PHM Technology for Memory Anomalies in Cloud Computing for IaaS

Xiwei Qiu[a], Yuanshun Dai*[a], Peng Sun[a,b], and Xin Jin[a]
[a]University of Electronic Science and Technology of China, Chengdu, China
[b]Hebi NLED Company Limited, Zhengzhou, China
(*corresponding author: Yuanshun Dai, email: ydai@uestc.edu.cn)

***Abstract*—The IaaS (Infrastructure as a Service) is one of the most popular services from todays cloud service providers, where the virtual machines (VM) are rented by users who can deploy any program they want in the VMs to make their own websites or use as their remote desktops. However, this poses a major challenge for cloud IaaS providers who cannot control the software programs that users develop, install or download on their rented VMs. Those programs may not be well developed with various bugs or even downloaded/installed together with virus, which often make damages to the VMs or infect the cloud platform. To keep the health of a cloud IaaS platform, it is very important to implement the PHM (Prognostics and Health Management) technology for detecting those software problems and self-healing them in an intelligent and timely way. This paper realized a novel PHM technology inspired by biological autonomic nervous system to deal with the memory anomalies of those programs running on the cloud IaaS platform. We first present an innovative autonomic computing technology called *Bionic Autonomic Nervous System* (BANS) to endow the cloud system with distinctive capabilities of *perception*, *detection*, *reflection*, and *learning*. Then, we propose a BANS-based Prognostics and Health Management (BPHM) technology to enable the cloud system self-dealing with various memory anomalies. AI-based failure prognostics, immediate self-healing, self-learning ability and self-improvement functions are implemented. Experimental results illustrate that the designed BPHM can automatically and intelligently deal with complex memory anomalies in a real cloud system for IaaS, to keep the system much more reliable and healthier.**

***Index Terms*—reliability, prognostics and health management, artificial intelligence, cloud computing, memory anomaly, Infrastructure as a Service.**

I. Introduction

Nowadays, cloud systems become increasingly large and complex for integrating a great number of heterogeneous computing resources to support various science, engineering and commercial applications. Since the cloud system plays an irreplaceable role in supporting internet-based applications, how to guarantee the reliability of the cloud system is indeed a crucial issue [1]. Due to large scale and high complexity of the cloud system, failures in a cloud system are difficult to be diagnosed and removed in a timely manner by using some traditional and manual mechanisms, for example, test-debugging approaches and fault detection and isolation (FDI) tools [2][3]. Therefore, failure prognostics and autonomic health management would be indispensable for a large and complex cloud system [4].

Prognostics and Health Management (PHM) is an efficient technology that performs modeling, assessment, and improvement of reliability under actual conditions of a complex system. The PHM has been successfully adopted in various IT systems for ensuring the system to be reliable in real time [5], especially for large-scale network systems, wireless sensor systems [6], and smart grid systems [7]. Two important advantages of PHM can be summarized as: providing early warning on failures and maintaining effectiveness through timely maintenance actions [8], that is, prognostics of failures and management of system health. It is difficult but obviously meaningful to endow a large-scale cloud system with such two key functions [9].

In a large-scale cloud system, varieties of programs and applications can be delivered by cloud users to the cloud system. These programs and applications are executed in virtual machines (VM) for realizing non-interfering share of heterogeneous computational resources [10]. For example, Amazon provides Infrastructure-as-a-Service (IaaS) (*e.g.*, Amazon EC2 and Amazon Elastic Container Service [11]) for cloud users. Then, the cloud users rent VMs from Amazon for running their personal programs and applications [12][13]. However, it is impossible that the cloud system tests or verifies the correctness of all these programs and applications in advance. Therefore, in realistic environment, a VM will inevitably encounter various failures caused by running user programs and applications. In a practical scenario, VMs must have some abnormal conditions before the final failure, which are usually regarded as unusual resource use, *i.e.*, abnormal changes in CPU, memory, disk and network metrics.

Memory anomalies are a common category of failures that occurs frequently in the users' applications running in the VMs of a large-scale cloud system. It is because when users develop the applications running on the VMs they don't care too much about memory leakage for the VMs are maintained not by them but by the cloud service providers. Recent researches on memory anomaly detection for the cloud system focus on statistics analysis of memory metrics [14][15], such as exploring metric space of a VM [16], or finding change trends of memory usages for programs and applications hosted on a VM [17]. However, these existing researches ignore an important fact that some memory anomalies not only lead to unusual memory usages but also result in some correlated abnormal changes in other metrics (*e.g.*, the CPU metric and

978-1-7281-8913-0/20/$31.00 ©2020 IEEE
DOI 10.1109/QRS51102.2020.00018

the network metric) [18]. This is because that there exist a kaleidoscope of causes of memory anomalies. For example, if an application running in a VM has a memory anomaly of memory leak, it may only affect the memory metric of the VM [14]. However, if it has been infected by a computer virus, abnormal memory usage definitely exists due to the virus must run in the memory [19], and it also has complicated effects on the CPU, disk, and network metrics depending on specific actions of the virus.

Therefore, due to the large-scale cloud system running various user programs and applications on the host, the memory exception problem becomes more serious and severe than ever before. Such problem cannot be effectively solved by using some traditional methods, because the traditional methods first need to use debugging [20], error location [21] or some specific detection tools [22] to find the specific causes of memory exceptions. However, this action is truly time-consuming, and thus it is very inefficient or even unrealistic from the perspective of the cloud system. More importantly, the cloud system has no responsibility or access to remove bugs, errors, or faults of all users' programs and applications without original codes. The cloud system can only manage its own health to prevent serious damage from unknown memory anomalies.

The PHM technology provides an efficient approach to manage the health of a system in real time. For the large-scale cloud system, the PHM also needs autonomic computing [23] and self-learning abilities to guarantee the health of the cloud system (*i.e.*, an expected normal system condition) against varieties of memory anomalies. To realize an intelligent PHM technology for the cloud computing, in this paper, we first integrate an innovative autonomic computing technology called *Bionic Autonomic Nervous System* (BANS) with the PHM technology. Today's large-scale cloud systems are just like a human but without an autonomic nervous system, and thus the BANS can endow the systems with powerful capabilities of self-management and self-improvement. After building up the BANS-based PHM (BPHM) structure, we further adopt the machine learning to build failure models for VMs, which can be used to realize the prognostics function of the cloud system. The deep learning technique is also studied to realize a self-healing module that recovers various memory anomalies timely. Finally, we propose a self-healing module in BPHM, which is updated by using reliability model and reinforcement learning (RL) technology to improve the accuracy of self-healing in real-time running cloud. Then, the BPHM can be very effective, which perceives complex memory anomaly phenomenon, detects failures in advance, reflects a healing plan timely, and learns knowledge to improve the accuracy of the BPHM.

Section 2 describes the design of the PHM system merging with our new biologically inspired technology BANS for dealing with the memory anomaly problem in the cloud system. Section 3 presents the implementation of the BPHM, where health index monitor, intelligent failure prognostics, self-healing and self-learning abilities are integrated in a holistic manner. Section 4 shows the experimental results of our researches. Section 5 is the conclusion and future works.

