The Effect of Real-valued Negative Selection Algorithm on Web Server Aging Detection

Yiwen Liang, Huan Yang, Jun Fu, Chengyu Tan✉, Aolin Liu, Shiwen Zhu
School of Computer Science, Wuhan University, Wuhan, China
Email: ywliang@whu.edu.cn, happyfairy106@163.com,doctorfj@163.com, nadinetan@163.com, aolin.liu@gmail.com, z-swen@163.com

Abstract— **Several researchers have reported the fact that web server systems executing continuously for a long time show a degradation of their performance, and/or an increased occurrence rate of hang/crash failures. This phenomenon has been called 'web server aging'. To avoid this problem, it becomes an important issue to detect web server aging in web server maintenance. Existing technologies depending on aging samples succeed in detecting known web server aging, but fail to detect novel aging because of the nondeterministic nature of the web server aging. Given that normal samples are much easier to acquire from running web server than aging samples, this paper proposed an immune-inspired real-valued negative selection algorithm to detect previously unseen web server aging, it only needs normal samples to train detectors that have the ability to classify a novel sample as web server aging or not. The basis of the detection is aging causes performance deviation from the normal state. Preliminary experimental results showed that the method could improve the performance of the detection of novel web server aging without responding to normal status.**

Index Terms— **Real-valued Negative Selection Algorithm (RNSA); Artificial Immune System (AIS); Web Server Aging Detection; Software Aging**

I. Introduction

Long-running web server systems are prone to suffer from software aging, which results in an increasing failure rate and/or degraded performance, such as increased response time or even sudden crashes [1][2]. This phenomenon usually results in huge losses, particularly to e-commerce and safety/business-critical applications hosted in the web server [1]. Unlike other types of faults, web server aging is more nondeterministic and is difficult to reproduce. This makes it hard to be predicted by most testing and forecasting techniques [3][4]. Therefore, detecting web server aging as early as possible is the basis to prevent unexpected failure of application services.

Grottke et al. [1] and Li et al. [2] use statistic-based detection techniques that compare current resource usage to historical aging modes and trend model of a web server to classify it ages or not. This method has high detection efficiency of known aging cases. However, the dynamic property of web server aging (e.g. non-deterministic) brings this method a big challenge for detection of novel aging [4][5].

To overcome the problem above, machine-learning-based detection method is proposed to classify current performance instance as aging or not, without the information of known aging modes. This is realized through measuring the performance deviation of a web server by classifiers trained by normal & aging data [4][5]. Experimental results show that the method can detect some unknown web server aging.

However, both methods above have some shortcomings. They need a large amount of aging samples that can only be obtained when aging occurs. But the transient and occasional nature of the web server aging inevitably makes the collection of aging samples a time-consuming and costly task [4]. As a result, detecting aging with only normal samples that can be easily acquired may be a better approach to web server aging detection.

With this idea, Yang [3] uses an artificial immune system approach, namely real-valued negative selection algorithm (RNSA) to detect unknown web server aging adaptively. The main idea of the RNSA is self/non-self discrimination, considering self as the normal (healthy) behavior of the system, and non-self as the abnormal (unhealthy) [6]. RNSA is able to detect previously unseen anomaly or fault with only self samples, this dramatic advantage makes it suitable for dealing with the difficulties to collect aging samples in web server aging detection. But the feasibility of this method hasn't been validated.

Based on the work above, we analyzed the resource usage and performance development of an Apache web server system (which is one of the most popular used web server systems), and discovered that the Apache shows evidence of aging after inserting a memory leak into the web site running in it. Therefore, in this paper, we carried out experiments to evaluate the performance of RNSA on web server aging detection, with an Apache web server instance. In addition, we suggested some improvements of the RNSA in [3] to enhance its performance for the web server aging detection, and described the principles to set the parameters that have an impact on the results of our approach.

II. Related Work

Web server aging refers to the phenomenon that web servers will show performance degradation, increasing failure rate, even a paroxysmal crash after longtime continuous execution [1].

doi:10.4304/jsw.7.4.849-855

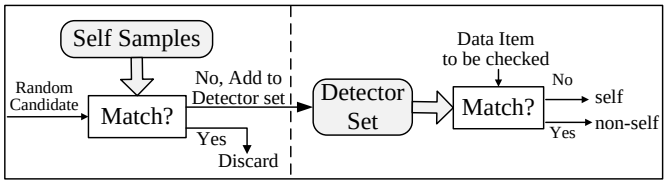


Figure 1. Mechanism of RNSA

Not only the causes are elusive [7], web server aging also has the following non-deterministic characteristics [4]:

1) performance of the web server depends on the application service and the time-varying system workload;
2) the aging profile changes along the time and the rate at which the web server ages is usually not stable or regular [5];
3) its causes are various and unpredictable, transient fault is particularly difficult to predict and reproduce [1][7].

