Copyright ⓒ 2021 KSII

A Study of Resource Utilization Improvement on Cloud Testing Platform

Jong-Yih Kuo*, Hui-Chi Lin and Chien-Hung Liu
Department of Computer Science and Information Engineering,
National Taipei University of Technology
Taipei, 106 Taiwan
[e-mail: {jykuo, huchlin, cliu}@ntut.edu.tw]
*Corresponding author: Jong-Yih Kuo

Received October 2, 2020; revised December 5, 2020; revised May 4, 2021; accepted May 24, 2021; published July 31, 2021

Abstract

This paper developed the software testing factory-cloud testing platform (STF-CTP) to address the software compatible issues in various smart devices. Software developers who only require uploading the application under test (AUT) and test script can test plenty of smart devices in STF-CTP. The challenge for the cloud test platform is how to optimize the resource and increase the performance in the limited resource. This paper proposed a new scheduling mechanism and a new process of the system operation which is based on the OpenStack platform. We decrease about 40% memory usage of OpenStack server, increase 3% to 10% Android device usage of STF-CTP, enhance about 80% test job throughput and reduces about 40% test job average waiting time.

Keywords: Android application testing, cloud-base testing, cloud resource management.

http://doi.org/10.3837/tiis.2021.07.008 **ISSN : 1976-7277**

1. Introduction

With the growing popularity of smart mobile devices, people are more and more dependent on the App developed on the device. Therefore, users experience has been getting more and more important in recent years [1] [2]. Users concern about the software compatibility on the various smart devices rather than the hardware specification. Developers are aware of the importance to the testing the App. However, testing on a single device is not enough. In recent years, the specifications of mobile devices developed by different vendors are not the same. Not only different OSs but also different sizes of devices may cause applications to make errors on different devices. Therefore, compatibility testing is very important before the software is shipped. The concerns of software compatibility are a major topic in smart device manufacturers and thousands of hundreds of software developers as well.

The software factory laboratory at National Taipei University of Technology has developed an integrated cloud testing platform, STF-CTP [3]. STF-CTP mainly focuses on the unit testing and automated acceptance testing with Android devices. It can solve the compatible issue on Android platform, which saves developers' time and money. STF-CTP enables App developers or testers to upload apps and test scripts for testing. After STF-CTP receives the test, it will dispatch the work to integrated station (IS) located in OpenStack virtual machine. [4] [5] [6] The IS connect to the specified device through Wi-Fi. After finish the test, Android device will be released and return the test result. At last, developers can access the test report and performance data via STF-CTP.

As the need of STF-CTP increases day by day, testers may want to upload a large number of test jobs to execute tests. In order to execute many test jobs at one time, STF-CTP needs to launch a lot of virtual machine to dispatch jobs to different IS. Besides, each testing device can only execute one test at a time; so, only when IS finish converting testing video record, it will release the device to let device get another testing job, which cost a lot of waiting time. After whole jobs are finished, IS will not close it selves, which will waste resource of OpenStack server. Therefore, how to use and allocate resources of devices and OpenStack server become an important issue.

Ala'a Al-Shaikh et al. [7] propose a utilization schedule algorithm to filter all requests that do not meet the optimal resource through layer-by-layer of sorting and puts the matching requests into the schedule, which eventually gets the best use of the utilization schedule. Bo Xu et al. [8] group virtual machines that are related to a certain service into a cluster and propose a segmentation method that can reduce the overall bandwidth requirement after deployment. How to use and allocate resources becomes an important issue. This study will explore how to maximize the utilization of resources, thereby improving the consumption of resources to achieve the same resources. The proposed architecture of this paper includes how to maximize the utilization of resources, how to improve the consumption of resources, how to improve the quality of STF-CTP production, and how to reduce average waiting time to the overall test. The remainder of this paper is organized as follows. Section 2 presents related work and method for finding the solution. In Section 3, we present an effective architecture to change the quantity of Integration Station (IS) in each OpenStack virtual machine and manage IS dynamical which can improve the resource utilization of STF-CTP. In addition, we propose a method to improve the utilization of testing devices that can reduce the execution time and average waiting time on the overall test. In Section 4, we conduct three experiments including the utilization of OpenStack Server, the utilization of Android device, and the performance of proposed STF-CTP architecture to confirm the effectiveness of the proposed solution. In Section 5, we summarize the conclusions and the contributions of this paper.

2. Related Work

This section introduces the background knowledge and research content related to this paper, including the choice of cloud-computing platform, the introduction and application of STF-CTP, how to make maximum use of resource utilization in cloud service, the algorithm of scheduling and deploy virtual machine and how to minimize the makespan of tasks.

2.1 OpenStack

Rakesh Kumar et al. [9] conducted a comparative study of architecture, operating system support, virtual machine migration and image management for several cloud-computing platforms such as Eucalyptus, OpenStack, CloudStack, and OpenNebula. They conclude that OpenStack is the most suitable solution for the cloud computing.

OpenStack is a free and open-source software platform for cloud computing and easy to build and manage public and private clouds. It allowed users deploy virtual machines and other instances that handle different tasks for managing a cloud environment. The STF-CTP is developed under OpenStack.

2.2 STF-CTP

The STF-CTP, software testing factory cloud testing platform, is a cloud testing service platform, developed by the National Taipei University of Technology software testing factory, which is based on OpenStack cloud computing services and Web UI. It mainly focuses on the unit testing and automated acceptance testing with Android devices. Acceptance testing supports different Android automation tools, including Monkey [10], MonkeyTalk [11], Robotium [12], UiAutomator [13] and Robot Framework [14]. In the STF-CTP environment, test jobs will be dispatched by Rabbit Message Queue [15]. Integration station (IS) will receive the job in rabbit message queue. The STF-CTP is required to execute many test jobs. It was developed on OpenStack to build virtual machines, and manage IS. IS connects Android device to execute test via Android Debug Bridge (ADB) [16]. After finish the test, Android device will be released and return the test result. At last, developers can access the test report and performance data via STF-CTP.