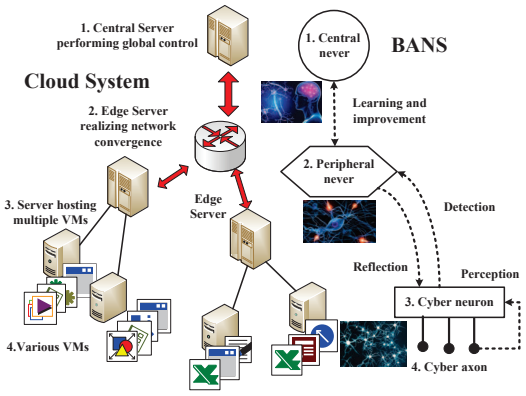


Fig. 1. Combination of the BANS and the cloud system.

II. Design of the BANS-based PHM

A. Distinctive capabilities of the BANS

Bionic Autonomic Nervous System (BANS) is a biologically-inspired technology. It is analogous to human autonomic nervous systems that work in an unconscious manner (*e.g.*, conditional reflection). Such an autonomic nervous system is indeed crucial and indispensable for keeping our body functioning well.

We should note that the autonomic nervous system have some unique and distinctive capabilities, particularly, *perception*, *detection*, *reflection* and *learning*. Take that a child accidentally touches a flame as an example. In the autonomic nervous system, skin temperature perceived by axons is transmitted to the neuron in real time. The axon's perception works in all times but most normal perception does not incur the reflection except some stimulus, such as the flame touch in this example. When the neuron detects this abnormal and harmful situation, it immediately sends an electrical signal to the peripheral nerve. The peripheral nerve is now triggered to take a conditional reflection (*i.e.*, stress action) of removing the finger away from the flame as soon as possible. Note that such conditional reflection is executed in a totally autonomic manner without any control of brain. At the same time, the peripheral nerve also transmits the information to the central nerve, which is connected with the human brain to learn experience or knowledge from events. Finally, with the irreplaceable help of the autonomic nervous system, the child will never deliberately touch the fire again (*i.e.*, self-learning and improvement).

The BANS is a novel technology that imitates such a human autonomic nervous system with those distinctive capabilities, and the BANS consists of four basic modules: the cyber axon, the cyber neuron, the peripheral never and the central never. The BANS could be a prefect remedy of the lack of an efficient autonomic management system for the large-scale cloud system, as shown in Fig. 1.

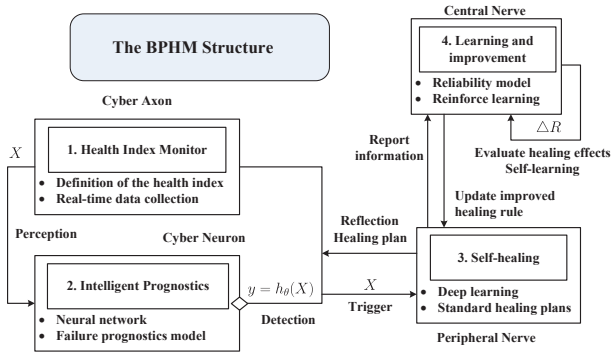


Fig. 2. The design of BPHM structure.

As shown in the figure, servers in the cloud system hosts various VMs. Each VM runs multiple perception threads that monitor operational conditions of the VMs in real time, which can be treated as cyber axons in the BANS. Meanwhile, cyber neurons have detection functions resident in the VM to determine whether an abnormal stimulation occurs. Once the cyber neuron detects an abnormal situation, it creates a signal for triggering the peripheral nerve, and also transmits the abnormal perception information to the peripheral never. The peripheral nerve connects the host server with the edge server. The edge server works for network fusion in the cloud system. The peripheral never has an important reflection function, which can immediately reflect a signal to the physical server for taking a quick stress reaction to prevent the server suffering from a severe consequence. This reflection procedure works in an autonomic way without the participation of central servers in the cloud system. In fact, the conditional reflection action can be further improved. That is, the peripheral nerve also transmits the information about perception and reflection to the central never (*i.e.*, central servers in the cloud system) for self-learning and improvement. The central nerve is capable of self-learning to derive useful experience or knowledge from the information reported by the peripheral nerve, and finally coordinates the peripheral nerve to achieve intelligent improvement.

B. BANS-based PHM Structure

To realize an autonomic and intelligent PHM system for solving the memory anomaly problem existed in the cloud system, the BANS should be customized with specific functions of the PHM system. That is, a deliberately designed BANS-based PHM (BPHM) structure. As shown in Fig. 2, the presented BPHM structure consists of four important modules corresponding to the cyber axon, the cyber neuron, the peripheral nerve, and the central nerve in the BANS. The design of these modules are described as follows.

1) Health index monitor (by the cyber axon): the cyber axon associated with the cyber neuron is realized as the health index monitor in the BPHM structure. The health index monitor is a resident process running in all VMs. It has multiple threads imitating cyber axons that perceive various health metrics. Now, these threads glean various real-time metrics to perceive the running condition of the host VM. The gleaned metrics include resource usages of CPU, memory, disk and bandwidth. Meanwhile, the cyber axon has a quantification of the health index derived from those metrics for the VM (suppose it is X). The health index is designed to comprehensively cover not only real-time data but also some historical information, such as mean values and max values of some metrics over a related short time period. Having the quantification of the health index, the VM-resident health index monitor transmits the health index ($X = \{x_1, x_2, \ldots, x_N\}$) to the cyber neuron in real time.

2) Intelligent prognostics (by the cyber neuron): the cyber neuron is designed as an intelligent prognostics module in the BPHM structure. The cyber neuron runs in the server to realize failure detection of its VMs. The failure detection function is derived by training a neural network (NN). The NN takes perceived health indices as the input and takes a signal representing whether a memory anomaly occurs as the output. The NN has the capability of machine learning, and thus can finally give us a failure detection function $y = h_w(X)$, where w and y are the optimal parameter set and the output of the NN, respectively. Now, the cyber neuron can perform the failure prognostics for various memory anomalies by using $y = h_w(X)$. Once the cyber neuron identifies an abnormal health index, it immediately generates a signal of exceptional occurrence and then triggers the peripheral nerve. This procedure just like the biological neuron conducts an electric signal from a stimuli.

3) Self healing (by the peripheral nerve): the biological peripheral nerve is responsible for creating an immediate reflection for executing a stress action in an unconscious manner. Therefore, the peripheral nerve in the BPHM also needs have two similar capabilities: self-healing imitating autonomic reflection, and some preset healing plans resembling unconscious stress actions. We can adopt a deep learning (DL) network to build up such a self-healing system to fulfill immediate healing. The inputs of the DL network are exceptional health indices, and the output of the DL network is a corresponding healing plan k ($k = 1, \ldots, K$) that is selected from K preset healing plans defined in advance. After training the deep learning network, the self-healing module can perform the conditional reflection that gives the immediate healing plan k for the detected failure. Meanwhile, the peripheral never also reports the information about the self healing to the central nerve for supporting further learning and improvement.