Since statistic-based detection has nothing to do against novel aging [3][4], software experts are now focusing on intelligent techniques that measure and analyze the performance deviation caused by behaviors exhibited by a web server to classify it as aging or not.

As aging always causes performance degradation of the web server and resource exhaustion of OS, Andrzejak et al. [4][5] mine classification rules of the server's performance characteristics from normal & aging training data sets. Then, the rules are applied to classifying new performance state of the server as aging or not. Although this conventional classification method is more robust against some unknown aging, the detection rate is not high, because samples for all classes (e.g. normal and aging) are needed during the training phase [8]. However, in web server aging application, aging samples are not available for training purpose, as it is difficult to obtain information about all possible aging [3][5][8].

Given that normal samples are much easier to acquire from running web server than aging samples, Yang [3] uses the immune-inspired real-valued negative selection algorithm (RNSA) to detect unknown web server aging adaptively requiring only normal samples. The RNSA simulates the process of selecting non-autoreactive lymphocytes inside the human thymus. In this process (shown in fig.1), lymphocytes that recognize body's own elements (self antigens) are eliminated, the remaining lymphocytes (detectors) will identify only foreign molecules (non-self antigens). The RNSA uses a real-valued representation of the self/non-self space to map the generated detectors back to the problem space easily [9][8]. More information about RNSA please refer to [6][10].

The advantage of RNSA is that it detects previously unseen anomalies with detectors trained from only self samples. This makes it suitable for dealing with the difficulties to collect aging (non-self) samples of a web server. But the authors of [3] hasn't validated their method in real problems.

III. RNSA-based Web Server Aging Detection

Web servers are supposed to operate normally in 25 days [1] since the latest restart, we consider the performance information of web servers in this period to be the self antigens. The system resource usage, average load, and performance of a web server are continuously monitored, and detectors trained by RNSA are applied to recognizing performance anomaly (non-self) of the web server in the operating condition.

Based on the work of [3], we will discuss some key issues of the RNSA-based web server aging detection and some improvement in this session, e.g. the problem space definition and the data normalization, the matching rule, details of the detector population generation, and the added detector evaluation.

A. the Problem Space Definition

The key idea of realizing web server aging detection with RNSA is to classify the performance state of a web server as healthy (self) or abnormal (non-self). For this purpose, the antigen is considered to be the web server performance state, which can be formalized as $Antigen(A_g)$ = {*freeMemRatio, usedSwapSpace, loadAvg, serviceRate, responseTimeAvg*}. These indicators have the ability to represent a web server's performance deviation [4][11], they and their corresponding formalized functions are described as follows.

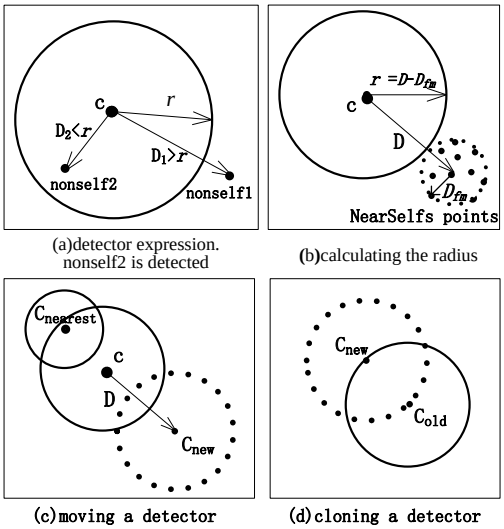
- **freeMemRatio**: it refers to the free physical memory ratio of the system, it is normalized as follows:

$$f(x) = \begin{cases} 100, & x \geq N_h \quad (1a) \\ 100 * \dfrac{x - N_l}{N_h - N_l}, & N_l < x < N_h \quad (1b) \\ 0, & x \leq 0 \quad (1c) \end{cases}$$