2.3 A Hierarchical Scheduling Strategy

A hierarchical scheduling strategy [17] addresses the problem of composition service scheduling and resource allocation in the cloud, especially the data-intensive and compute-intensive service. The implementation of each step regarded as the smallest unit and resource allocation depends on the job completion rate which is based on the work as a weight to reduce the work time to complete. Observing STF-CTP, we took the similar approach, but the scenario of test application test is hard to implement like the data-intensive and compute-intensive service. The software must be entirely executed on the target device to ensure the compatibility of the operating version and the hardware difference.

2.4 Resource Utilization in Cloud Computing as an Optimization Problem

Ala'a Al-Shaikh et al. [7] combined 0/1 Knapsack Problem [18] and Activity-Selection Problem [19] to propose a way to maximize resource scheduling. The paper mentioned that in the case of a given resource, hundreds of requests accumulate over time to use that resource. Each request has a profit index. For a given resource regarding profit obtained by utilizing that resource and the number of time slices during which the resource will be utilized, they design

an algorithm. The algorithm filters all requests that do not meet the optimal resource through layer-by-layer of sorting and puts the matching requests into the schedule, which eventually gets the best use of the utilization schedule. Then obtain the maximum profit of that resource.

The shortcomings of the algorithm are that the use of 0/1 Knapsack Problem's picks up or not pick up the request problem. The resource does not accept all requests and the requests that are not put into the schedule will be released and will not be used.

2.5 Deployment Method of VM Cluster Based on Graph Theory

The deployment of the virtual machine is the core problem in the cloud computing. The efficiency of the single virtual machine is too low; the overall bandwidth after deployment is high. Therefore, it will be more efficient to group virtual machines that are related to a certain service into a cluster and configure physical hosts in the form of a cluster.

Bo Xu et al. [8] use the energy minimization model to segment virtual machines into different clusters. Then, Bo Xu et al. change the deployment of virtual machine cluster by using max-flow min-cut problem that is based on graph cut theory. After that, the structure assumes the shape of a network. When modeling an energy minimization network, add a source point and a sink point to the network. This virtual machine segmentation method can reduce the overall bandwidth requirement after deployment.

2.6 An Optimized Task Scheduling Algorithm in Cloud Computing

Scheduling of job and maintaining load is a main issue in cloud environment by considering parameters such as throughput, resource utilization, cost, computational time, priority, performance, bandwidth, resource availability. In order to provide better quality of service (QoS), Shubham Mittal et al. [20] , Sharma et al. [21] , and Cui et al.[22] implement an optimized task schedule algorithm to reduce the makespan by dispatching tasks into different resources. This algorithm adapts the advantage of Min-Min, Max-Min and RASA algorithms to perform various check for finding an optimized task which can lead to minimum makespan. Their optimized task schedule algorithm first compute total execution time for task on each resources and compare each resources execution time for tasks to decide which task will be dispatched to which resource. Because this algorithm aims to reduce the makespan of all task, all the task may will be dispatched to the fastest resource when other resources handle the execute time of one task is greater than the total execute time of task in the fastest resource. This situation may did not make good use of resource.

2.7 Self-Organizing Map

Scheduling jobs to minimize the completion time of tasks is important, as it can increase the utilization, productivity, or profit of a cloud. In order to minimize the Job Completion Time (JCT), Li et al. [23], determine task placement plan and resource allocation plan for jobs and then formulate the problem of scheduling a single job as a Non-linear Mixed Integer Programming problem. They focus on the problem of scheduling embarrassingly parallel jobs composed of a set of independent tasks and consider energy consumption during scheduling. Because each task should be placed on only one server, they define resource availability constraint and energy consumption constraint to limit the total amount of computing resource that can be allocated to that job on a server and to limit the total amount of energy that can be consumed by a job.

2.8 Comparisons of Related Approach

In this section, three approaches, Manikandan et al. [24], Pan et al. [25], and Veerendra et al. [26], are compared with our proposed approach are shown in **Table 1**.

Table 1. The comparison experiment architecture

Approach	Purpose	Method	Results
MCFCMA [24]	To reduces load balancing in Cloud Quantum Computation.	1. Modified canopy fuzzy c-means (MCFCM) clustering algorithm to solve the issues of overloading and lag in load balancing. 2.Particle swarm optimization algorithm is uses to facilitate the optimal selection of virtual machines.	To reduce load balance, and improve parallel execution of tasks by measuring the memory, execution time, and cost.
ARIMA [25]	To exploiting real-time trends of performance changes of cloud infrastruc-tures and generate dynamic workflow scheduling plans.	1.ARIMA-time-series prediction model. 2.Critical-path-duration-estimation-based VM selection.	To reduce monetary cost while following constraints of Service-Level-Agreement.
AFO[26]	To improves speed and resources by adopting hybrid technology.	1.Balance workload and enhance tasks using certain Time period of execution, Response Time of Resources, Energy Utilization criterion. 2.Adaptive Fruit Fly Optimization technique improves comprehensive process flow, Resource Response time and optimizes adequate time.	To Scale down energy consumption in the cloud strategy.
STF-CTP	To improve the utilization of testing devices	1.The monitoring and dynamic resource scheduling method which is based on FIFOj. 2.Dynamic IS management.	To improve the resource utilization that can reduce the execution time and average waiting time on overall test.

In the study [24], the research is through the use of independent tasks in cloud computing to allocate resources through the summary of the improved canopy fuzzy c-means algorithm (MCFCMA). In order to assign tasks to their corresponding resources, a particle swarm-based optimization algorithm (PSO) is used. The proposed method can independent task selected based on load feed-back cluster the requested task using MCFCMA and schedule a task to each virtual machine. The proposed system overcomes issues in load balancing and load scheduling; this can be proved by its precision and privacy calculation.