4) Self Improvement (by the central nerve): In biology, the brain and spinal cord make up the central nerve, which mainly takes charge of activity improvement. Therefore, the central server in the cloud system can be treated as the central nerve in the BPHM structure. As shown in Fig. 2, the central nerve have two key functions of learning and improvement. The reinforce learning (RL) is a powerful technique can achieve the goal of self-learning and self-improvement. The

objective of the RL is to improve the accuracy of the self-healing. This is because that the reflected healing plan k given by the self-healing may not be the most accurate one for the detected failure due to complex memory anomalies. Hereby, we use reliability models to computer an error $\triangle R$ to train the RL model. That is, after receiving the information reported by the peripheral never, the central nerve use the information to derive reliability metrics R' that quantifies the system reliability after healing. Then, the central nerve computes an error as $\triangle R = 1 - R'$, where '1' represents the idealistic condition of the cloud system without any failures. Now, $\triangle R$ can be treated as a deviation between the health condition after executing the reflected healing plan k and the perfect health condition of the cloud system. Although such perfect health condition is hard to achieved, the RL will try to make the deviation smaller and smaller through the self-learning, that is, keep improving the accuracy of the self-healing.

III. Realization of the BPHM

A. Health Index for Perception

Since cloud users rent VMs from the cloud system for running their personal applications, there may exist unknown possibilities of causing memory anomalies, *e.g.*, program bugs, software faults, viruses and malware. In principle, the cloud system cannot access the original codes of the users applications to find specific causes of memory anomalies. Therefore, running state of the rent VMs should be perceived in real time for preventing failures. Here, we give a vector named as the health index to describe the state of the VM. The design of the health index satisfies the following principles.

1) *Observability*: the health index must include multiple observable metrics. These metrics can be observed, captured, and analyzed in real time, which like human body condition that can be perceived by axons anytime.

2) *Failure correlation*: a failure definitely has a negative effect on the system health. Therefore, the health index has important correlation with the failure. That is, if a VM has a memory anomaly, the health index should be able to reveal some correlated and abnormal changes that potentially implies a failure may happen.

3) *Health-related*: the metrics in the health index should be health-related to provide valuable information for supporting the follow-up self-healing and self-improvement in the BPHM.

Thus, the i-th cyber axon can monitor the i-th health index at any time t, to obtain $x_i(t)$. Suppose there are a total number of N health indices fulling the above three conditions. Then, there should be N cyber axons to monitor their corresponding health index at any time t, generating a set of $X(t) = \{x_1(t), x_2(t), \ldots, x_N(t)\}$, at a certain time t. $X(t)$ will become the input for the following intelligent prognostics by the corresponding cyber neuron.

B. Intelligent Prognostics for Detection

There are two important steps to realize the intelligent prognostics to detect whether a failure happens according to gleaned health indices, which is presented as follows.

Step 1. Train a neural network to learn the failure prognostics: To endow the cyber neuron with the capability of the intelligent prognostics, we can adopt the NN to learn the knowledge about the failure prognostics. The NN has been widely applied into many fields that require artificial intelligence. The training set of the NN is a batch of datasets including two parts of data, *i.e.*, normal health indices with a label of '0' and abnormal health indices with a label of '1'. The abnormal health indices are gleaned from some actual memory anomaly problems that are observed before. Suppose the input of the NN is $\mathbf{X}$, that is,

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)}, & x_2^{(1)}, \ldots, & x_N^{(1)}, y^{(1)} \\ x_1^{(2)}, & x_2^{(2)}, \ldots, & x_N^{(2)}, y^{(2)} \\ & \ldots & \\ & \ldots & \\ x_1^{(M)}, & x_2^{(M)}, \ldots, & x_N^{(M)}, y^{(M)} \end{bmatrix} \tag{1}$$

where $y^{(m)}$ ($y^{(m)} \in \{0, 1\}, m = 1, 2, \ldots, M$) is the label of the mth sample ($m = 1, 2, \ldots, M$), and $X^{(m)} = [x_1^{(m)}, x_2^{(m)}, \ldots, x_N^{(m)}]$ is the health index of the mth sample in $\mathbf{X}$. Now, the NN gives a failure prognostics model (*i.e.*, detection function) of $y = h_w(X)$, where w is the parameter sets of the NN. Now, for a health index X gleaned real time, the output y is derived as

$$y = \begin{cases} 1, & h_w(X) \geq 0.5 \\ 0, & h_w(X) < 0.5 \end{cases} \tag{2}$$

For deriving the failure prognostics model with an optimal parameter set w, the cost function of the NN is given as

$$C(w) = \\ -\frac{1}{M}\left(\sum_{m=1}^{M} y \cdot \log h_w(X) + (1-y) \cdot \log\big(1 - h_w(X)\big)\right) \tag{3}$$

and we can use a gradient descent algorithm to derive w for $\min_w C(w)$ by iterating

$$w_j = w_j - \alpha \cdot \frac{\partial}{\partial w_j} C(w), \text{ for all } w_j \in w \tag{4}$$

where α is a learning rate of the NN.

Step 2. Detect a failure and trigger the successive self-healing: Now, the cyber neuron has the intelligent failure model to sense stimulus (*i.e.*, various memory anomalies). The real-time health index $X(t)$ gleaned by the cyber axon is transmit to the cyber neuron as the input of $y(t) = h_w(X(t))$. Once the cyber neuron find that the output becomes $y(t) = 1$, it immediately transmit a signal to the peripheral nerve for triggering the self-healing action.

C. Consequence-oriented Self-Healing for Reflection

After receiving a trigger signal from the cyber neuron, the peripheral nerve needs to reflect a healing plan for dealing with

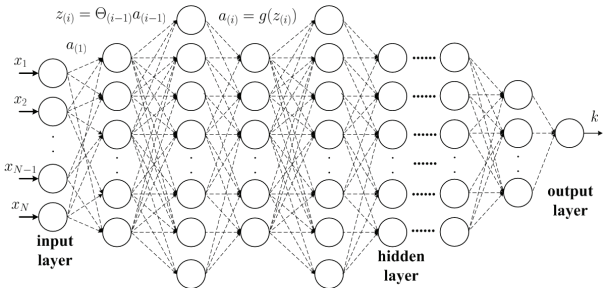


Fig. 3. Structure of the deep learning network.

the failure timely. We call this procedure as the consequence-oriented self-healing. Since the self-healing should give a healing plan immediately to alleviate consequence that may be caused by the faults, appropriate healing plans must be preset in advance. Here, the healing plans are designed from the consequence-oriented perspective of the cloud system, which are different from those that focus on program debugs, logical errors, or software faults in users' applications. Some principles to design the consequence-oriented healing plans are listed as follows.

1) Not to stop the program, or change the codes, nor recompile it. This is because the cloud IaaS providers cannot access the software codes that users develop, install or download on their rented VMs.

2) Not to reboot the physical server (host machine) when encountering problems in a virtual machine, thereby minimizing the effect on other ongoing tasks and processes. This is very important for the cloud IaaS, since there are other cloud users VMs are hosted on the same physical server.

3) Even though the consequence-oriented healing plan cannot remove bugs or recompile the program, it should has a significant effect on restraining/alleviating the consequence/damage of the occurred memory anomaly. For example, limiting the upper memory usage of the process with a slight memory leak.