  where x represents the free physical memory ratio, N_h and N_l are its maximum and minimum value respectively.
- **usedSwapSpace**: it refers to the swap space of OS that have been used. It is normalized as follows:

$$f(x) = \begin{cases} 100, & x \geq N_{sh} \quad (2a) \\ 100 * \dfrac{x}{N_{sh}}, & 0 \leq x < N_{sh} \quad (2b) \end{cases}$$

  where x is the swap space used, and N_{sh} is its maximum value.
- **loadAvg**: Average cpu load of the system, shows the average number of processes in the cpu queue in a specific period of time. It's normalized similarly to the usedSwapRatio, but with N_{lh} as its maximum value.
- **replyRatio**: It shows the ratio of number of actually served requests to number of incoming requests per second, its value varies from 0 to 100.
- **responseTimeAvg**: It shows the average response time of all requests in a specific period of time. It's



(a)detector expression. nonself2 is detected

(b)calculating the radius

(c)moving a detector

(d)cloning a detector

Figure 2. Detector Generation Details

normalized similarly to the usedSwapRatio, but with N_{rth} as its maximum value.

From the above, we can see that A_g is expressed as a 5-dimensional real-valued vector. Therefore, the problem space of the web server aging detection can be seen as a 5-dimensional space with its boundary in the domains of all indicators above. In this space, an A_g is considered as a point, a detector (antibody) is considered to be a hypersphere with a 5-dimensional vector corresponding to its center and a real parameter representing its radius. Then, an antigen is recognized by a detector, once the distance between them isn't more than the radius of the detector.

B. Detectors Expression

Detectors are the key components of the real-valued negative selection algorithm, it is considered as a hypersphere able to cover antigens in a certain domain. In this paper, we define the detector as *d=(c, r, distance)*:

- $c = (c_1, c_2, \cdots, c_5)$ is the center of the detector. It corresponds to a 5-dimensional point inside the problem space of the web server aging detection. It has the same presentation as the A_g in this paper.
- $r \epsilon \mathbb{R}$ $(r > 0)$: It represents the radius of a detector hypersphere which determines the size of the detector. As shown in fig.2(a), an antigen (A_g) satisfied with $distance(d, A_g) \leqslant r$ is recognized as non-self (performance anomaly) by the detector d.
- *distance*: It refers to the matching rule, which evaluates the affinity between the detector and another data item to measure whether they match or not. As it's described as a uniform membership function of the detector in this paper, we describe the detector as *d=(c, r)* briefly.

C. the Matching Rule

The matching rule describes the function that estimates whether a detector matches an antigen (or another detector). Generally speaking, RNSA requires a partial matching rule to ensure the diversity of detectors with the purpose of reducing holes [3].

Because the *Euclidean* distance employed in [3] causes the evaluated distance out of boundary of the problem space, we replace it by *Minkowski* distance which is also suitable for real-valued representation but follows the rule of the boundary, so as to make the computation intuitive and exact.

Minkowski distance is computed as shown in equation (3), and it equals to the *Euclidean* distance when n=2. Thus, the distance/affinity *(D)* between a point x and a detector, *d=(c, r)*, is computed as in (3), where c is the center of the detector. Matching is determined when $Minkowski(c, x) \leqslant r$ is true .

$$D(d, x) = Minkowski(c, x) = (\sum_{i=1}^{n} |c_i - x_i|^n)^{1/n} \quad (3)$$

D. the Detector Generation Algorithm

In order to maximize the coverage of non-self space, the real-valued negative selection algorithm aims to train a set of matured detectors that match none of self samples and are away from each other. Fig.3 shows the basic steps of the detector generation algorithm.

We propose some improvements of the detector generation algorithm used by [3] to make the algorithm more appropriate for actual web server aging detection, such as improving the training process, adjusting the moment to clone better-fitted detectors, normalizing the center of every detector, and restraining the boundary of the new center generated by detector moving or clone.

1) ***Radius calculation for the detectors****:* With a set of self antigens as input, the algorithm starts with a set of candidate detectors, which are generated at random and then matured through an iterative process. Especially, the center of each candidate detector is chosen randomly but different from each other, and the radius is a variable parameter waiting to be assigned.