In the study [25], the researches address the performance-variation-aware workflow scheduling problem by leveraging a time-series-based prediction model and a Critical-Path-Duration-Estimation based (CPDE) VM Selection strategy. The proposed method can exploit real-time trends of performance changes of cloud infrastructures and generate dynamic workflow scheduling plans. This study performs extensive experimental case analysis over real-world third-party commercial clouds and show the effectiveness of the proposed method.

In the work [26], the research is based on Adaptive Fruit Fly Optimization (AFO) technique called the metaphorical approach to optimize resources in cloud computing. Balance workload and enhance results for all mentioned tasks using the certain Time period of execution, Response Time of Resources (RTR), Energy Utilization Criterion Adaptive Fruit Fly Optimization technique improves comprehensive process flow, Resource Response time and optimizes adequate time. The proposed method can improve speed and resources by adopting hybrid technology. The implemented model works entirely by an increase in utilization thereby lower costs. Energy-intensive AFO-CS algorithms improve overall process flow and reduce the total processing time and RTR.

By comparing with other cloud computing methods shown in **Table 1**, it can be seen that our proposed approach is better than the three approaches for improving cloud computing resource utilization.

3. The Optimization of STF-CTP Resource Management

This section describes the system architecture and workflow of STP-CTP, and analyzes the utilization of CPU resource. Based on the analysis of the current process, we present a method to optimize the resource of STF-CTP.

3.1 STF-CTP Operation Procedure and Architecture

STF-CTP bases on OpenStack to manage virtual machines and the rabbit message queue uses as a communication bridge between the web server and virtual machines. **Fig. 1** shows STF-CTP operation model and procedure. First, developers will upload an application and test scripts. When the web server receives test job, it will dispatch the job to rabbit message queue. Integration station (IS) receives the job in rabbit message queue. IS will connects devices to execute test via Android Debug Bridge (ADB). When finishing the test, IS will release devices. Test result will transfer to rabbit message queue. Developers can review the test result through the web server.

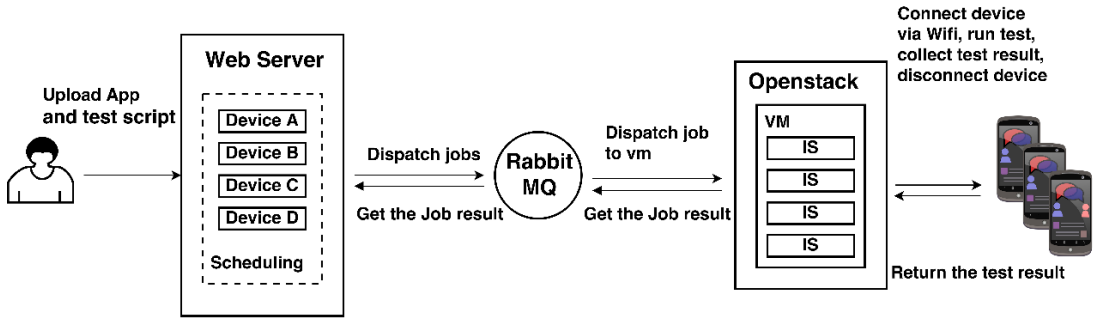

Fig. 1. STF-CTP operation model

Job scheduler of STF-CTP was followed Kuo et al. [27][28]. The monitoring and dynamic resource scheduling method which is based on FIFO, except the current required resource was occupied by another job. The next job will be executed earlier than previous one. Also with reference to Kuo et al. [28], proposed the STF-CTP resource monitoring and management service designed and applied enabled direct OpenStack operation and management. In addition, the automated VM monitoring mechanism was incorporated in the STF-CTP, which provided the STF-CTP with the operation statuses of all VM-related resources.

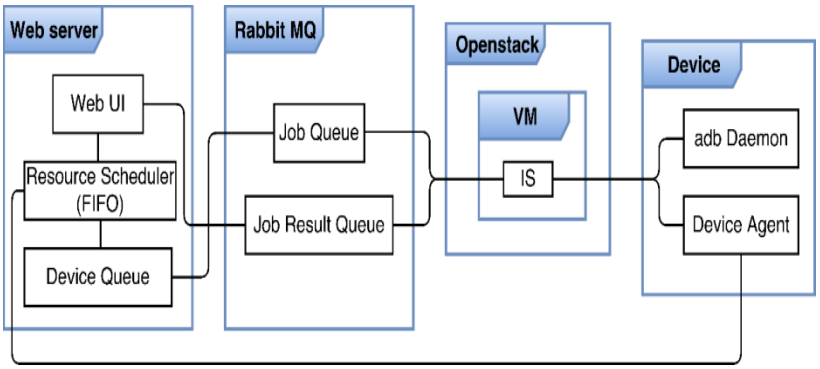

Fig. 2. STF-CTP architecture

STF-CTP consists of the web server, rabbit message queue, OpenStack, and smart devices. Web server is responsible for the test job scheduling. Rabbit message queue is used for inter-process communication between the Web server and IS. OpenStack manages virtual machines. Every virtual machine has its own IS. The smart devices execute the test and report the result to IS. Each component has its elements, as shown in **Fig. 2**. The details are described below.

1. Web UI: It's a user interface for developers.
2. Device Queue: The job is assigned to the device queue by resource scheduler depends on the usage of the physical smart device.
3. Resource Scheduler: The scheduler updates the latest usage of devices, scheduling the job in FIFO method.
4. Job Queue: The job will be dispatched to the device queue when the smart device is available.
5. Job Result Queue: This queue stores the test result for web server access.
6. IS: IS checks the job queue, connecting the devices to execute the job through the ADB. The result will be reported to job result queue by IS as well.
7. ADB Daemon: It is a program installed on the smart device. IS will send the ADB commands to ADB Daemon and ADB Daemon will execute the commands.
8. Device Agent: It is a monitor program that propose by Kuo et al. [28] that responsible for the preparation of test jobs and monitors the network flow of the Apps during the test.

