A LOG-BASED TRACE AND REPLAY TOOL INTEGRATING SOFTWARE AND INFRASTRUCTURE

Noriko Hanakawa [1]and Masaki Obana[2]

[1] Information management department, Hannan University, Osaka, Japan
[2] Department of Information Science and Technology,Osaka Institute of Technology,Osaka, Japan

ABSTRACT

We propose a log-based analysis tool for evaluating web application computer system. A feature of the tool is an integration software log with infrastructure log. Software engineers alone can resolve system faults in the tool, even if the faults are complicated by both software problems and infrastructure problems. The tool consists of 5 steps: preparation software, preparation infrastructure, collecting logs, replaying the log data, and tracing the log data. The tool was applied to a simple web application system in a small-scale local area network. We confirmed usefulness of the tool when a software engineer detects faults of the system failures such as "404" and "no response" errors. In addition, the tool was partially applied to a real large-scale computer system with many web applications and large network environment. Using the replaying and the tracing in the tool, we found causes of a real authentication error. The causes were combined an infrastructure problem with a software problem. Even if the failure is caused by not only a software problem but also an infrastructure problem, we confirmed that software engineers alone could distinguish between a software problem and an infrastructure problem using the tool.

KEYWORDS

Web application, infrastructure, web server, network device, seamless debugging, detecting faults, maintenance

1. INTRODUCTION

Recently, large-scale computer system is important for human life in modern society. Large-scale computer system consists of large software and large infrastructure[1]. Software means application package software, middleware, and application server software, and application client software. Infrastructure includes network equipment, server computers, client computers, mobile terminals, various operating systems, and physical and logical network design. Infrastructure is designed according to 5 layers with considering software design [2]. Once large-scale computer system starts working, various system failures may occur. Some failures may be caused by physical breakdown of equipment, some failures may be caused by setting errors of network equipment's configuration files, and some failures may be caused by software faults. It becomes more difficult to detect causes of system failures as computer system becomes more complicated [3]. Software engineers alone cannot detect causes correctly, infrastructure engineers alone cannot find causes efficiently. Only engineers who have both infrastructure knowledge and software knowledge can certainly detect causes of such system failures[4]. However, in current industry, infrastructure engineers are clearly distinguished from software engineers. The two type engineers are assigned to quite different tasks and works. Infrastructure engineers learn infrastructures knowledge and techniques without software knowledge, software engineers learn software knowledge and techniques without infrastructure knowledge. Large-scale computer system becomes more complicated and software and infrastructure technologies become

DOI:10.5121/ijsea.2018.9401

complicated more and more. Although the higher technologies of infrastructure and software are deeply influenced each other, engineers are clearly divided into infrastructure engineers and software engineers. It is difficult for software engineers to detect software faults caused by network environment because of lack of infrastructure knowledge. It is difficult for infrastructure engineers to detect infrastructure faults depended to software architecture.

Therefore, we propose an analysis tool for integrating infrastructure and software. Especially, we focus on web application system that consists of application software, client software, middleware such as web server software and database software, server computers, client computers, and network equipment. A feature of the tool is the usage of various system logs such as operating system logs, network equipment logs, web server logs, and application logs[5]. Based on the logs, we can graphically replay system failure phenomena on the tool. The replay of failure phenomena includes not only data transfer on software programs but also a movement of packets in the network environment. Even software engineers who have little infrastructure knowledge can seamlessly trace software program data transfer and packet movement on network environment. Even infrastructure engineers who have little software knowledge can smoothly understand data transfer from software to infrastructure. Using our too, both engineers can efficiently detect faults of software or infrastructure in the complicated computer system.

The main gap between conventional scientific knowledge and our research is a log analysis technique mixing software and infrastructure. Infrastructure log analysis is useful to detect system failures caused by infrastructure faults. On the other hand, software log analysis is usual techniques in software debugging process. Usually, these analyses are separated by different kinds of engineers. The approach of the log analysis technique mixing software and infrastructure is a new aspect of our research. In addition, we have proposed a new process metrics for infrastructure [4] and infrastructure log analysis [2]. The proposed tool is partially based on the ideas of the previous researches' results. In addition, the tool is applied to the industrial real logs in order to detect causes of system faults. The paper evaluates the usefulness of the tool using a huge amount of log.

In this paper, section2 shows related works. Section 3 shows an outline of infrastructure design steps. Section 4 explains the tool for web applications compute system integrating software and infrastructure. A trial of the tool is shown at section 5. A practice of the tool in a real computer system is shown at section 6, section 7 shows summary and future researches.

2. RELATED WORKS

There are many researches of developing web applications. Elmam et al. proposed web application test techniques using session data [6]. The session data is gathered as users operate web applications. The method helps test those applications from a functional standpoint. As a result, the test techniques can generate more efficient test data than test data generated by a white box test technique. The session data is a kind of system logs. The approach of collecting logs is similar to our approach. However, the target of the research is different. Out target is not only web application software but also infrastructure. The scoop of the research is different. In addition, Wang et al. proposed JSTrace for fast reproducing web application errors [7]. Record-replay techniques are used to reproduce errors in web applications. However, the long event trace data prevent from efficient reproducing errors. The JSTrace is a tool to eliminate unnecessary event trace data of JavaScript. As a result, 97% event trace data was eliminated. The JSTrace is similar to the tracing and the replaying of our tool. The tracing and the replaying also eliminate unnecessary log data by time and HTTP header data. The target of the JSTrace is only JavaScript program. Our proposed tool includes software logs and infrastructure logs. Thetool is useful when you do not know if the system failure is due to software defects or infrastructure defects.