At each iteration, the radius of every candidate detector, *d=(c,r)*, should be calculated or updated adaptively based on the *Minkowski* distance between this candidate and the self points. The radius is calculated as follows (shown in fig.2(b)):

$$r = D(d, MidSelf) - D(FarSelf, MidSelf) \quad (4)$$

$$MidSelf = (\sum s)/k, \quad s \epsilon NearSelfs \quad (5)$$

Where *MidSelf* is the center point of *NearSelfs* which refers to the k nearest self points to d, *FarSelf* is the farthest self point from the *MidSelf* in the *NearSelfs*, then, *D(d,MidSelf)* shows the *Minkowski* distance between d and *MidSelf*, *D(FarSelf,MidSelf)* (D_fm) shows the *Minkowski* distance between *MidSelf* and *FarSelf*.

A candidate detector is considered to match self and will be discarded in subsequent iteration, once its radius is negative; Otherwise, this candidate is matured if the

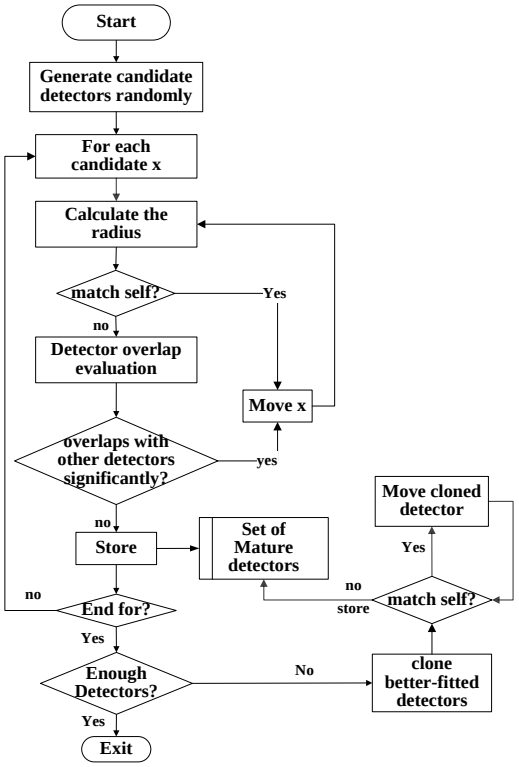


Figure 3. RNSA Detector Training

overlap between it and other existing detectors is not more than threshold ξ (ξ: provides the maximum allowable overlap among the detectors, adjusted by experiment).

2) ***Regeneration of discarded detectors****:* As mentioned in [3], a discarded detector *d=(c,r)* matching self or overlapping with existing detectors is moved to become a new candidate detector to improve the efficiency of detector generation. As shown in fig.2(c), this exactly mean the center of *d* is moved.

The new center should be moved away from the nearest neighbor detector and/or self point of the discarded detector to maximize the coverage of the detectors. So, the new center is calculated as follows:

$$c_{new} = c + \eta * dir \quad (6)$$

$$dir = (c - c_{nrest})/D(c, c_{nrest}) \quad (7)$$

Where d_{nrest}=(c_{nrest}, r_{nrest}) is the nearest detector (or self point) to d, η is a variable that provides the level to move a detector away from self points or other existing detectors. According to the experimental investigation, c_{new} should be normalized as shown in section 3.A.

3) ***Better-fitted detectors clone****:* As shown in fig. 3, if there aren't enough matured detectors after the first round of training, the population will be filled with the clones of the better-fitted detectors (shown in fig. 2(d)). A detector is considered as better-fitted detector, if it has larger coverage but small overlap with other detectors (evaluating method is shown in the next subsection).

Given that *d=(c, r)* is the detector to be cloned, and d_{clone}=(c_{clone}, r_{clone}) is the cloned detector which is far away from d's nearest detector and located at a distance r from d, r_{clone} and c_{clone} are defined as:

- $r_{clone} = r$.
- $c_{clone} = c + r * (c - c_{nrst})/D(c, c_{nrst})$, where c_{nrst} is the center of the detector which is nearest to d.

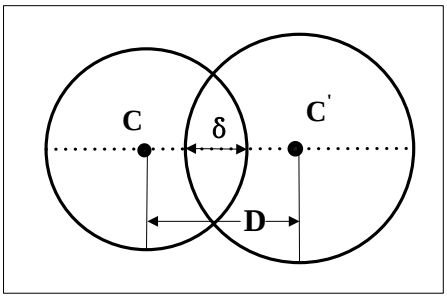


Figure 4. the overlap between two detectors

E. the Detector Evaluation

In this paper, to generate more effective detectors, we favor the detectors having larger coverage of the non-self space (i.e. with big radii) but with minimum overlap among them.