4) The healing plan should consider the generality which can fit the heterogeneity of memory anomalies existed in various users programs and applications.

After designing the healing plans satisfying the above requirements, the reflection rule can be determined to match them to the corresponding memory anomalies. For example, the reflection rule can be set according to certain categories of the memory anomalies.

In Fig. 3, $z_{(i)}$ and $a_{(i)}$ are the input vector and the output vector of layer i, and $z_{(i+1)} = \Theta_{(i)} a_{(i)}$ gives the transform form layer i to layer $i+1$. Function $a_{(i)} = g(z_{(i)})$ is an activation function applied to $z_{(i)}$, such as a sigmoid function $g(z) = 1/(1+e^{-z})$. This will give us activation for hidden layer i. Finally, the output layer gives k as the output of the DL network. The DL network uses the gradient descend with momentum and adaptive learning rate algorithm [24] to perform convergent iteration.

After training the DL network, we can have the self-healing function of $k = \pi_\Theta(X)$, where Θ is an optimal parameter set of the DL network, and X is an exceptional health index transmitted from the cyber neuron. As soon as the peripheral nerve receives the exceptional health index X, it can immediately reflect a healing plan k $(k = 1, 2, \ldots, K)$ to make a stress action. Finally, the peripheral nerve also reports X, k and $\pi_\Theta(X)$ to the central nerve for supporting the self-learning and self-improvement.

D. Cloud Service Reliability Modeling for IaaS

Since the reflection rule of the self-healing module during the training process is to learn some historical training data, it may not be the most accurate rule for healing various memory anomalies in a new cloud platform. Therefore, a reinforce learning for self-improvement module is essential and crucial for the BPHM, which can increasingly coordinate and update the self-healing for a better accuracy to adapt to the new cloud platform. This module embedded into central nerves should fulfill two important functions: quantifying healing effect of a reflected plan and learning knowledge from this healing action.

Thus, we first present an effective approach to quantify the healing effect of a reflected healing plan given by the self-healing module, that is, a reliability-related metric $\triangle R$. To derive such an evaluation metric, a cloud service reliability model for the IaaS system need to be built up first. As the cloud IaaS system provides VMs to users, we make the following assumptions for modeling the cloud service reliability.

1)VMs created by the same template of a cloud system are homogeneous with an identical failure rate. The failures of those VMs follow the exponential distribution with the constant parameter λ_0, which has been widely accepted in reliability area, see *e.g.* [25][26].

2)Once a VM is assigned to a cloud user, the failure rate of the VM becomes different. This is because that the user's personal applications hosted on the VM have some inevitable effects on the failure rate of the VM. Suppose the cloud system assigns N VMs to different users, and thus the failure rates of these VMs are $\lambda_1, \ldots, \lambda_n, \ldots, \lambda_N$, respectively.

3)The cloud system also runs M hot-standby VMs for guaranteeing service reliability of the cloud system. The failure rates of those hot-standby VMs are λ_0. Once a VM assigned to the cloud user is failed due to a failure, the cloud system migrates the user's applications to a hot-standby VM, and the number of the hot-standby VMs becomes $M-1$. The live migration between VMs is technologically feasible in cloud computing, see *e.g.* [27].

4)If a hot-standby VM is failed due to failures of itself, M also changes to $M-1$.

Given the above assumptions, such a cloud IaaS system can be treated as a k-out-of-n system [28], and states of this system can be modeled by a Markov process, as the following Fig. 4.

State m $(m = 1, 2, \ldots, M)$ represents that there are m hot-standby VMs remaining in the cloud system, and state -1 means that the cloud service failed because at least one necessary VM required by the N users is unavailable. For

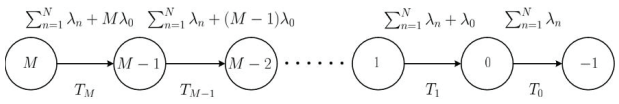


Fig. 4. Markov model for the k-out-of-n cloud system.

$m = M, M-1, \ldots, 1$, there are two events result in that the state transmits from m to $m-1$. One is that a VM assigned to the cloud user has a failure, and thus the user's application is migrated to a hot-standby VM. Another is a hot-standby VM is failed due to failures of itself. Therefore, the state transmits from m to $m-1$ with the rate of $\sum_{n=1}^{N}\lambda_n + m\cdot\lambda_0$. As for state 0 shown in the figure, there is no hot-standby VM anymore. Therefore, the system transmits from 0 to -1 with the rate of $\sum_{n=1}^{N}\lambda_n$. Let random variable T_m $(m = M, M-1, \ldots, 1, 0)$ represent the random time of the transition from state m to $m-1$. The one-step transition probability from state m to $m-1$ during time interval t can be derived as

$$\begin{aligned} F_m(t) &= \Pr\left(T_m < t | S(0) = m\right) \\ &= 1 - \exp\left(\left(-\sum_{n=1}^{N}\lambda_n + m\lambda_0\right)\cdot t\right),\ 0 < t < \infty \end{aligned} \tag{5}$$

where $\{S(t), t \le 0\}$ represent the stochastic model shown in Fig. 4.

Now, suppose T is the random time for the cloud system transmits to state -1. The cumulative distribution function (CDF) of T can be derived by

$$F(t) = F_M(t) * F_{M-1}(t) * \cdots * F_m(t) * \cdots * F_0(t) \tag{6}$$

where '*' denotes the Stieltjes convolution of two functions. Having the CDF of T, we can give a service reliability of the cloud system as

$$R(t) = 1 - F(t) \tag{7}$$

Now, different healing plans change parameters of the reliability stochastic model. For example, for the standard healing plans of limiting the upper memory usage and suspending a suspicious progress, they have positive effect on decreasing the failure rate of the VM. As for the healing plan of the VM migration, it also change the failure rate of the host server to a new value. For the other healing plans, (*i.e.*, the VM rollback and the VM reboot), VM failure rate λ also becomes different. More details of the stochastic models related to the failure recovery for the cloud system can be found in our prior research [29][30].

E. Reinforce Learning for Self-Improvement

The BPHM has a program to estimate parameters of the service reliability model of the cloud system. The parameters of the model for a healing plan can be estimated by using some statistical method, such as the maximum likelihood estimation (MLE). Once the central nerve receives information about a conditional reflection for a memory anomaly from the peripheral nerve, it use the information to train the reinforce learning network. The central nerve first updates the parameters in the service reliability model to derive the reliability metric $\triangle R$. Then, $\triangle R$ can be inputted into the reinforce learning network, and the object of the reinforce learning is to minimize error $\triangle R$. After performing the self learning, the central nerve update the self healing module for realizing self improvement. The reinforce learning technique is shown in Fig. 5.

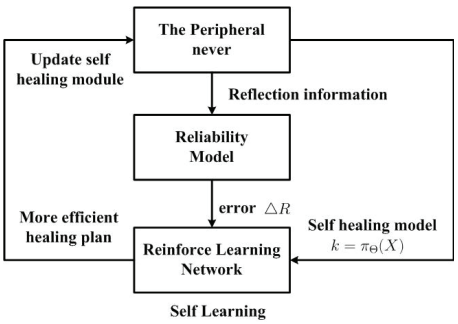


Fig. 5. Reinforce learning for learning and improvement.