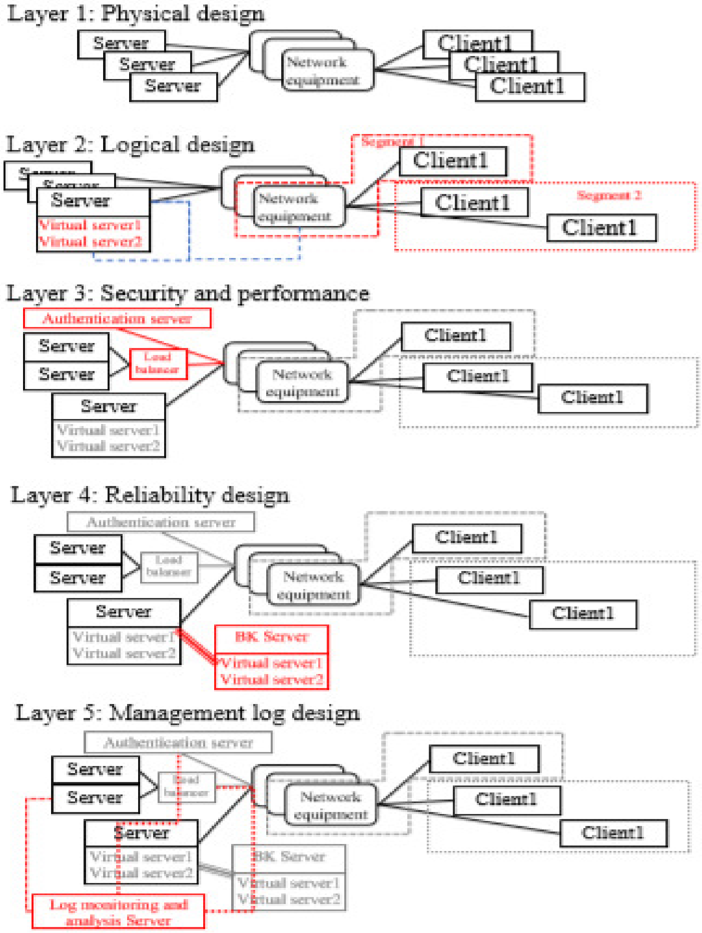


Fig.1.5layers for infrastructure design

On the other hand, there are many useful infrastructure trace and analysis tools[8]. Wireshark is used for research of the bottlenecks associated with packet capturing using commodity hardware in local area networks (LANs) without losing data [9]. And, Wireshark is useful in universities' education for network environment with web applications [10]. Wireshark is free software. HTTP requests and HTTP response and TCP traces are collected in the Wireshark [11]. Of course, purpose of tools such as Wireshark is only network analysis of infrastructure, not software. DynaTrace is also a tool for analysing infrastructure [12][13]. The DynaTrace transaction traces among computer servers and web applications are available. On the other hand, Visual studio .net framework is useful tool for tracing web applications. The Visual studio .net framework [14] helps software engineers detect program faults. Our tool including the replaying and the tracing is a similar tool that combined the Wireshark with the Visual studio .net framework. Purpose of our tool is seamless tracing and replaying not only software logs but also infrastructure logs.

3. INFRASTRUCTURE

This section shows an outline of infrastructure construction [2]. Fig.1 shows a process of constructing infrastructure with 5 layers. Infrastructure is designed according to the 5 layers with considering software design. In addition, the 5 layers may not be necessarily carried out in order.

3.1. LAYER1: PHYSICAL DESIGN

The physical design of infrastructure is simple. Server computers and client computers and network equipment such as routers, switching hubs are set up for physical areas such as buildings, rooms. Each equipment is connected with the other equipment by network cables or wireless communication. Computers and network equipment at the physical design phase are minimum configurations of the computer system.

3.2. LAYER 2: LOGICAL DESIGN

The logical design of infrastructure is influenced by software designs. For example, if you gave three kinds of web applications, you should construct three application server computers and web servers. If you need e-mail function, you should prepare SMTP server and POP server. Usually, server computers are set up to several virtual machines. Each virtual machine is set up as a server computer. The server computer on the virtual machine is set up a server such as a web server, SMTP server. In addition, client computers and network equipment are categorized to several virtual LANs. Virtual LAN is useful when network environment is huge. A virtual LAN is a meaningful area in the logical design. For example, computers in a section of a company are set up a virtual LAN, even if the section is divided into two groups that are geographically distant.

3.3. LAYER 3: SECURITY AND PERFORMANCE DESIGN

Up to the layer 2, the computer system can basically operate. Next, we have to consider security and performance of the computer system. To achieve high security, we have to prepare authentication servers and firewall servers. To achieve high performance, network equipment such as road balancers is added to the computer infrastructure.

3.4. LAYER 4: RELIABILITY DESIGN

Next, we have to improve reliability of the computer system. Database system and applications are copied to the other computers in order to recover data and system when the database is crashed, or the server is broken.

3.5. LAYER 5: MANAGEMENT LOG DESIGN

Finally, we design monitoring and analysis functions for the computer system using log data on server computers and network equipment. Usually, several servers for monitoring and analysis are added to the infrastructure. Administrators manage the infrastructure statuses through the monitoring and the analysis-tools.

4. THE PROPOSED ANALYSIS TOOL

4.1. PURPOSE OF THE TOOL

The purpose of the tool is that software engineers detect proper causes of system failures by themselves after computer system starts running. The feature of the tool is usage of system logs among network equipment, servers, middleware, and application software. Especially, the tool focuses on web application system. Software engineers usually have little infrastructure knowledge. Software engineers need assistance from infrastructure engineers when complicated system failures occur on web application system. Especially, we aim that software engineers alone can distinguish infrastructure faults with software faults through the tool. In addition, a target of the tool is web application system. Therefore, we focus on tracing and replaying HTTP requests and HTTP responses.

4.2. An Outline Of The Tool

Fig.2 shows an outline of the tool. The tool consists of 5 steps. The first, second, and third steps are preparation for starting the tool. After starting the tool, tracing and replaying are started in the tool. At first, engineers set up configurations of servers and network equipment at a constructing infrastructure phase. Next, engineers make software programs with functions of outputting log. The program are parts of a web application. After that, the computer system starts working at an operation phase. Many users use the computer system with recording logs. Next step is when system failures occur. The replay tool is useful. The replay tool can replay the computer system situation including infrastructure and software even if time has passed since the failure occurred. The tool can be rewound until the failure occurrence time. Finally, using thetool, software engineers can seamlessly trace HTTPrequests and HTTP responses on both infrastructure and application software.

4.3. Essential Elements Of Log

Network equipment, various servers, middleware and application software output different log formats and various information. Therefore, in the tool, essential elements of logs are defined as follows,

- Time : date/month/year:hour:minute:second+time zone
  (an example: 04/Jun/2017:16:39:09 +0900)
- IP_{source} : Source IP address
- $IP_{destination}$: Destination IP address
- Protocol : HTTP, DHCP, SMTP, FTP, TLS, etc.
- Info : various messages
  an example
  "GET /wiki.cgi?page=ABC HTTP/1.1" 200 5408, or
  “Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.79 Safari/537.36 Edge/14.14393", or
  program name: XXXX.phpVariable: data1 = “abc”, data2=”def”, data3=1

The above 4 elements of the “Time”, the “IPsource”, the “IPdestination”, and the “Protocol” have to be collected on each network equipment, each server, middleware, application software. The above “Info” element is also collected if possible. The data of “Info” is required in the step5 “Tracing” of the tool (See Step5:Tracing subsection of this section). If the “Info” data is sufficiently collected in computer system, the tracing of the tool will be better. Quality and quantity of the “Info” data influence the success of the tracing.