Detectors which do not match any self are sorted according to their radii. The detector with large radius gets selected for the next generation population if it has small overlap with existing detectors, otherwise, it will be moved. So, the overlapping measure W of a detector is computed as the sum of its overlap with all other detectors as follows:

$$W(d) = \sum_{d \neq d'} w(d, d'), \quad w(d, d') = (exp(\lambda) - 1)^m \quad (8)$$

$$\lambda = \begin{cases} 1, & \delta \geq 1 \quad (9a) \\ \delta, & 0 < \delta < 1 \quad (9b) \\ 0, & \delta \leq 0 \quad (9c) \end{cases}$$

$$\delta = (r + r' - D)/2r$$

Where w(d,d') is the overlap between two detectors *d=(c,r)* and *d'=(c',r')*, m is the dimension of the data space, D is the distance between two detector centers c and c'. As shown in fig. 4, the two detectors are disjoint when $\delta \leq 0$, so $\lambda = 0$, $w(d, d') = 0$; They are joint when $0 < \delta < 1$, and d is covered by d' and should be moved when $\delta \geq 1$.

F. the Life-cycle of Detectors

The detector population must have the ability to coverage dynamically and update periodically in order to adapt to the change of the observed web server. In this respect, supposing P_{death} to be the mortality rate of detectors, each detector should get a life cycle T_{life}=$1/P_{death}$.

In this paper, we use the life cycle management method suggested by Hofmeyr [12]. A matured detector is activated, if it accumulates enough matches to exceed a

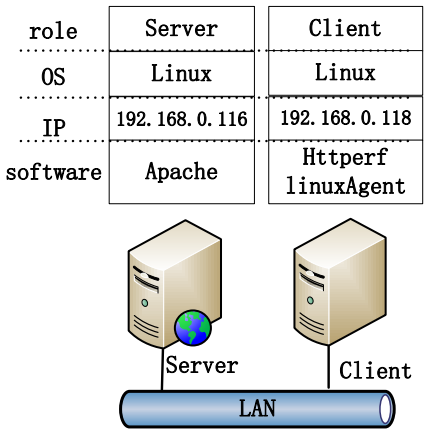


Figure 5. Experimental deployment and components

threshold τ in the period of T_{life}. Otherwise, it dies and is replaced with a new candidate detector shown in fig.2(c) to update the detector set. The threshold τ can be adjusted by experiment, here are some suggestions [1,5].

If an activated matured detector receives the confirmation signal from the administrator, it becomes a memory detector. The memory detector will be activated quickly by once antigen matching, which improves the efficiency of detecting known aging. More information about the life-cycle management and the memory detector please refer to [12].

IV. Experiments

The aim of our experiments is to evaluate the effect of the real-valued negative selection algorithm on the detection of web server aging. For the purpose of experimentation, Apache is used as the web server to be monitored. Apache is one of the most popular web server systems [1], it usually responds to business requests stably, but shows aging phenomenon when faults appear in application software deployed in it.

As clients communicate with the Apache through HTTP protocol, all experiments are performed in a small local area network (LAN, shown as fig. 5). The LAN consists of two machines, one is the server running Apache version 2.0.54 on a Linux platform (version RHEL 5, CPU: Pentium(R) Dual-Core 2.80GHz, RAM: 1GB), the other is the client operating Linux monitoring tool developed by our research group (called linuxAgent) and httperf [13].

The linuxAgent is designed to monitor the resource usage information (e.g. phyMemory usage, swapSpace usage and loadAvg) of the server through the /proc virtual file system of the Linux OS. Httperf is well known not only for its ability to generate workload, but also as performance monitor of the web server, the replyRate and responseTime of Apache that compose the antigen of this paper can be provided by it.

The experiment is composed by three steps corresponding to the RNSA: collecting self, training detectors and detecting web server aging.

*1) **Collecting self**:* First, the server and the Apache are restarted, and the httperf is executed periodically to generate requests accessing web pages with sizes of 500 bytes, 5 kb, 50 kb and 500 kb (similar to [1]) in the Apache with probabilities of 0.35, 0.5, 0.14 and 0.01 respectively. Then, the indicators composed the antigen is collected as self samples by linuxAgent for 25 days with an interval of 30 seconds.