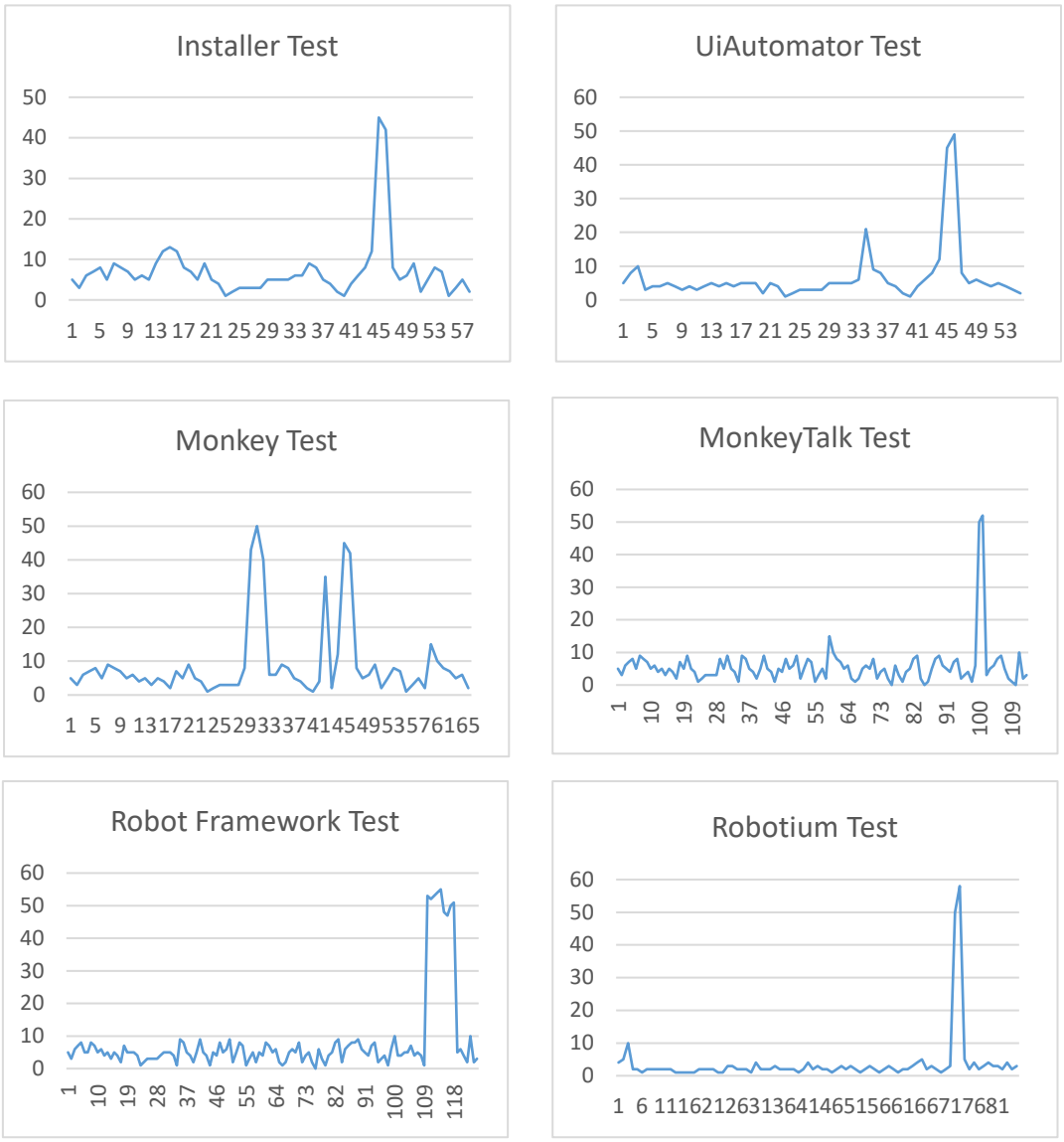

Fig. 3. CPU utilization in different test job

3.2 Analysis of STF-CTP CPU Utilization

Table 2. Virtual machine specification

Machine	Virtual Machine
OS	Ubuntu 12.04 64-bits
CPU	2 Cores
Memory	2 GB
Disk	20 GB

Table 3. The purpose of test job

Test job name	Purpose
Installer test	Ensures that the App can be installed under different conditions, for example, for first-time installations, upgrades, complete or custom installations.
Robot Framework test	Robot Framework is automation test framework and written in Python. It supports keyword-driven and mainly used for regression testing and acceptance test-driven development
Monkey test	Monkey is a command-line tool in Android that can run on a real device. It sends user event streams such as key presses, touch screen inputs, gesture inputs to the system to conduct stress test and test the stability and robustness of Apps.
UiAutomator test	UiAutomator is released after Android 4.1. It is used to do UI testing. For example, the login interface, enter the wrong username and password, then click the login button to see if it can log in or whether there is error message.
MonkeyTalk test	MonkeyTalk's platform is mainly composed of 3 parts: MonkeyTalk IDE, MonkeyTalk Agent and MonkeyTalk Scripts. 1. MonkeyTalk IDE - A system application for recording, playing and developing test programs. 2. MonkeyTalk Agent - The library needs to be included in the "app to be tested" to drive the test. 3. MonkeyTalk Script - Understandable and maintainable test program.
Robotium test	Robotium is a black-box testing. Developers only need APK rather than source code. With the help of Robotium, developers can write functional tests and acceptance tests across several activities.