On the other hand, we have to consider performance of network equipment, servers and software. Logs require huge storages and much processing time. If all messages are recorded to logs on a core switch device, the performance of the core switch will be suddenly low. Therefore, we have to consider roles of network equipment and servers. Some network devices may output only “Time”, “IPsource”, “IPdestination”, and “Protocol”. Some servers may output all messages. An example of the collected data guideline is as follows,

(1). Non intelligent network device: collecting no data
(2). Intelligent network device such as core switch: “Time”, “IP_{source}”, “$IP_{destination}$”, and “Protocol”
(3). Non-core network device such as distribution switch: all data

(4). Fire wall server: “Time”, “IPsource”, “IPdestination”, and “Protocol”

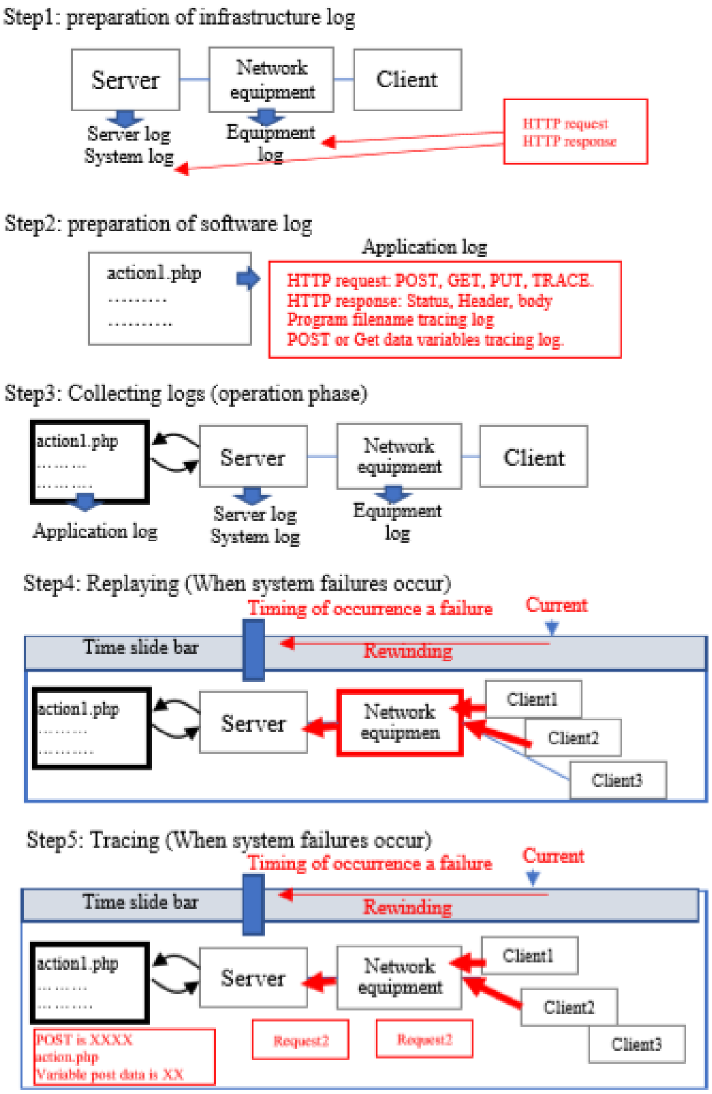


Fig.2 Five steps for the tool

(5). Web server, DB server: default format log includes “Time”, “IP$_{source}$”, “IP$_{destination}$”, and “Protocol”

(6). Web Application: HTTP request information, program name, variables, and other program information (software engineers can make program codes in order to output necessary information into logs in web applications)

In this way, non-intelligent devices do not output any logs, some core switches cannot record “Info” data. Web applications can output various information for replaying and tracing. Therefore, we should consider how the missing data is covered. If some network devices cannot record any logs, the missing logs may be covered with the logs of neighboring intelligent network devices. The missing data is covered by the covering techniques [15].

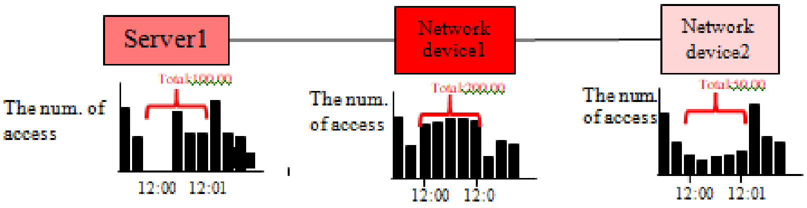


Fig.3 Images of heavy load of servers and devices

4.4. Step1:Preparation Of Infrastructure

Various servers also output various log. Network equipment outputs various tracing log. Engineers set up configuration files so that HTTP requests and HTTP responses can be recorded. If log volume is huge, engineers record only the above 4 essential elements (See 4.3. Essential elements). The log recording functions usually are supported by operating system, server software, middleware and network equipment.

For example, a typical log of Apache web server is as a follow,

```
------------------------------------------
150.89.2.6--[04/Jun/2017:16:39:06 +0900] "GET /wiki.cgi?page=3 HTTP/1.1" 200 5408 "-" " Edge/14.14393" "150.89.13.147"
-------------------------------------------
```

In this case, Time is “04/Jun/2017:16:39:06 +0900”, $IP_{destination}$is “150.89.13.147”, IP_{source} is “150.89.2.6”, Protocol is “HTTP”, Info is “ GET /wiki.cgi?page=3 HTTP/1.1" 200 5408 "-" " Edge/14.14393”.

4.5. Step2: Preparation Of Software

Software engineers make software program including trace functions. The trace functions mean that HTTP requests and HTTP responses can output logs. In addition, program file names and data of variables are recorded to the log files. Program file names are useful to recognize behaviours of program. The data of variables are useful to recognize correct receiving data through GET or POST methods. After that, the HTTP response are generated. The HTTP response data is also recorded into the log files.

Examples of php programs for HTTP request are as follows.

- $Time == date("Y-m-d H:i:s")
- $ IP_{source}=$_SERVER[‘REMOTE_ADDR’];
- $$IP_{destination}$=$_SERVER[‘SERVER_ADDR’];
- $ Protocol=$_SERVER[‘SERVER_PROTOCOL’];
- $Client type=$_SERVER[‘HTTP_USER_AGENT’];
- $Info = $ file_get_contents('php://input');

If you need the other data, various data such as user request time($_SERVER[‘REQUEST_TIME’]) can be recorded.

