] HONG Yuan, Liu Zhen, “Modeling Adaptive Perception System of Virtual Human Based on Q-Learning”, Journal of Computer Applications, Vol. 31, No. 1, 2011, pp. 78-81.
[12] GUO Hui-Peng, HUAI Jin-Peng, DENG Ting, “A Dependable and Adaptive Approach to Supporting Web Service Composition”, Chinese Journal of Computers, Vol. 31, No. 8, 2008, pp. 1434-1443.

Qingtao Wu got his PhD degree in computer science from East China University of Science and Technology, in 2006, on network and information security. He’s Associate Professor in Computer Science at Electronic & Information Engineering College of Henan University of Science and Technology, China. He’s currently managing and leading 2 projects supported by the National Natural Science Foundation of China to address the autonomic mechanism for the retainment and enhancement of system security. His main research interests include computer system security, intelligent information processing, etc.

Lina Zhu received her Bachelor's degree in Computer Science and Technology in 2010. She is currently a Master Degree Candidate directed by Dr. Qingtao Wu in Computer Science at Electronic & Information Engineering College of Henan University of Science and Technology, China. Her research is focused on the network security.

Ying Lou got his PhD degree in computer science from Northwestern Polytechnical University, in 2011, on web service and security. He's Associate Professor in Computer Science at Electronic & Information Engineering College of Henan University of Science and Technology, China. His main research interests include information service, intelligent information processing, etc.

Ruijuan Zheng got her PhD degree in computer science from Harbin Engineering University, in 2008, on autonomic system security. She's Associate Professor in Computer Science at Electronic & Information Engineering College of Henan University of Science and Technology, China. Her main research interests include computer system security, network security, etc.