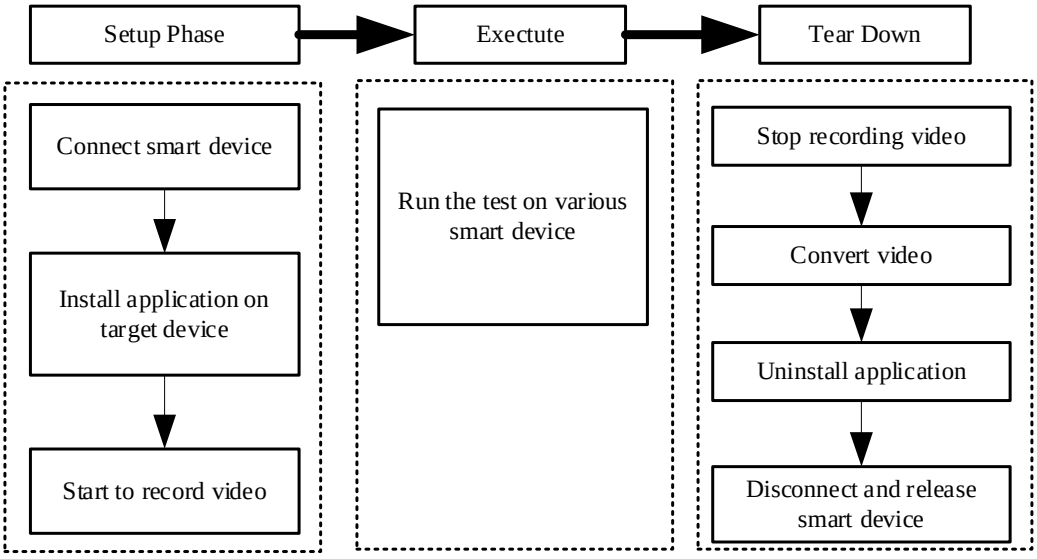

Fig. 4. Testing procedure on device

The CPU is the major resource in the STF-CTP. The specification of the virtual machine show in **Table 2**. In **Fig. 3**, X-axis is execution time, and Y-axis is the utilization of CPU. We observed that the peak of CPU usage is about 50% in six different tests, and it happens on the final stage of the entire test. **Table 3** shows the purpose of six different test jobs. The application we use to run the test is OnMyWay that was developed by STF-CTP team. We design four different scripts except Installer test and Monkey test. It is because Installer test is to test whether application can be successfully install or not and Monkey test is to simulate the

user touch the screen, pressing keyboard and other operations, mainly aimed at the robustness and stability of the application. The rest of the CPU usage remains low. The reason is that the test is executed on the device and CPU resource is not required in this phase. After that, the test details will be converted to the test video that consumes the CPU resource at the end of the test.

Fig. 4 shows three steps for executing the test job on the smart device. The details are described below.

1. Set up: Connect and wake up the smart device through Wi-Fi. Install and launch the Apps according to the test type. Launching the recording program records the test video while executing the test.
2. Execute test: Execute the test according to the test type. Some tests are executed by the smart device, others executed by PC terminal.
3. Tear down: Terminate the recorded program, and get the test video from the smart device to IS. After converting the test video, it will uninstall the Apps and release the smart device.

3.3 Proposed Method on STF-CTP

This section describes the procedures of our proposed method on STF-CTP.

3.3.1 Test Environment Setup

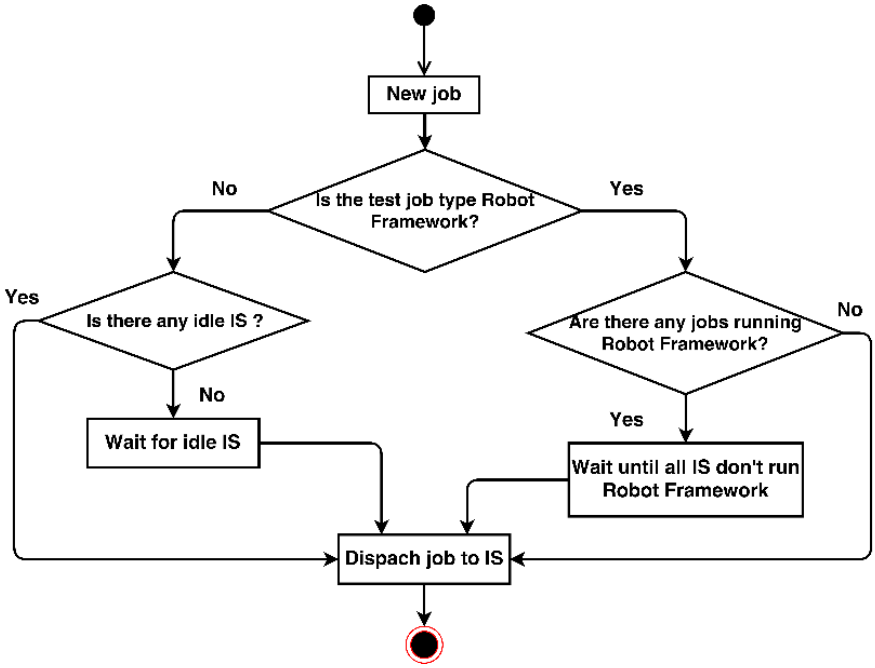

Fig. 5. Scheduler decision tree of Robot Framework job

A virtual machine has more than one IS. In other words, a virtual machine can execute multiple tests at the same time. It also means that each type of test has its method. Unfortunately, one virtual machine only can execute one Robot Framework test. Robot Framework is not supported parallel execution, but it can execute with another test type at same time. **Fig. 5** illustrates how the scheduler dispatches the Robot Framework job.

Table 4. Modified method for parallel execution

Test type	Modification method
Installer, Monkey, UiAutomator, Robotium	It requires the "-s" variable to decide which Android device to execute the test. "-s": IP or a serial number of Android device.
MonkeyTalk	MonkeyTalk uses ant script to execute the test, requires four variables to decide which Android device to execute the test. "-port": Android device connecting port "-adbLocalPort": ADB local port for connection of ADB. "-adbRemotePort": ADB remote port for connection ADB. "-adbSerial": IP or a serial number of Android device.
Robot Framework	It requires the "ANDROID_SERIAL" environment variable to decide which Android device to execute the test. "- ANDROID_SERIAL ": IP or a serial number of Android device.

In order to make all the test types can be executed in parallel; each test type will be modified as shown in **Table 4**. Robot Framework test type is modified because it does not support parallel execution. Therefore, the modification method of Robot Framework makes it execute with tests of other test types.