4.6. Step3: Collecting Logs

After the computer system development finishes, the computer system starts working on real business processes. Many users use the computer system. Operating systems and servers, middleware, network equipment and web applications and software are continuously outputting log files. The logs accumulated in a log server (See Layer 5 of Fig.1). Basically, the essential elements are recorded in the logs.

4.7. STEP4: REPLAYING

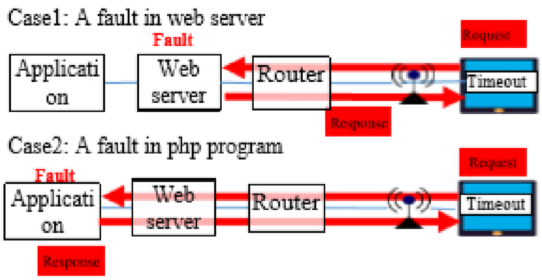


Fig.4 Images of tracing a HTTP request and response

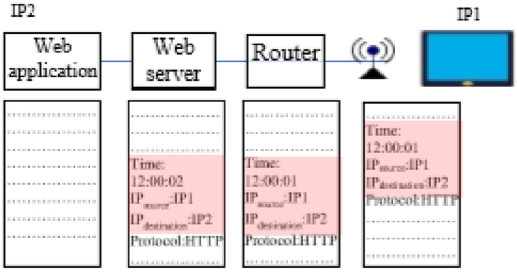


Fig.5 The tracing mechanism of the tool

When a system failure occurs, the replay tool is useful. The tool collects automatically all log files on the computer system. The most feature of the tool is that software engineers can rewind time. Usually, system failures suddenly and unexpectedly occur anytime. Engineers cannot immediately investigate the system failures. If time passes from the failure occurrence, same failures may not occur because network situation and server situation and application situation are different. Especially, system failures that sometimes occur are difficult to investigate because of non-reappearance of the failures. Moreover, software engineers and infrastructure engineers often try to reappear the failures on a test computer environment. The test computer environment is greatly different from real computer system environment. The number of users and the number of accessing servers and the number of clients are greatly different. If the failures are influenced by such volume of accessing computers or servers, the reappearance of the same failures is difficult on the test computer environment.

Therefore, the replay tool is useful. The tool can replay the real situations of network, servers and web applications. Engineers can recognize all situations when the system failures occurred. For example, in step 4 of Fig.2, huge clients try to connect to one network device. The load of the network device suddenly becomes heavy. The red rectangle of the network device presents heavy load. Even software engineers can understand problems of the heavy load of the network device. Therefore, the tool helps software engineers detect correct faults for the system failures.

Then, the heavy load of servers or network equipment is calculated by the logs. For example, if you rewind time to 12:00, the replay tool collects logs that were recorded from 12:00 to 12:01 (See Fig.3). The number of accesses of the Server 1 of Fig.3 is100,00 from 12:00 to 12:01. The number of accesses of the Network device 1 is 200,00, the number of accesses of the Network device 2 is 50,00. The number of accesses is a basis of the colour of the rectangle of a device. In Fig.3, Network device1 has deep red colour because the Network device1 has the biggest accesses. Network device2 has light red colour because the Network device2 has the smallest accesses. In this way, the replay tool visually presents load statuses of all servers and network equipment. Software engineers can easily understand network load status at the failure occurrence time in the replay tool.

4.8. STEP5: TRACING

The tracing is the most characteristic function in the tool. The tracing helps software engineers distinguish software problems with infrastructure problems. Sometime, an incomprehensible system failure occurs. The incomprehensible failure mean that causes of the failure are not clear.

The incomprehensible failure may be caused by infrastructure faults, or the incomprehensible failure may be caused by application program faults.

For an example, a user accesses a web application through a smartphone with Wi-Fi network. A “Forbidden error” frequently occur. The “Forbidden” failure usually occur when permissions of file and folders are wrong in the web application system. Also, the “Forbidden” failure occur when “Deny” command setting is wrong in web server Apache. The “Forbidden” error may occur by both program faults and infrastructure faults. In infrastructure problem case, the httpd.conf file for Apache web server has “Deny from 150.9.160.45” command. If the IP address is wrong, a client computer cannot access the server computer with the “150.9.160.45”. As a result, “Forbidden error” occur on the client computer.

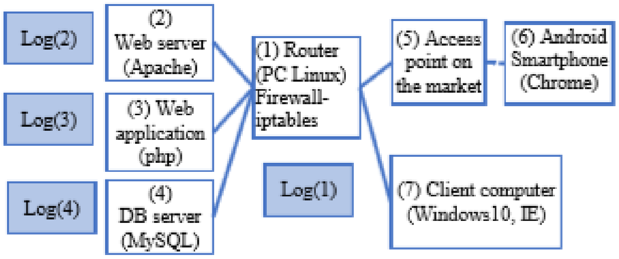

Log(2)
(2) Web server (Apache)
Log(3)
(3) Web application (php)
Log(4)
(4) DB server (MySQL)
(1) Router (PC Linux) Firewall-iptables
Log(1)
(5) Access point on the market
(6) Android Smartphone (Chrome)
(7) Client computer (Windows10, IE)

Fig.6 A trial computer system for the evaluation

On the other hand, if a php program has wrong code, the client computer also has “Forbidden error”. The php program has a function of making list of a specific folder. The php program has as a follow:

```
foreach(glob(abc/*') as $file){ file_get_contents($file); }
```

If the “abc” folder includes an unreadable file, the “Forbidden error” occurs on the client computer. Of course, the php program has to check readable files or not. A software engineer alone cannot detect a fault of setting the httpd.conf file. An infrastructure engineer alone cannot detect a fault of the php program code. In this way, the system failure such as the “Forbidden error” depends on not only program faults but also infrastructure faults.

In the case, the tracing function of the tool is useful. Images of the tracing is shown at Fig.4. If the httpd.conf file is wrong, the HTTP response will return from the web server. If the php program code is wrong, the HTTP response will return from the web application. The red arrows of Fig.4 mean the “Tracing” result. A software engineer alone can understand the wrong setting of the web server in the case 1. An infrastructure engineer alone can understand the wrong program code in the web application in the case2.