As shown in the figure, after receiving the report information (*i.e.*, the reflection rule, the reflected plan, the exceptional health index, and so on) from the peripheral never, the central nerve first updates the parameter of the reliability model (changes of the parameters may be slight, but it is not a problem since the reinforce learning is a continuous and real-time process), and then compute the error for the reflected healing plan. This error and the report information become the inputs of the reinforce network. Then, after training the reinforce network with the important self-learning capability, a more efficient healing plan is derived, and the central nerve also updates the self-healing model and the reflection rule in the peripheral nerve to realize self-improvement.

The reinforce learning network can be realized by using an Actor-Critic algorithms [31][32], where the service reliability model is the critic of the reinforce learning network, and the self-healing model with the reflection rule is the actor of the reinforce learning network.

IV. Implementation and Case Study

A. Environment of the Case Study

This is a real case study running on a cloud computing system that provides a typical IaaS for VMs leased for teaching and research in our university. Users of this cloud IaaS system, including students, teachers, and researchers, can request a VM for running their personal applications to do research or teach. For example, a student can build a big-data testing environment using VMs provided by our IaaS cloud system. A teacher can also preset software demos in a VM for teaching. Meanwhile, a researcher can deploy a complicated development environment among multiple VMs (*e.g.*, a Hadoop system or a TensorFlow AI system). Then, they can develop and test new techniques and do scientific

experiments by utilizing the computational resources of our IaaS cloud.

The infrastructure of our cloud system consists of several hundred physical servers. The central servers and the edge servers are deployed with Inventec cloud controller servers K900 and Asus GPU servers ESC4000 G3, respectively. The computing resource pool of the cloud system are deployed with plenty of servers that have Intel Xeon CPU E5-2407 (4×2.2 GHz), 96 G memory, and 2TB disk each.

In this case study, we choose 40 identical physical servers, where 8 VMs are hosted on one physical server. Among the 320 VMs, we set up 200 VMs to simulate different usages for teaching and research by running corresponding programs, and prepare 100 hot standby VMs for backups. Each VM is assigned 4G memory, 2 CPU core, and 128G disk. The remaining 20 VMs are reserved for the real-time reliability evaluation and self-learning function of central nerves.

We have already implemented the proposed BPHM for our cloud IaaS system. Each VM has one cyber neuron for prognostics function running in it as a process and multiple cyber axons running as threads to monitor the health indices. Each physical server (host machine) has one peripheral nerve for self-healing function running in it as an OS-level program. There are 20 VMs particularly work as central nerves for reliability modeling and self-learning function with the implementation by the open-source TensorFlow. Historical data gleaned by our cloud IaaS system is used to train the failure prognostics module and the self-healing module in the BPHM.

In this experiment, we have a fault generator to randomly generate faults among different types of the memory anomalies, and then inject the fault into a randomly selected VM. The fault generator follows a Poisson distribution. Then we can do the case study to verify the BPHM in our cloud IaaS system.

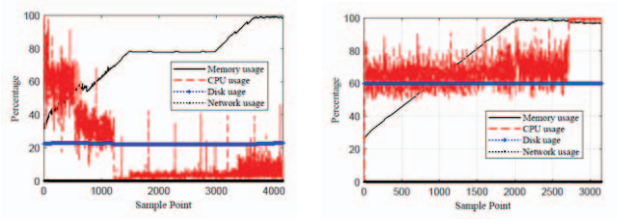
The following subsection 4.2 first describes the health index monitored by cyber-axons. Subsection 4.3 shows the experimental results of failure prognostics by cyber-neurons. Subsection 4.4 shows the experimental results of self-healing by peripheral nerves. Subsection 4.5 shows the experimental results of self-learning by central nerves. Subsection 4.6 shows the overall performance of BPHM with regular maintenance.

B. Health Index Monitored by Cyber-Axons

In this case study, we define a health index as a vector consists of multiple metrics related to resource usages of the VM, that is, $X = \{x_1, x_2, \ldots, x_N\}$. Elements in the health index includes both historical and real-time data gleaned by the cyber axons. The design of the health index we implemented here is described as follows:

Memory usage related metrics: since the memory usage is the most obvious phenomenon effected by the memory anomaly, the health index includes more metrics related the memory usage:

1) *Instant memory usage*: real-time memory usage metric of the VM, denoted by x_1;



(a) No correlation memory anomaly (b) CPU-correlated memory anomaly

Fig. 6. Two examples of historical data for different memory anomalies.

2) *Derivative of the memory usage*: it is a metric that needs to be computed. It is obtained as $(x_1(t_s) - x_1(t_{s-1}))/\triangle t$, where $\triangle t = t_s - t_{s-1}$ is the time span between two continuous sample points. This metric is denoted by x_2.

3) *Mean memory usage*: it is the mean value of the memory usage over S sample points (S is usually set as a relative small value). It can be calculated as $\sum_{s=1}^{S} x_1(t_s)/S$, where $x_1(t_s)$ is the instant memory usage at sample point t_s $(s = 1, \ldots, S)$. This metric, x_3, can help the cyber axon have an information of historical conditions of the memory usage.

4) *Increase possibility of the memory usage*: given S sample points $t_S, t_{S-1}, \ldots, t_2, t_1$, find the number of the samples that satisfies $x_(t_s) - x_1(t_{s-1}) > 0$, suppose it is S'. Then, this metric denoted by x_4 is written as S'/S.

CPU usage related metrics: in general, the CPU usage has a strong correlation with the memory usage because the CPU often performs too busy if memory is not enough. Therefore, we also give more metrics to describe a comprehensive CPU usage situation. Similar with the metrics designed for describing the memory condition, multiple metrics are collected and computed to show the condition of the CPU, including instant CPU usage (x_5), derivative of the CPU usage (x_6), mean CPU usage (x_7), and increase possibility of the CPU usage (x_8).

Disk and network related metrics: the I/O throughput (x_9) is also correlated with memory consumption for virtual memory communicating with disks via I/O. The network throughput of the VM (x_{10}) can be treated as valuable metrics for memory anomaly when users apply VM to provide web services.

Health index: after normalizing these metrics, the health index can be derived as $X = \{x_1, \ldots, x_N\}$ $(N = 10)$, which is transmitted to the cyber neuron for intelligent failure prognostics. Note that the health index can be customized by a specific cloud system for some special demands.

According to the designed health index for our cloud system, the cyber axon gleans real data for all VMs rent to cloud users. Hereby, we illustrate two data examples of the health index that is being monitored by cyber axons. Fig. 6 shows data sets of health index when two kinds of memory anomalies happened. In Fig. 6 (a), we can find that the memory usage keeps continuous increase, and the CPU usage decreases dramatically. This may be a memory leak. That is, all memory resource has been occupied by an abnormal program, and thus all normal programs can not request memory resource any more, which results in that the CPU resource becomes idle.