3.3.2 Effect of Quantity of IS

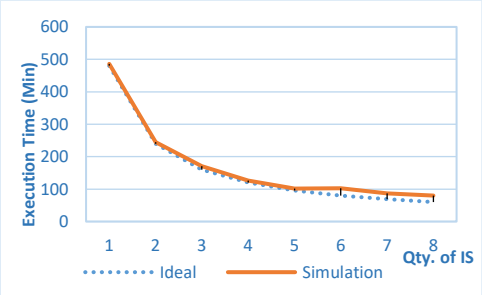

Fig. 6. Execution time under different quantity of IS

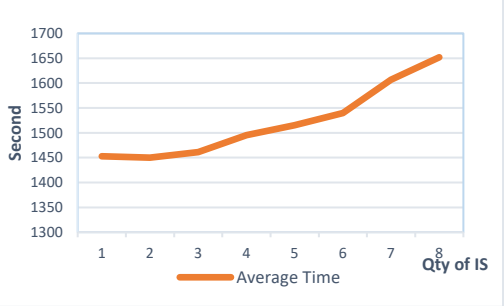

Fig. 7. Average waiting time under different of quantity IS

Regarding STF-CTP architecture, the quantity of IS will have different test result. This paper takes monkey test for example because it consumes the highest CPU average utilization among all tests. We assumed that there are 20 money jobs in the queue, waiting for executing with different quantity of IS. Because of the limitation of our testing environment, it supported to create eight IS at the same time.

It can be observed that the execution time cannot have obvious improvement when the quantity of IS more than four (see **Fig. 6**). As shown in **Fig. 7**, the average waiting time

increased with an increase of the quantity of IS. Under this circumstance, we can conclude the quantity of IS will be an important factor of the execution time and average time.

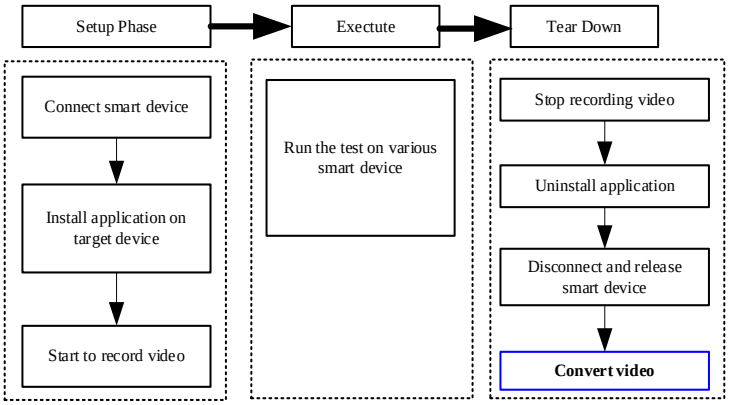

Fig. 8. Revised tear down procedure

3.3.3 Improve the Utilization of Device

Originally, the device resource will be released after converting the video which makes the device occupied and cannot be released until finish converting video. If the device is occupied, it cannot receive the new job, which will make the test job's average waiting time longer. Hence, we proposed a new tear down the procedure to release the device resource earlier before converting videos. The released device can be assigned or executed by other IS job regardless of converting video. **Fig. 8** depicts the revised tear down procedure.

3.3.4 Dynamic IS management

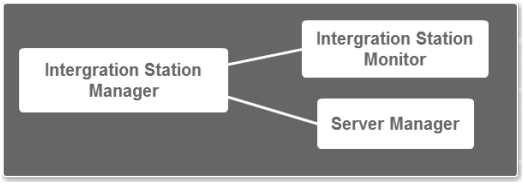

Fig. 9. IS management model

To optimize the quantity of IS in OpenStack servers, we propose a model to achieve IS dynamic, as shown in **Fig. 9.** Each component is described below:

1. IS Manager: Get the status of OpenStack server, virtual machine and IS by Server Manager and IS Monitor. And, manage the quantity of IS and a virtual machine as well.
2. IS Monitor: Monitor the status of IS and communicate with IS.
3. Server Manager: Monitor the status of the server and create or delete virtual machine by OpenStack API.

While IS is dynamic, the status of IS become three types from two types. Each component is described below:

1. Idle: IS is available to execute the new job.
2. Busy: IS is busy and cannot execute a new job until job finish.
3. Shutdown virtual machine: When the closed IS is the only IS on the virtual machine, it needs to be closed with the virtual machine as well. Because it takes a short time to close the virtual machine, this state is added to let STF-CTP know the state of this IS which will avoid assigning the job to this IS.

3.3.5 Mechanism of Dynamic IS

The IS dynamic mechanism having two-part, dynamic start IS and dynamic shutdown IS. The details are described below.

1. Dynamic start IS: When STF-CTP has a job to do, and the total IS quantity (IS_{total}) is less than the maximum quantity of IS (IS_{max}), and the quantity of job (Job) is more than the quantity of idle IS (IS_{idle}), STF-CTP will start the quantity of IS_{start}, as shown in equation (1). And the CPU and memory utilization of OpenStack server decide the IS_{max}. If the utilization is over 90%, STF-CTP will not start new IS.

$$IS_{start} = \begin{cases} Job - IS_{idle}, & If\ Job > IS_{idle}\ and\ IS_{total} < IS_{max} \\ 0, & otherwise \end{cases} \quad (1)$$

2. Dynamic shutdown IS: When IS is idle (IS_{idle_time}) for a while (timeout α), STF-CTP will shutdown the quantity of $IS_{shutdown}$. Timeout α is set by STF-CTP manager, as shown in formula (2). The manager can adjust Timeout α according to the test interval and the number of test jobs. If the Timeout α value is very big, it may cause unnecessarily memory usage of the server.