Next, a mechanism of the tracing is described. Fig.5 shows an outline of the mechanism. We assume that a system failure occurs at 12:00:00, the IP address of the smartphone is “IP1”, the IP address of the web application is “IP2”. The tracing function searches a log record line with “Time is around at 12:00:00”, “IPdestination is IP1” or “IPsource is IP1” and “Protocol is HTTP”. If the data of the 4 elements are recorded in the logs, the device and the server received the HTTP requests and the HTTP responses. If the data of the 4 elements are not recorded in the log, the device and server did not receive the HTTP requests and the HTTP responses. In the case of Fig.5, the access point, the router, the web server received the HTTP request. However, the web application did not have the log of the HTTP request. Therefore, the tracer of the tool presents the result like the case 1 of Fig.4. The mechanism of the tracing is simple. If the logs have sufficient "Info" data, the correctness of the tracing function will be improved because the tracer gets useful information to classify each HTTP request and each HTTP response.

5. An Evaluation Of The Tool

This section shows usefulness of the tool in the limited computer system called as "a trial computer system".

5.1. A Trial Computer System

The trial computer system is shown at Fig.6. The trial computer system was built at my laboratory in local area network. The trial computer system consists of a web server (Apache) [16], an application (php program), a database server (MySQL) [17], a router, an access point, a smartphone client, and a client computer. The router was built in a computer with Linux. The router computer is set the firewall with iptables [18] service. The access point is goods on the market. The access point does not have a function of outputting logs. Client terminals are a smartphone and a notebook computer. The smartphone is an android terminal with chrome browser. The notebook computer is set up windows10 with the IE 11 browser.Fig.7 and Fig.8 show images of the tool. We use the tool in a part of visualization of the tracing function and the replaying function in the trial case.

5.2. Step1: Preparation Of Infrastructure

We set up the devices and the servers for recording logs. The recording logs are shown at TABLE1. All logs include the 4 essential elements (Time, IPsource, IPdestination, and Protocol).

Table I. Logs Of The Tral Computer System

(1) Router	Jun 27 19:45:41 room8306-infratest kernel: [F_FOR] IN=eth0 OUT=eth1 SRC=192.168.1.24 DST=150.9.162.46 LEN=52…..
(2) Web server	150.9.160.45 - - [27/Jun/2017:19:45:41 +0900] "GET /test.php HTTP/1.1" 403 288….
(4) DB server	140423 10:30:11 [Warning] '--default-character-set' is deprecated and will be removed in a future release. Please use '--character-set-server' instead….
(5) Access point	NON

5.3. Step2: Preparation Of Software

The web application of Fig.6 is prepared. The application is built in php language. The php program also has functions of outputting "php program file names", "function names in the php programs", "values of important variables", "header of HTTP request", and "body of HTTP request".

5.4. Step3: Collecting Logs

The trial computer system starts working. The servers start running. The client computer and the smartphone access the web application using URL (http://150.9.162.46/test.php) with browsers.

The logs of Fig.6 are recorded in the trial computersystem. The two system failures occurred. The first failure is “not found error (404)”, the other failure is a no response error.

5.5. STEP4: REPLAYING

Fig.7 shows a result of the replaying in the tool. At first, we move the time slide bar on the tool. We can rewind time to the failures’ occurrence time. Then, we can see how heavy the loads are. The loads are presented as red rectangles in Fig.7. The red rectangles are at the bottle of the router and the servers. Because this trial computer system is very simple, the logs were not so much. The web application was accessed several times from the only two clients. The loads of the servers and the router were not heavy. In addition, the rectangle of the database server is not red. That is, anyone did not access the database server. In this way, load statuses and running status are presented as the red rectangles at the time of the slide bar.

5.6. STEP5: TRACING

The tracing result is shown at Fig.8. At the time of the failures occurrences, we click the notebook computer. The tracing starts on the tool. At first, IPsource and Time of the essential elements are determined. The value of IPsource is IP address of the clicked notebook computer. The value of Time is the failure occurrence time. Next, IPdestination and Protocol of the essential elements are determined.

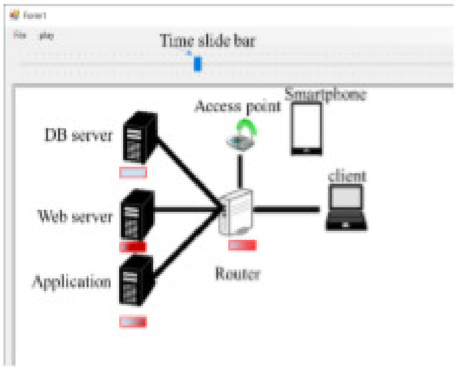


Fig.7 Replaying result in the trial case

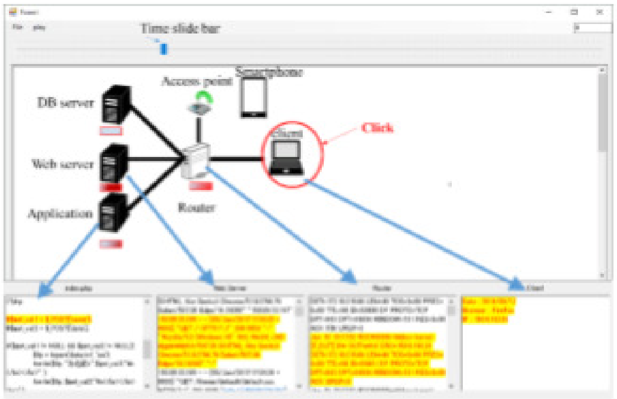


Fig.8 The tracing result in the trial

The value of IPdestination is IP address of the web application. Protocol is HTTP because this system is a web application system. Based on the values of the 4 essential elements, the tool searches the log files. If lines of the log files include the same values of the 4 essential elements, the lines of the log files are coloured. The lower part of Fig.8 shows the coloured lines of the log files. The yellow coloured lines mean the lines that have the same values of the 4 essential elements. Therefore, we can trace the HTTP request from the clicked notebook computer.

In addition, the web application tracing is most characteristic. The web application is built in php language. The php program has codes of outputting the essential elements such as IPsource and IPdestination to a log file. Moreover, the php program file names, the function names and the main variables' values are also recorded to the log file. The codes of the recording the program file name and the variables are embedded into the php program. The log file of the php program also has a role of debugging information. The program file names and the values of the variables in the log file are useful to detect software faults in a normal debugging process. The tracing tool can present the program file names and values of the variables that are synchronized with the other logs of devices and servers. An example of the web application logs is as a follow:

```
------------------------------------------------------
Program file name: index.php
Time: 12:00:00
IPsource: 150.9.160.2
IPdestination: 150.9.160.5
Protocol: HTTP
Info: row data of POST
Variable $name = hanakawa
Variable $address = Japan
Variable $tel = +81001001001
Function name: CheckSameName ()
……………
------------------------------------------------------
```

The above log from the php program is sufficiently useful to detect faults in php programs. In this way, the logs of the tool are seamless data without distinction between infrastructure and software.