To make the Apache run stably in this step, we employ a value of 50 requests per second as the connect rate of the httperf, and the configuration parameters MaxClients and MaxRequestPerChild of the Apache is set to 250 and 100 respectively (as shown in table IV-A).

*2) **Training detectors**:* Matured detectors are trained by the detector generation algorithm described in section 3 using self samples acquired in last step as input.

*3) **Detecting web server aging**:* In the phase of real-time aging detection with RNSA, we employ the matured detector population generated in last step to discriminate the antigens that represent the web server status. The result can classify the web server performance state as healthy or abnormal. If the distance between the antigen A_g and any matured detector is less than or equal to r, it proves that the web server ages when A_g occurs.

Our experiment has two scenarios. The httperf is operated to generate requests as in step 1) for 4500 seconds in each scenario. Each scenario is repeated for 5 times.

The aim of the first scenario (S1) is to verify our detection method when the Apache runs normally because of the absence of faults. So, the same parameters are set as in step 1).

The aim of the second scenario (S2) is to verify our detection method when the Apache shows evidence of aging after being inserted into a memory leak accidentally. For the purpose of speeding up the Apache aging, we should make it overloaded. As a consequence of investigation, the variable MaxRequestPerChild is adjusted to 0 [1], and the connection rate of the httperf is set to a value of 800 requests per second.

A. Principles to Set Parameters

The parameters in the RNSA were set as follows: $N_h = 50.451(\%)$, $N_l = 7.787(\%)$, $N_{sh} = 575716(kb)$,$N_{lh} = 1.01$,$N_{rth} = 2.936(ms)$. They were set based on the statistical results of the monitoring values of the indicators described in section 4.1. Take freeMemRatio for example, the average monitoring value was 29.119(%), the standard deviation was 21.332(%). Therefore, we set $N_h = 50.163 + 33.429 = 83.592(\%)$, $N_l = 29.119 - 21.332 = 7.787(\%)$. The population of detectors was set to more than 120, k was set to 5, ξ was set to 12, η was set to 0.3 (as shown in table IV-A).

B. Results

We analyzed the results of the experiments described above. First, let's have a look at the monitoring values of indicators in every scenario. For both scenarios, the x-axis

TABLE I.
PARAMETERS USED IN THE EXPERIMENTS

Parameter	Description
MaxClients	maximum number of server processes allowed to start
MxRqstPCld	MaxRequestPerChild: maximum number of requests a server process serves
connectRate	number of requests the httperf sent to the server (/s)
k	the number of nearest self points considered when calculating the radius
ξ	the maximum allowable overlap among the detectors
η	variable that provides the level to move a detector

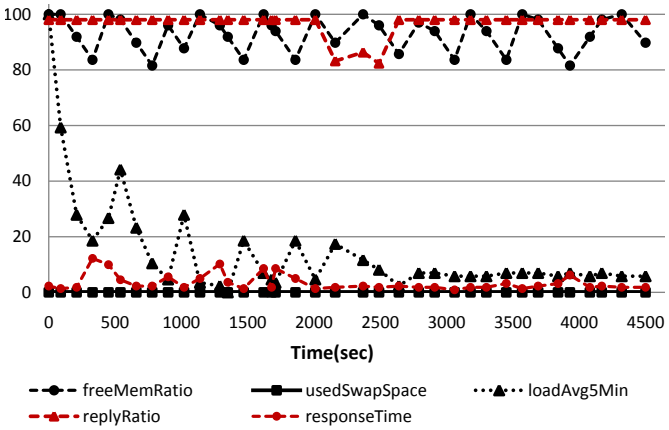


Figure 6. Status of the Apache in S1

represents time in seconds while the y-axis represents the normalized value of the indicators.

In S1, although without source of aging injected, we noticed that the performance status of the Apache is changing over time, and there is specificity among the indicators. The loadAvg5Min shows big variation, while the usedSwapSpace, the responseTime and the replyRatio are relatively stable in fig.6. The data is from a randomly selected experiment because there is a small variation among the data from the repeated experiments. In addition, we observed an seasonal feature of the freeMemRatio, this may be due to the child process is killed by the Apache after handling scheduled number of requests (controlled by configured parameter MaxRequestsPerChild (=100 in S1)) , then the memory it employs is released.