TABLE I
EXPERIMENT RESULTS FOR THE FAILURE PROGNOSTICS MODULE.

Description	Number of Data	Number of Misreported Data	Value
Accuracy rate	$M_{nor} + M_{ab} = 25656$	$M_{err1} + M_{err2} = 159$	99.3803%
Type 1 error	$M_{nor} = 21186$	$M_{err1} = 114$	0.5381%
Type 2 error	$M_{ab} = 4470$	$M_{err2} = 45$	1.0067%

However, in Fig. 6 (b), there is a totally different memory anomaly. We can find that the CPU usage remains at a high level with the increase of the memory usage, which means that the occurred memory anomaly may be a CPU-correlated memory anomaly, such as a virus occupies a large amount of the CPU resource.

C. Failure Prognostics by Cyber-Neurons

We first use almost 40000 historical records of the health index (including normal indices and abnormal indices) to train the cyber neuron and the peripheral nerve. After training the cyber neuron and the peripheral nerve, the BPHM system has capabilities of the failure prognostics and the self healing. Then, a verification dataset with 21186 (denoted by M_{nor}) normal health indices and 4470 (denoted by M_{ab}) abnormal health indices are inputted into the BPHM to verify the trained cyber neuron and peripheral nerve. Given a health index X, the trained cyber neuron use the failure prognostics model $y = h_w(X)$ to output a result as $y = 1$ or $y = 0$. The diagnosis accuracy is defined as

$$\text{diag}_{acc} = \frac{M_{corr}}{M_{nor} + M_{ab}} \tag{8}$$

where M_{corr} is the number of health indices that are diagnosed correctly. From (8), the experimental results shows that the trained failure prognostics module in the BPHM achieves an accuracy of $\text{diag}_{acc} = 99.3903\%$ where there are only 159 data do not have a correct diagnosis result. Two types of errors are used to analyze 159 indices with wrong diagnosis results, that is,

1) *Type 1 error*: if a normal health index is diagnosed as that it will lead to a failure, the diagnosis result is treated as a type 1 error. Suppose there are M_{err1} diagnosis results having the type 1. The type 1 error rate of the failure prognostics module is given by

$$\text{err}_1 = \frac{M_{err1}}{M_{nor}} \tag{9}$$

In the verification dataset, there are 114 normal indices have the type 1 error, and the type 1 error rate is $\text{err}_1 = 144/21186 = 0.5381\%$.

2) *Type 2 error*: if an abnormal index of a memory anomaly that should be detected is actually missed, it results in a type 2 error. Let M_{err2} represent the total number of indices with the type 2 error. The type 2 error rate is written as

$$\text{err}_2 = \frac{M_{err2}}{M_{nor}} \tag{10}$$

There are 45 abnormal indices have been missed by the self-diagnosis module, and the corresponding type 2 error rate is computed as $\text{err}_2 = 45/4470 = 1.0067\%$.

If a failure detection result has the type 2 error, it is missed by the BPHM. The corresponding memory anomaly will lead to the service reliability decrease of the cloud system. However, the problem of the type 2 error can be effectively solved by adopting the BPHM and regular maintenance, which will be introduced in the following subsections.

D. Self-Healing by Peripheral Nerves

To reflect a healing plan for an exceptional health index, a conditional reflection rule should be preset in advance. In practice, abnormal healing indices are observed as different symptoms caused by different memory anomalies. Therefore, it is rational to define the reflection rule according to categories of the memory anomalies.

In fact, with the run of our cloud system, we found that if more metrics in the health index are affected by a memory anomaly, the self-healing reflection usually needs to deal with the consequence of all infected components. Therefore, in our implementation, we give the categories of the memory anomaly according to the correlation of abnormal changes among different metrics in the health index. There are five categories of various memory anomalies are listed as follows.

1) *No-correlation memory anomaly*: such a memory anomaly only incurs abnormal changes in the memory usage metric. The cause of the no-correlation memory anomaly could be a memory leak.

2) *CPU-correlated memory anomaly*: a CPU-correlated memory anomaly mainly results in abnormal changes in both the memory usage metric and the CPU usage metric. For example, in the Window 10 system, some system processes such as ntoskrnl.exe and Runtime Broker running without a correct runtime environment can lead to a memory overflow with a high CPU consumption.

3) *Disk-correlated memory anomaly*: it is similar to the CPU-correlated memory anomaly. But it affects memory usage metrics and disk usage metrics. For example, if the physical memory is used up, the virtual memory will cause lots of I/O with disks.

4) *Network-correlated memory anomaly*: it can be found as exceptional phenomenon in both network and memory resources. For example, memory anomalies caused by a web service where the program forgets releasing connections after the web services done.

5) *Serious memory anomaly*: it obviously affects more than two resources above, and the cloud system treats it as a serious

TABLE II
THE PRESET REFLECTION RULE FOR MEMORY ANOMALIES.

No.	Categories of the memory anomaly	Preset healing plan
1	No correlation memory anomaly	Limit memory usage
2	CPU-correlated memory anomaly	VM rollback
3	Disk-correlated memory anomaly	Suspend suspicious process
4	Network-correlated memory anomaly	VM migration
5	Serious memory anomaly	VM shutdown and reboot
6	Type 1 error from prior self-diagnosis	No action

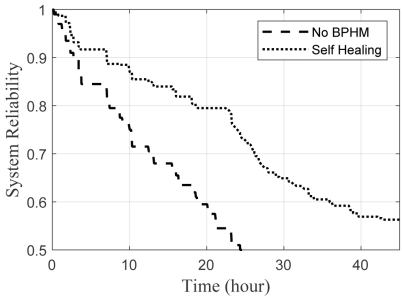


Fig. 7. Service reliability analysis of the cloud system by using the self healing.

memory anomaly that may result in sever damage to the system health.

According to the categories of the memory anomaly, the reflection rule can be further defined. In this experiment, the set of healing plans as following Table 2 shows the preset reflection actions.

In Table 2, a healing plan of 'no action' is necessary because that some normal cases misreported as failures by the cyber neuron when type 1 errors occur. Now, 4470 abnormal health indices in the verification dataset are transmitted into the peripheral nerve for verifying the capability of the self healing. The reflection accuracy rate of the peripheral nerve is defined as the rate of choosing a correct one according to the preset healing rule. The reflection accuracy rate of the peripheral nerve on the 4470 abnormal health indices is 90.2401%.

Fig. 7 shows the effect of the self-healing module by running the cloud system in a real case of two days, which is compared with the situation that no BPHM in the cloud system. As shown in the figure, if the cloud system does not have the BPHM, the service reliability of the cloud system decreases dramatically. It also can be found that the self healing has an obvious effect on slowing down the decrease trend of the service reliability of the cloud system to manage a better system health.