5.7. EVALUATION OF THE TOOL IN “NOT FOUND ERROR(404)”

The system failure “Object not found error (404)” occurred in the notebook computer.

1) **Case1:** problems in the web server

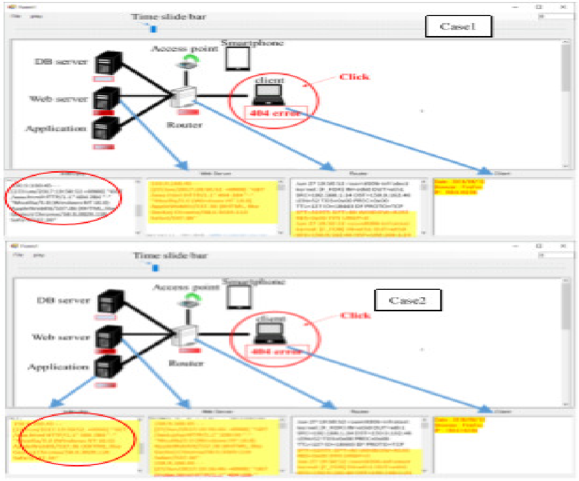


Fig.9 The tracing result in the 404 error case

The configuration file of the web server Apache has a “DocumentRoot” attribute. Usually, the value of the document root is set up a default value “htdocs”. However, if the document root is a wrong value, the first page “index.php” will be “Object not found error (404)”.

2) **Case2:** problems in the php programs

The php program generates html tag codes such as “<a href =”c:/user/hanakawa/htdocs/abc/check.php”>”. The code includes an absolute pathname. The php program does not run correctly on the web server because the pathname of the “c:/user/hanakawa/htdocs/abc/check.php” is absolute. As a result, “not found error (404)” occurs. This is a typical novice fault about pathname.

The results of the case 1 and case 2 are shown at Fig.9. The tool can trace the HTTP request in the logs of the servers and the web application, and the router. Please see the log areas of the web applications in both the case 1 and the case 2 (See red circles in Fig.9). The log area of the web application in the case 1 is empty, in contrast, the log area of the web application in the case 2 is not empty. That is, in the case 1, the HTTP request did not arrive at the web application. In contrast, the HTTP request arrived at the web application in the case2.

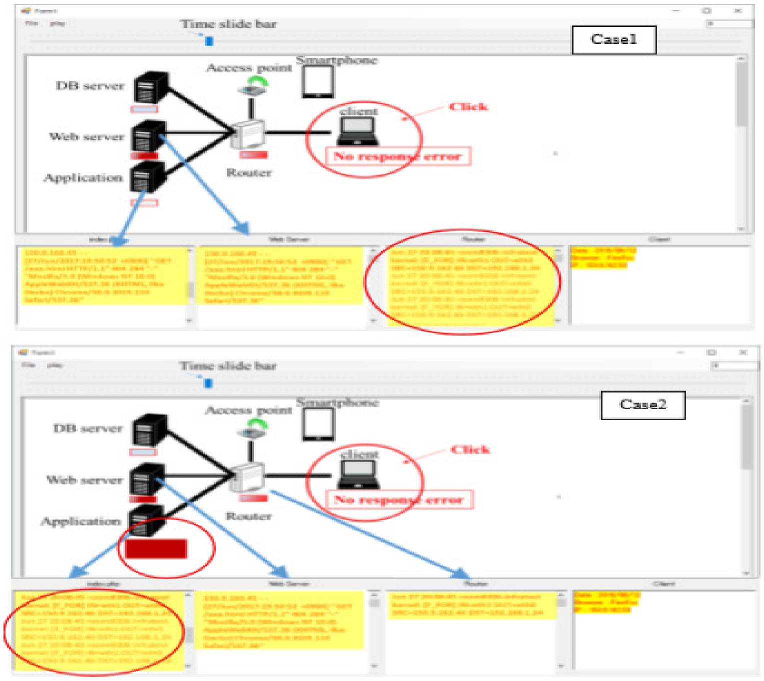


Fig.10 The tracing result in the no response error

The difference of the arrival to the web application is important when software engineers investigate program faults. In the case 1, they can know that the php program is not wrong. In the case 2, they can recognize higher possibility of the php program faults. In this way, the result of the tool helps software engineers divide software faults and infrastructure faults in web applications.

5.8. Evaluation Of The Tool In "No Response Error"

The system failure "no response error" occurred in the notebook computer.

1) **Case1:** problems in the firewall setting in the router

The firewall has a filtering function. The filtering setting is difficult without sufficient knowledge of network environment. Many mistakes of setting the filtering occurred in various computer system [19]. Therefore, it assumes that the filtering setting is wrong. The notebook computer receives "no response error".

An example of the wrong setting is a mistake of an iptables configuration file in a firewall. The forward and input commands were commented out in the iptables configuration file as follows,

```
#-A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
#-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
```

The comment-outs prevent from proper accesses to the web application. Therefore, "no response error" occurred on the notebook computer.

2) **Case2:** problems in the php programs

If the php program has an endless loop code, the php program cannot return the HTTP response. Of course, the endless loop code is wrong. Software engineers should find the endless loop code immediately. Because the php program cannot return the HTTP response, the notebook computer receives "no response error".

The results of the case 1 and case 2 are shown at Fig.10. The tool can trace the HTTP requests and the HTTP responses in the log files. In the case 1, the HTTP request from the notebook computer arrived at the web application. After that, the web application returned a HTTP response to the web server. The web server tried to return the HTTP response to the router. However, the firewall did not permit the forward the HTTP response to the notebook computer. Therefore, too many retries of the forward action of the HTTP response occurred. The log area of the router presents the too many retries of the forward (See a red circle of the case1 of Fig.10). Because the log area of the router is limited on the tool, all retries cannot are shown on the Fig.10. However, if you scroll the log area of the router, you can see the too many retries of the forward commands. A software engineer alone can understand that the router has any problems, and, the php program has no faults.