In S2, a memory leak is injected into the web site in the Apache, we observed abnormal states of the OS resource usage and the performance of the Apache. As shown in fig. 7, we found a sharp decline of the freeMemRatio, and an abrupt increase of the usedSwapSpace with a little variation, but the replyRatio and the loadAvg are similar to 6. We also noticed that there is a large variation of the responseTime.

Table IV-B shows the results from the RNSA for the detection of the aging of the Apache. The values in the table are the average and the standard deviation in five repeated experiments. The most notable observation about

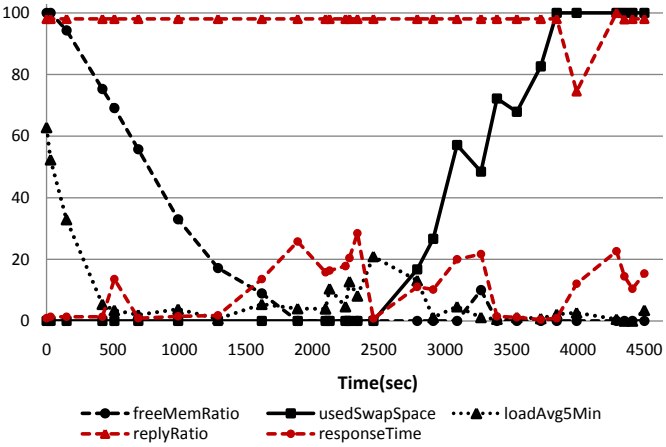


Figure 7. Status of the Apache in S2

TABLE II.
RESULTS FROM THE EXPERIMENTS

Scenario	A_g Type	Num of A_g	Num of A_g detected	FP/FN rate(%)
S1	self	151	0	FP=0(0)
S2	non-self	151	135.7(0.58)	FN=9.49(0.38)

the data in the table is that the algorithm performs much better in practice than we expected with low FP and FN rate. The FP (false positive) rate corresponding to the ratio of self antigens that be mistaken as non-self antigens by detectors. The FN (false negative) rate corresponding to the ratio of non-self antigens that be mistaken as self antigens by detectors.

C. Discussions

The results of the experiments show that the RNSA could identify the aging state from the normal state when the Apache is running. The RNSA is able to recognize previously unseen aging, as the detectors are trained to match the the aging (non-self antigen) independent of aging samples. In addition, the coverage of the detector is not constant because the radius of the detector is calculated according to its 'position' relative to self antigens and other detectors dynamically. This makes it able to identify various aging conditions of the web server systems. It could be concluded that the RNSA has a high detection rate and low false alarm rate (including FP and FN rate) when applied to the detection of web server aging.

During the experiments, we discovered that the Euclidean distance was not suitable for n-norm problem space where $n \geq 2$, as the distance may overstep the boundary. But *Minkowski* distance $Minkowski(c, x) = (\sum |c_i - x_i|^n)^{1/n}$ which equals to the Euclidean distance when n=2 follows the rule of the boundary. We also noticed that the detector that has been moved would be out of boundary if its new center hasn't been normalized. However, it's difficult to avoid that part of the detector with center normalized falls outside of the boundary of

the application problem, these detectors may be called boundary detectors. The future work is to explore how to restrain the number of the boundary detectors.

V. Conclusion

Experience is the best teacher. In this paper, we carried out experiments to evaluate the performance of RNSA on web server aging detection, with an Apache web server instance. In the experiments, the httperf is used to generate artificial workload and collect performance indicators of the Apache. The linuxAgent designed by our research group is used to monitor the resource usage information from the /proc virtual system of Linux. In addition, we described the principles to set the parameters that have an impact on the results of our approach.

Due to the investigation of the experiments, we proposed some improvements of the RNSA in [3] to make it more suitable for real problems. They are the adjustment of the detail algorithm process, the normalization of the antigen and the center of every detector, improvement of the matching rule and the evaluation of the overlap among detectors.

Experimental results show that the improved RNSA has a high detection rate and low false alarm rate when applied to web server aging detection. The future work is to explore how to restrain the number of the boundary detectors to enhance the performance of our method.

VI. Acknowledgment

This work is supported by the Defense Industrial Technology Development Program (No. A1420080183) from Ministry of Education of China. This work is also supported by the Research Grant (No. 61170306) from National Natural Science Foundation of China (NSFC).