E. Self-Learning by Central Nerves

According to the preset reflection rule, the peripheral nerve may not reflect a most suitable healing plan. Meanwhile, type 1 errors also need to be handled in the self-healing module. Therefore, the central nerve performing the self learning in the BPHM is indispensable. After using the self-learning module presented in Section 3.4, the BPHM can coordinate and update

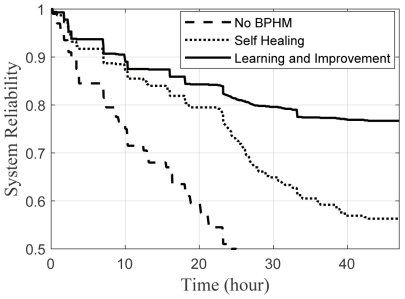


Fig. 8. Service reliability analysis of the cloud system by using the self learning.

the reflection rule (including add the healing plan of no action for the type 1 error) to improve the accuracy of the BPHM.

Fig. 8 demonstrates that the service reliability of our cloud system running for two days by using the self-learning module. As shown in the figure, once the BPHM has the key ability of the self-learning, the BPHM has more obvious effects on slowing down the degradation of service reliability. That is, with the run of our cloud system, the BPHM keeps learning knowledge or experience from healing actions. Therefore, the gap between the line of the self-learning and the dot line of the self healing (*i.e.*, without the self-learning) becomes larger and larger. This implies the learning procedure of the self-learning modules. With the knowledge accumulation by using the self-learning, the reflection rule is increasingly improved. Therefore, the BPHM becomes more accurate on dealing with complex memory anomalies.

However, there still exist some type 2 errors that cannot be removed by using the BPHM, which was accumulated in the system although few. Therefore, regular maintenance of the cloud system is also necessary. In fact, the regular maintenance can be associated with the BPHM for achieving a better effect on guaranteeing the health of the cloud system.

F. Realistic Performance of BPHM with Regular Maintenance

Even though the type 2 error cannot be totally eliminated by the BPHM automatically, it also can be effectively removed by combining the BPHM with the regular maintenance. This is rational since not only the cloud system but also all IT systems definitely have regular maintenance strategies. In fact, the designed BPHM also brings some advantages on optimizing maintenance strategies with less frequencies.

To show the realistic performance of the BPHM, we run the cloud IaaS system eight days. After running the cloud system for every two days, we do a manual maintenance to recovery the cloud system to an original state (restart all the VMs and services in the midnight). Fig. 9 shows the change of the service reliability of the cloud system with the regular maintenances.

As shown in Fig. 9, there are four running cycles (*i.e.*, each two days for running the cloud system without the manual maintenance) of the cloud system. At the ends of four run cycles, the BPHM makes the system reliability

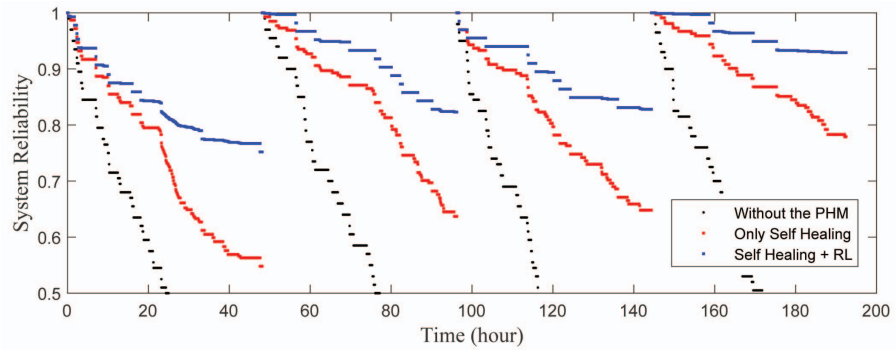


Fig. 9. Service reliability analysis of the cloud system with regular maintenance.

remains at 0.7523, 0.8230, 0.8288, and 0.9293, respectively. This demonstrates that the realistic performance of the BPHM has a significant effect on guaranteeing the health of the cloud system. However, as for several running cycles at the beginning of the BPHM, the realistic performance may not be too obvious or even a slight decrease. For example, as for the second and third run cycles shown in Fig. 9, the lowest service reliability are 0.8230 and 0.8288, respectively. This mainly because that

1) *The BPHM needs time to learn knowledge*: in realistic environment, various memory anomalies appears in a complex random manner. At the beginning of the BPHM, there may be partial categories of the memory anomalies existed in the cloud system. Therefore, knowledge learned from these memory anomalies may not be comprehensive for covering all categories of memory anomalies. For some memory anomalies that occurred rarely before, if they suddenly appear in the cloud system in a high frequency, the BPHM needs a time period to learn the knowledge about those new memory anomalies. In this phase, the system reliability may not be improved obviously.

2) *Random occurrence of type 2 errors*: Since type 2 error cannot be removed by the BPHM, the frequency of type 2 error occurrence also make the service reliability of the cloud system has a slight fluctuation. That is, if there are more type 2 errors occur in a run cycle, the service reliability has more decrease caused by those errors. However, due to the error rate of the type 2 error is a relative small value, it does not bring serious negative effect on the service reliability.

Meanwhile, the designed BPHM also brings some benefits for optimizing maintenance strategies of the cloud system. That is, after the BPHM learns enough knowledge to ensure the cloud system running with a relatively high reliability, the frequency of carrying out manual maintenance can be reduced accordingly. This potentially implies the BPHM has a positive effect on saving the maintenance cost.

V. Conclusion and Future Work

In the cloud IaaS system, users rent VMs to run their personal programs. Therefore, how to guarantee the service reliability of the cloud system under the cloud users unknown actions is indeed a complicated problem for the cloud provider. To solve such a problem, this work presented a new application of the PHM technology, named as BPHM. The primary innovation of the BPHM is that it takes advantage of the biological system BANS, and thus it can imitate human with powerful abilities of perception, detection, reflection, and learning and improvement.

The contribution of the BPHM can be summarized as: 1) we realized the cyber axons for perceiving the real-time health index of VMs. This is an efficient approach to find a memory anomaly as soon as possible for the cloud IaaS system, since cloud users personal programs and applications in VMs are inaccessible for the cloud provider; 2) we also designed the cyber neuron for prognostics of failures in the cloud system. By using the neural network to train the cyber neurons, those VMs are capable of the failure prognostics. Different form traditional failure detection techniques, the presented failure prognostics is more intelligent. It can detect VM failures from the health index instead of finding errors, bugs, or faults in users' programs or applications; 3) we implemented the self healing for the peripheral nerve with timely reflection function. In the cloud IaaS system, a physical server may host multiple co-located VMs rented by different cloud users. Therefore, this immediate conditional reflection is very important for the cloud IaaS system. It can effectively prevent a VM failure bringing more harm to the cloud system, such as a server failure; 4) a cloud service reliability model for the IaaS was presented in this work. It can be used to evaluate service reliability of the cloud IaaS system. Then, the healing effect of a healing plan can be quantified as a reliability-related metric. It provides a novel theoretical base to implement the self-learning; 5) the ability of learning and improvement is a distinctive feature of the BPHM. After learning knowledge from healing actions, the central nerve updates the reflection rule in the peripheral nerve to improve the accuracy of the self healing increasingly. That is, with the run of the BPHM, it can become more intelligent and more accurate.