In the case 2, the router and the web server normally received the HTTP request from the notebook computer. However, the web server and the router did not receive the HTTP response. We focus on the log area of the php program. There are too many logs about the HTTP request and the HTTP response. Especially, the red rectangle of the web application is deep red, and bigger (See a circle of the case 2 of Fig.10). The red rectangle of the web application presents how heavy the web allocation's load is. That is, the php program had heavy load. The too many logs and the too heavy load of the web application suggest some faults in the php program. A software engineer can detect the endless loop program easily, in the same way, an infrastructure engineer can understand easily the high possibility of the web application software faults using the tool.

6. A Practice Of The Tool In A Real Computer System

6.1. Purpose

To evaluate the tool, the tool was applied to a real large-scale computer system. The summary of the target computer system presents at Table2. The target computer system starts working from April of 2017. However, several system failures occurred. The most frequent failure is "Authentication error" on a web application. The failure was investigated in the tool.

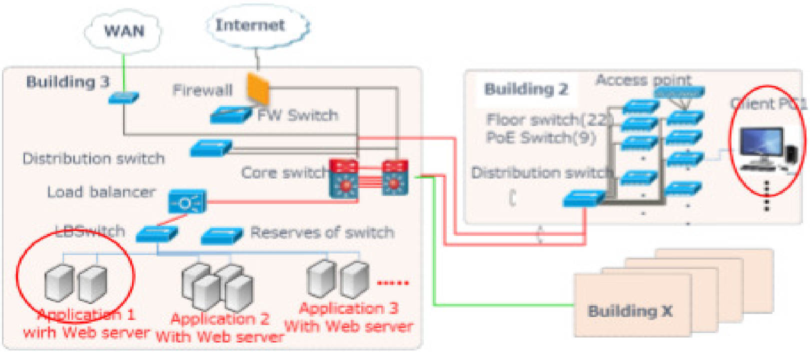
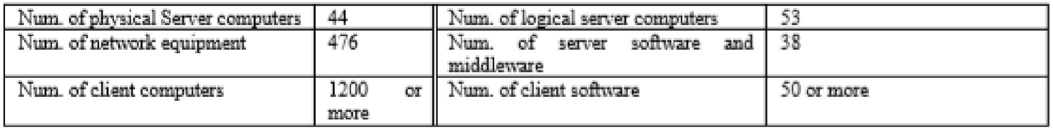


Fig.11 The computer environment at the failure occurrence

Unfortunately, the tool cannot apply the target computer system because the scale of the computer system is too large. The visualization of the computer system is difficult in the tool. Therefore, the images of this section are made manually to be like the tool screen. In addition, the computer system has already completed. Therefore, all logs in the tool were not recorded. Several logs of the servers, the network devices and web applications were missing.

Table Ii. Logs Of The Tral Computer System



Num. of physical Server computers	44	Num. of logical server computers	53
Num. of network equipment	476	Num. of server software and middleware	38
Num. of client computers	1200 or more	Num. of client software	50 or more

6.2. A Target Failure

Fig 11 shows the computer system environment of the system failure. The failure occurred at 14:37 on June 16, 2017. The failure was an authentication error of the “Application1” on the “Client PC1” (see Fig.11). The “Application1” runs on two server computers connected with the load balancer. Therefore, if the first server computer for “Application1” has heavy load, the second server computer for “Application1” runs instead of the first computer.

6.3. Replaying And Tracing Results

At first, we rewound the logs to around at14:37 on June 16, 2017. Next, we replayed the logs one second at a time. Fig.12 shows the tracing result of the computer system at 14:37:24 on June 16, 2017. The IP address of the “Client PC1” is “150:11:160:1”. The HTTP request from “150:11:160:1” at 14:37:24 moved to the core switch. Then, the request arrived at the “AP2-1” of “Application2” server computer. After that, the HTTP request arrived at the “AP1-1” and “AP1-2” of Application 1 server computers. The red filled circles in Fig.12 indicate where the HTTP request passed. That is, the accesses of the IP address “150:11:160:1” were recorded in the log files. The other computers and devices had no logs about the IP address.

In the Fig.12, we found strange phenomena on the “AP1-1” and “AP1-2”. The “AP1-1” and “AP1-2” are the same web application “Application1”. The reason of the two server computers is the load distribution. The load balancer connected with the “AP1-1” and the “AP1-2” distributes the load of the computers. Usually, a series of HTTP requests arrived at a server computer because of consistency of the series of the HTTP requests. The management of the series of requests is called “session”. However, the HTTP requests from the “Client PC1” at 14:37:24 were simultaneously reached into the two server computers. The phenomena of the reaching into two server computers at the same time indicated a possibility of some faults in the infrastructure or the software. We understood that the load distribution functions had any problems in the infrastructure or the web application software. Because the session of the series of the HTTP requests was broken, the “Authentication error” occurred on the “Client PC1”.

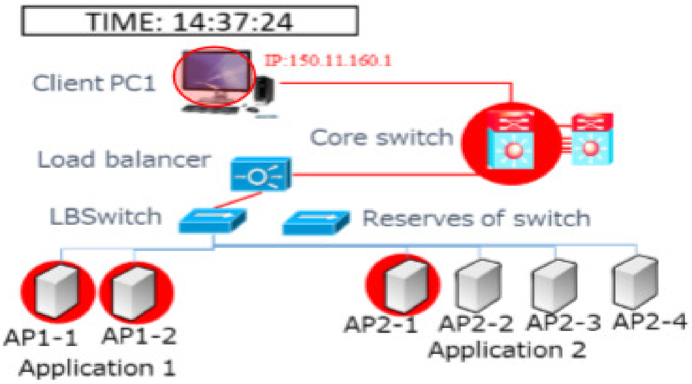


Fig.12 The replaying and tracing result

6.4. INVESTIGATING

We Investigated The Detail Of The Logs At 14:37:24 On The “AP1-1” And “AP1-2”. Unfortunately, The Load Balancer Log Files Were Not Recorded.

The first request to the “Application 1” from the “Client PC1” at 14:37:24 was “GET” command for “ssoLogin” because the user of the “Client PC1” has already logged-in the “Application 2”. Then, the “Application 2” uses the single sign-on function when the “Application 2” goes to the “Application 1”. Therefore, the session between the “Client PC1” and the “AP1-1” has been established. After that, the “AP1-1” received various requests from the “Client PC1”. The number of the requests to the “AP1-1” was 60. However, the “AP1-2” simultaneously received 12 requests from the “Client PC1”.