$$IS_{shutdown} = IS_{idle\_time} > \alpha \quad (2)$$

3.3.6 Proposed System Architecture

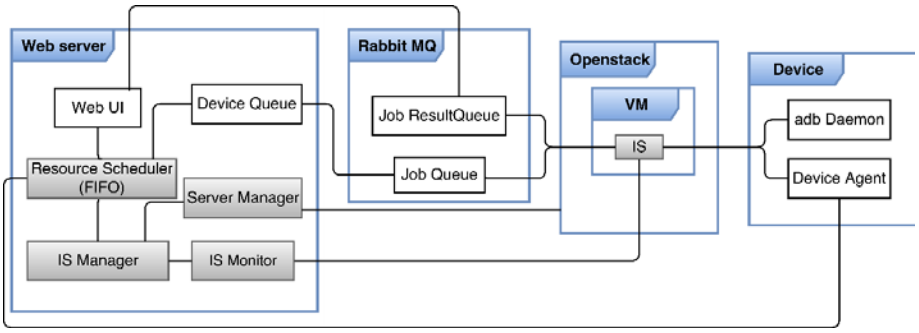

Fig. 10. Proposed system architecture

Adding and modifying five components on original STF-CTP architecture, details have shown in **Fig. 10**. Each component is described below:

1. Resource Scheduler: Add Robot Framework decision tree into scheduler which is still first in first out. If Robot Framework job cannot be executed, STF-CTP will let next job execute first.
2. IS: Receive web server notification and execute the job from Rabbit Message Queue. Create or terminate IS on the virtual machine by receiving the command from IS Manager. When IS itself is idle and over timeout α , it will be shut down and notify Server Manager to shut down the virtual machine as well if the IS is the last one on the virtual machine.
3. Server Manager: Monitor OpenStack server and deal with the problem about the virtual machine by OpenStack API.
4. IS Monitor: Monitor IS and communicate with IS.
5. IS Manager: Receive the status of OpenStack server, virtual machine and IS from Server Manager and IS Monitor. Manage the quantity of IS and virtual machine and decide the new IS create at which virtual machine and which IS execute the job.

By using IS Manager to dynamically add or delete IS and virtual machine, it can save a lot of waiting time and make good utilize the resource. IS Manager monitors IS status through IS Monitor, knowing if IS ready for work or a type of work test in progress. Assignment work is also transmitted through IS Monitor to the IS. When IS received instructions, it would add a new IS it their own virtual machine. When the entire virtual machine mounts the maximum load capacity IS, it is necessary to learn from the Server Manager whether the OpenStack server can add a new virtual machine. Server Manager will complete the addition or deletion of virtual machine through calling OpenStack API. We can add a new condition that only when OpenStack server uses no more than 90% of the CPU and Memory then OpenStack can add a new virtual machine.

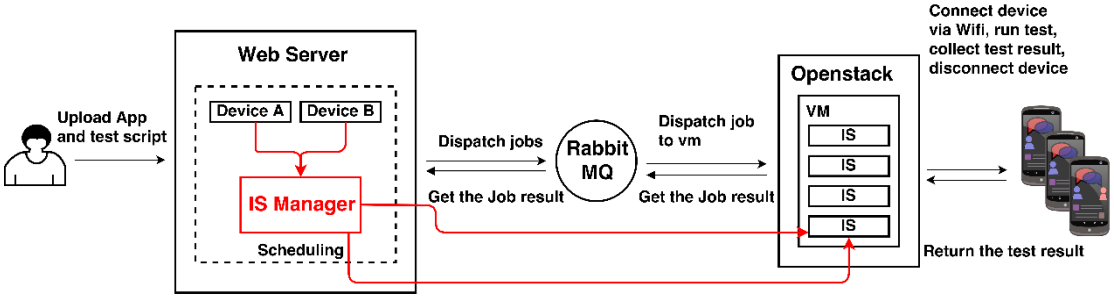

Fig. 11. Proposed system workflow

3.3.7 Proposed System Workflow

Fig. 11 illustrates the new workflow on STF-CTP, and details are described as below. The following procedure starts with the star mark is the new procedure.

1. An application under test and test scripts are uploaded by developers.
2. *IS Manager gets information by IS Monitor and Server Monitor and create new IS.
3. The job is assigned to Rabbit Message Queue by the web server.
4. *IS Manager notifying IS to get the test job from Rabbit Message Queue.
5. Integration Station (IS) receives job in Rabbit Message Queue.
6. IS connects the smart device to execute test via Android Debug Bridge (ADB). The smart device will be released when the test is finished.
7. The test result will be transferred to Rabbit Message Queue.
8. Developers can review the test result through the web server.
9. *IS will shut down itself when it's idle time over timeout.

4. Experimental Results

The three experiments were conducted to verify the different objectives of the system.

4.1 Experimental Environment

The experiments were conducted in a 100-Mbit network environment, in which the combination of 802.11-n and 802.11-ac wireless networks was used. A physical machine was employed as the CTP server, and five VMs that had been turned on were used as the test-running units. Ten smart devices with Android platform were used as the hardware devices for running tests. The environments are shown in **Table 5** and **Table 6**. The mobile phone brands which shown in **Table 7** included HTC, Sony, LG, Samsung, and OPPO, and the Android operating system version is from 4.1 to 4.4. **Table 8** lists the test application what we use to develop the test script and run test.

Table 5. CTP web server specifications

Machine	CTP server
OS	Windows 7 64-bits
CPU	Intel Core i7-4700 6M Cache 3.40GHz (4 Cores)
Memory	8 GB
Storage	128GB Solid State Disk
Disk	3 TB

Table 6. OpenStack server specifications

Machine	OpenStack Server
Version	OpenStack Kilo (April 2015)
OS	Ubuntu 12.10 64-bits
CPU	Intel Xeon E5-2630 15M Cache 2.30GHz (6 Cores) *2
Memory	116 GB

Table 7. The smart device specifications

Brand	Name	Android version	Quality
HTC	Butterfly S	4.3	1
HTC	Butterfly 2	4.3	1
Samsung	S3	4.1.2	1
Sony	Xperia Z2a	4.4	1
Sony	Xperia Z3	4.4.4	1
Sony	Xperia Z3 C	4.4.4	1
LG	G2	4.2.2	1
LG	G3	4.4.2	1
OPPO	N1 mini	4.3	1
OPPO	N1	4.2	1

Table 8. Test application

App name	Version	Use instructions
ezWeight	1.0	Weight monitoring recorder, capable of calculating and recording BMI.
Text Edit	1.5	A notebook that can add and delete files.
Calculator	1.1	Simple calculation function.
Bodha Converter	1.0	Conversion between each carry and ASCII Code.