We also implemented the BPHM in a real cloud IaaS system, and carried out a case study to verify the effectiveness of the BPHM technology. The experimental results exhibit that the BPHM is an intelligent, autonomous, and adaptive system

that can effectively deal with complex memory anomalies. In principle, the PHM needs have two key functions of early warning on failures (failure prognostics) and preventative maintenance to keep system health (health management). This case study verified that the cyber axon and the cyber neuron in the BPHM satisfied the demand of the failure prognostics with a high accuracy of detecting failures from real-time health indices. Meanwhile, our experimental results also demonstrated that the self healing module in the BPHM had a significant effect on alleviating the decrease of the cloud service reliability. More importantly, the self-learning module continued improving the accuracy of the self-healing actions, and as a result the cloud system could steadily remain at a higher service reliability. This verified that the peripheral nerve and the central nerve in the BPHM have excellent abilities of managing the system health. Hereby, the experimental results illustrate that the proposed BPHM is effective and efficient to achieve both functions of PHM intelligently.

In our future work, we will extend the BPHM to other types of cloud computing systems, such as SaaS (Software as a Service) and PaaS (Platform as a Service) offered by cloud. This extensions is feasible by using the BPHM presented in this work, but some modules in the BPHM need to be customized. For example, health indices monitored by cyber axons will focus on different perspectives; the self-diagnosis by cyber neuron need consider more layers across infrastructure, platform and software layers; self-healing module in peripheral nerve can be designed with more access to APIs of a platform; and the cloud service reliability model evaluated by central nerve is different for SaaS and PaaS from the IaaS.

REFERENCES

[1] Bala A, Inderveer C. Fault tolerance-challenges, techniques and implementation in cloud computing. International Journal of Computer Science Issues (IJCSI), 2012, 9(1): 288-293.

[2] Gao J, Bai X, Tsai W T. Cloud testing-issues, challenges, needs and practice. Software Engineering: An International Journal, 2011, 1(1): 9-23.

[3] Chiang L H, Russell E L, Braatz R D. Fault detection and diagnosis in industrial systems. Springer Science & Business Media, 2000.

[4] Buyya R, Calheiros R N, Li X. Autonomic cloud computing: Open challenges and architectural elements. 2012 third international conference on emerging applications of information technology. IEEE, 2012: 3-10.

[5] Kim N H, An D, Choi J H. Prognostics and health management of engineering systems: An introduction. springer, 2016.

[6] Cheng S, Tom K, Thomas L, et al. A wireless sensor system for prognostics and health management. IEEE Sensors Journal, 2010, 10(4): 856-862.

[7] Hansen C K. A prognostic model for managing consumer electricity demand and smart grid reliability. 2012 IEEE Conference on Prognostics and Health Management. IEEE, 2012: 1-6.

[8] Vichare N M, Pecht M G. Prognostics and health management of electronics. IEEE transactions on components and packaging technologies, 2006, 29(1): 222-229.

[9] Kirillov A, Kirillov S, Pecht M. The Problem of PHM Cloud Cluster in the Context of Development of Self-maintenance and Self-recovery Engineering Systems. Engineering Asset Management-Systems, Professional Practices and Certification. Springer, Cham, 2015: 1509-1520.

[10] Cui Y, Zhu L, Cai Z, et al. An adaptive traffic-aware migration algorithm selection framework in live migration of multiple virtual machines. International Journal of Performability Engineering, 2020, 16(2): 314-324.

[11] Amazon. Amazon elastic container service, highly secure, reliable and scalable way to run containers. https://aws.amazon.com/esc/?nc2=h_ql_prod_ct_ecs

[12] Newcombe C, Rath T, Zhang F, et al. How Amazon web services uses formal methods. Communication of the ACM. 2015, 58(4): 66-73.

[13] Deelman E, Singh G, Livny M, et al. The cost of doing science on the cloud: the Montage example. IEEE International Conference on High Performance Computing, Data, and Analytics. 2008.

[14] Sor V, Srirama S N. A statistical approach for identifying memory leaks in cloud applications. First International Conference on Cloud Computing and Services Science (CLOSER 2011). SciTePress, 2011: 623-628.

[15] Teixeira B, Louren?o J, Farchi E, et al. Detection of transactional memory anomalies using static analysis. Proceedings of the 8th workshop on parallel and distributed systems: Testing, analysis, and debugging. 2010: 26-36.

[16] Guan Q, Fu S. Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures. 2013 IEEE 32nd International Symposium on Reliable Distributed Systems. IEEE, 2013: 205-214.

[17] Dahlstedt J, Lonnebring P, Vidstedt M. System and method for memory leak detection in a virtual machine environment: U.S. Patent 7,757,202. 2010-7-13.

[18] Wang T, Xu J, Zhang W, et al. Self-adaptive cloud monitoring with online anomaly detection. Future Generation Computer Systems, 2018, 80: 89-101.

[19] Gionta J, Azab A, Enck W, et al. Seer: practical memory virus scanning as a service. Proceedings of the 30th Annual Computer Security Applications Conference. 2014: 186-195.

[20] Venkataramani G, Roemer B, Solihin Y, et al. Memtracker: Efficient and programmable support for memory access monitoring and debugging. 2007 IEEE 13th International Symposium on High Performance Computer Architecture. IEEE, 2007: 273-284.

[21] Novark G, Berger E D, Zorn B G. Exterminator: Automatically correcting memory errors with high probability. Proceedings of the 28th ACM SIGPLAN Conference on Programming Language Design and Implementation. 2007: 1-11.

[22] Chiang L H, Russell E L, Braatz R D. Fault detection and diagnosis in industrial systems. Springer Science & Business Media, 2000.

[23] Kephart J O, Chess D M. The vision of autonomic computing. Computer, 2003, 36(1): 41-50.

[24] Hamid N A, Nawi N M, Ghazali R, et al. Improvements of back propagation algorithm performance by adaptively changing gain, momentum and learning rate. International Journal of New Computer Architectures and their Applications, 2011, 1(4): 866-878.

[25] Travostino F. Seamless live migration of virtual machines over the MAN/WAN. Elsevier B.V. 2006.

[26] Xie M, Dai Y, and Poh K L, Computing Systems Reliability: Models and Analysis. New York, NY, USA: Kluwer, 2004.

[27] Trivedi K S, Probability and Statistics With Reliability, Queuing, and Computer Science Applications. New York, NY, USA: Wiley, 2001.

[28] Zuo M J and Tian Z, Performance Evaluation of Generalized Multi-State k-out-of-n Systems. IEEE Transactions on Reliability, 2006, 55(2): 319-327.

[29] Qiu X, Dai Y, Xiang Y, et al. Correlation modeling and resource optimization for cloud service with fault recovery. IEEE Transactions on Cloud Computing, 2019, 7(3): 693-704.

[30] Qiu X, Dai Y, Xiang Y, et al. A hierarchical correlation model for evaluating reliability, performance, and power consumption of a cloud service. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2015, 46(3): 401-412.

[31] Konda V R , Tsitsiklis J N . On actor-critic algorithms. SIAM Journal on Control and Optimization, 2000, 42(4).

[32] Bahdanau D, Brakel P, Xu K, et al. An actor-critic algorithm for sequence prediction. International Conference on Learning Representations, ICLR, 2017.