The all received requests to “AP2-2” were “GET” command for getting image files. Therefore, the logs of the “Application 1” were investigated more. The “Application 1” is a web application in Java language. Usually, to keep the session between the client computer and the server computer, an HTTP request header includes “session ID” using cookies. However, an HTTP request header without “session ID” sends to the server because of performance of the web application. That is, when the image files get from the server, the HTTP request without session ID sends. The reason is that the getting image files do not need consistency of the processing.

After that, the last access of the HTTP request from “Client PC1” at 14:37:24 was on the “AP12” for getting image files. Therefore, the session between the “AP1-1” and the “Client PC1” was broken at the last access. Therefore, the next HTTP request from “Client PC1” reached to “AP12” because the “AP1-1” had heavy load. Therefore, the next request was not able to be processed on the “AP1-2”. The “Client PC1” was not authorized to access “AP1-2” of the “Application 1”.
This case was complicated. The software had a problem of sending session ID at the getting image files, at the same time, the infrastructure had a problem with the load balancer setting. The software problem was that the HTTP request header did not include session ID for keeping the
Fig.12 The replaying and tracing result

session. The problem of the load balancer was that “Fast cash” setting was wrong. The fast cash is for downloading repeatable image files. However, paths for the fast cash function did not include the path of the image files of the “Application 1”. If either of them was doing the appropriate treatment, the failure would not have occurred. Moreover, the failure rarely occurred. Of course, if the “AP1-1” server’s load is a little, the failure will not occur. Software problems and infrastructure problems and network load are influenced each other. The detecting the faults in such situation is too difficult for software engineer alone.

The replaying and the tracing such as Fig.12 helps software engineers investigate faults in the cases of both software problems and infrastructure problems. Even if software engineers cannot completely find infrastructure faults, they will be able to distinguish software problems from infrastructure problems using the replaying and the tracing of the tool.

7. SUMMARY

We propose a tool for evaluating web applications. The feature of the tool is seamless evaluation between software and infrastructure. The logs recording actions are set to network devices, servers, and web applications in the system development phase. After that, if system failures occur at the system operation phase, the replaying and the tracing tool helps software engineers detect faults. Even if the failure occurs caused by the complicated faults that are influenced both software and infrastructure, software engineers alone will be able to distinguish between software faults and infrastructure faults using the replaying and the tracing of the tool.

In future, the tool of the replaying and the tracing will be improved to adapt a large-scale computer system. After that, we will evaluate in several real system faults of the large-scale computer system.

ACKNOWLEDGEMENT

This work was partially supported by JSPS KAKENHI Grant Number JP26330093. We thank Fujitsu Marketing Limited and Hannan university for providing non-disclose and valuable system design documents, log data.

REFERENCES

[1] Noriko Hanakawa and Masaki Obana, “Process metrics for system quality with specifications' shifts from a bid phase to an operation phase”, the 30th International Conference on Software Engineering & Knowledge Engineering (SEKE2018), Jul, 2018 (appearing).

[2] D. Serpanos, and T. Wolf, Architecture of Network Systems. Morgan Kaufmann Publishers Inc., San Francisco, CA, 2011.

[3] Noriko Hanakawa, Masaki Obana, ”A Proposal for Resolving the Computer System Failures with Infrastructure Problems and Software Problems”, 2018 International Conference on Software Engineering and Information Management (ICSIM 2018), Jan. 2018, pp. 45-50.

[4] Masaki Obana, Noriko Hanakawa, ”System Metrics for Infrastructure Based on Network Design Sheets”, IPSJ Journal, vol.57, no 10, pp. 2272—2283, 2016 (In Japanese).

[5] A. Miranskyy, A. Hamou-Lhadj, E. Cialini and A. Larsson, "Operational-Log Analysis for Big Data Systems: Challenges and Solutions," IEEE Software, vol. 33, no. 2, pp.52-59, Mar.-Apr. 2016.

[6] S. Elbaum, G. Rothermel, S. Karre, and M. Fisher II., “Leveraging User-Session Data to Support Web Application Testing,” IEEE Transaction Software Engineering, vol.31, no.3, pp.187-202, March 2005.

[7] J. Wang, W. Dou, C. Gao, and J. Wei, “JSTrace: Fast Reproducing Web Application Errors,” Journal of Systems and Software, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2017.06.038.

[8] R. Murillo-Garcia and G. L. Miotto, "A Log Service Package for the ATLAS TDAQ/DCS Group," in IEEE Transactions on Nuclear Science, vol. 54, no. 1, pp. 202-207, Feb. 2007.

[9] A. Dabir, and A. Matrawy, “Bottleneck Analysis of Traffic Monitoring using Wireshark,” 2007 In Proceedings of the Innovations in Information Technologies (IIT), Dubai, 2007, pp. 158-162.

[10] V. Y. Hnatyshin, and A. F. Lobo, “Undergraduate data communications and networking projects using opnet and wireshark software,” In Proceedings of the 39th SIGCSE technical symposium on Computer science education (SIGCSE '08). ACM, New York, pp.241-245, 2008.

[11] https://www.wireshark.org/download.html

[12] F. Willnecker, A. Brunnert, W. Gottesheim, and H. Krcmar. “Using Dynatrace Monitoring Data for Generating Performance Models of Java EE Applications,” In Proceedings of the 6th ACM/SPEC Intern,tional Conference on Performance Engineering (ICPE '15). ACM, New York, pp.103-104, 2015.

[13] F. Willnecker, M. Dlugi, A. Brunnert, S. Spinner, S. Kounev, W. Gottesheim, H. A. Krcmar, M. Beltran, W. Knottenbelt, and J. Bradley, “Comparing the Accuracy of Resource Demand Measurement and Estimation Techniques,” In Proceeding of the Computer Performance Engineering 12th European Workshop, Madrid, Spain, pp.115-129, 2015.

[14] S. Guckenheimer and J. J. Perez, Software Engineering with Microsoft Visual Studio Team System (Microsoft .NET Development Series), Addison-Wesley Professional, 2006.

[15] M. Audris, S. Forrest (Editor), S. Janice (Editor), S. Dag I. K.(Editor), Chapter 7 Missing Data in Software Engineering, Guide to Advanced Empirical Software Engineering, pp.185-200, Springer London, 2008.

[16] https://httpd.apache.org/

[17] https://www.mysql.com/

[18] https://www.centos.org/docs/5/html/Deployment_Guide-en-US/ch-iptables.html

[19] T. Abbes, A. Bouhoula, and M. Rusinowitch, “An inference system for detecting firewall filtering rules anomalies,” In Proceedings of the 2008 ACM symposium on Applied computing, Brazil, pp.2122-2128, 2008.