References

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

[2] L. Li, K. Vaidyanathan, and K. Trivedi, "An approach for estimation of software aging in a web server," in *International Symposium on Empirical Software Engineering (ISESE'02)*. IEEE Computer Society, 2002.

[3] H. Yang, Y. Liang, C. Tan, and J. Fu, "Detecting software aging of web servers with real-valued negative selection algorithm," in *the 2nd International Conference on Data Storage and Data Engineering*. IEEE, 2011, pp. 298–302.

[4] A. Andrzejak and L. Silva, "Using machine learning for non-intrusive modeling and prediction of software aging," in *Network Operations and Management Symposium (NOMS '08)*. IEEE, 2008, pp. 25–32.

[5] A. Andrzejak, L. Silva, and D. E. Informćtica, "Robust and adaptive modeling of software aging," 2008.

[6] J. Greensmith, A. Whitbrook, and U. Aickelin, "Artificial immune systems," in *International Series in Operations Research & Management Science*, 2010, pp. 421–448.

[7] Y. Huang, C. Kintala, N. Kolettis, and N. Fulton, "Software rejuvenation: Analysis, module and applications," in *Twenty-Fifth International Symposium on Fault-Tolerant Computing*. IEEE Computer Society, 1995.

[8] F. Gonzalez and D. Dasgupta, "Anomaly detection using real-valued negative selection," *Genetic Programming and Evolvable Machines*, pp. 383–403, 2003.

[9] F. Gonzalez, D. Dasgupta, and J. Gomiz, "The effect of binary matching rules in negative selection," in *Genetic and Evolutionary Computation (GECCO '03)*, 2003, pp. 198–198.

[10] J. Zhou, "Negative selection algorithms: from the thymus to v-detector," Ph.D. dissertation, The University of Memphis, 2006.

[11] Y. Jia, J. Su, and K. Cai, "A feedback control approach for software rejuvenation in a web server," in *International Conference on Software Reliability Engineering Workshops*. IEEE, 2009, pp. 1–6.

[12] S. Hofmeyr and S. Forrest, "Architecture for an artificial immune system," *Evolutionary Computation*, vol. 8, no. 4, pp. 443–473, 2000.

[13] D. Mosberger and T. Jin, "httperf: a tool for measuring web server performance," *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 3, pp. 31–37, 1998.

Author Biographies

Yiwen Liang received a B.S., a Master and a PhD degree in computer science from Wuhan University, Wuhan, China, in 1983, 1990 and 2002, respectively. Since 1983 he works for the Wuhan University in the School of Computer Science where he is now a Professor, doctoral supervisor of Computer Science and the leader of the Department of Computer Application.

Prof. Liang currently holds a NSFC Fellowship focusing on AIS, health management of the information system and anomaly detection. He has been awarded three NSFC research funding as principal investigator on topics including AIS, Danger Theory, Network Security and Anomaly Detection. Prof. Liang is an associate Secretary General of the Natural Computing Committee of Chinese Association for Artificial Intelligence (CAAI), a member of the council of the IEEE education association China sub-commission.

Huan Yang is a PhD student in School of Computer Science at Wuhan University. She received a B.S. degree in Computer Science and Technology from Huazhong Normal University, Wuhan, China in 2007, and she is in a combined M.S.-P.H.D. program at Wuhan University from 2007. Her research interests are in the field of software reliability, artificial immune system, and software health management. Currently, she focuses on applying the AIS on web server aging.

Jun Fu is a PhD student in School of Computer Science at Wuhan University. He received a B.E. degree in Computer Science from Wuhan University in 2006, and he is in a combined M.S.-P.H.D. program at Wuhan University from 2006. His research interests include artificial intelligence, artificial immune system, network security and malware detection.

Chengyu Tan is currently an associate professor in the School of Computer Science at Wuhan University. She received a B.E. (1990) and a M.S. (1996) degrees from Wuhan University of Hydraulic and Electrical Engineering, and completed a PhD in software and theory at Wuhan University in 2007. Her research interests include artificial immune system and natural computation.

Aolin Liu is a Master student in School of Computer Science at Wuhan University. She gained a B.E. degree from Wuhan University in 2010.

Shiwen Zhu is a Master student in School of Computer Science at Wuhan University. He received the B.E. degree from Wuhan University,Wuhan, China in 2011, and at present he is pursing his M.S. degree in Wuhan University.