4.2 Utilization of OpenStack Server

To compare the utilization of OpenStack server, **Table 9** shows the test scenario between the original architecture and proposed architecture. The first fifty tests will be executed and the coming fifty tests will be executed after five minutes on 10 Android devices. The original architecture uses 10 IS and proposed architecture IS are dynamic according to the usage and the test job amount.

Table 9. Test application

Architecture	Original architecture	Proposed architecture
Quantity of test job	50+50	50+50
Test job interval	5 minutes	5 minutes
Quantity of IS	10	dynamic
Quantity of smart device	10	10

According to the experimental test result in **Fig. 12** and **Fig. 13**, proposed architecture has higher CPU usage and save about 40% usage of memory capacity than original architecture in average. The main reason is each VM supports multiple IS in proposed architecture result in less VM needed in execution. In addition, the drop curve during 601s to 701s in proposed architecture is the idle VMs were closed, which also can save unnecessary memory usage.

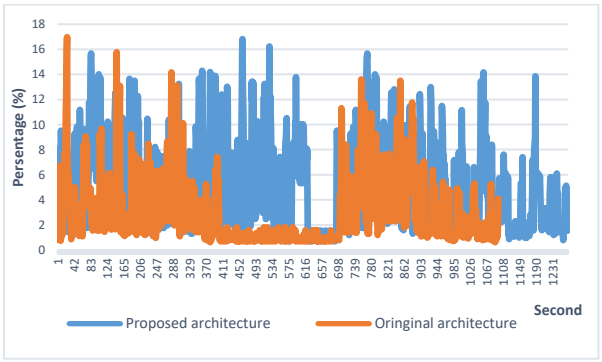

Fig. 12. CPU utilization of OpenStack server

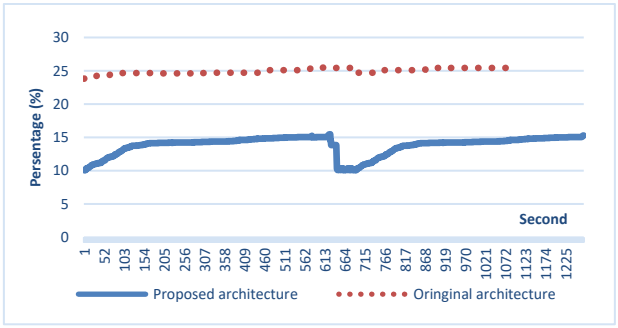

Fig. 13. Memory utilization of OpenStack server

4.3 Utilization of Android Device

This section will present the proposed architecture improvement about utilization of Android device. There two kinds of video on STF-CTP. One is video only and the other is a video with the subtitle, so the experiment executes two different tests. The test pattern is also shown in **Table 10**. The Android device specification is shown in **Table 11**.

Table 10. Utilization of Android device experiment specifications

Test type	Test execution time	Video type	Quality of test job
Monkey	24 minutes	Video	1
Robot Framework	37 minutes	Video and subtitle	1

Table 11. Android device specifications

Brand	Name	Android version
HTC	Butterfly 2	4.2.2

The test result is shown in **Fig. 14**. It can be observed that the usage of Android device in original architecture has a short idle time because Android device was not released until

converting the video finish. Oppositely, there is no idle Android device in switching test based on our proposed architecture. The main change is the Android device will be released before converting the video; therefore, the Android device will immediately be assigned to execute other test job so that proposed architecture's device will be occupied more than the original architecture's device which increases the utilization of Android device and reduces the execution time. To sum up, the utilization of Android device in proposed architecture is better than the original one.

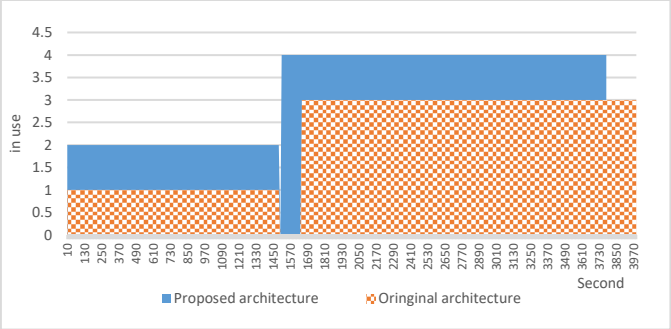

Fig. 14. Utilization of Android device

4.4 Performance of Proposed STF-CTP

Table 12. The performance of STF-CTP experiment specifications

Architecture	Original architecture	Proposed architecture
Quality of test jobs	60、120、240	60、120、240
Quality of virtual machine	5	Dynamic (maximum is 5)
Quality of Android device	10	10

The experiment shows the improvement of the proposed architecture by executing 60, 120, 240 tests. Detail is shown in **Table 12**.

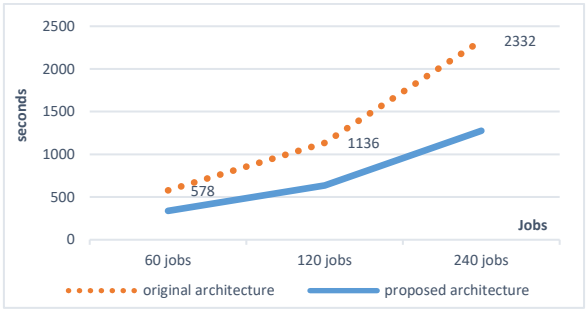

Fig. 15. Average Waiting Time of STF-CTP Test

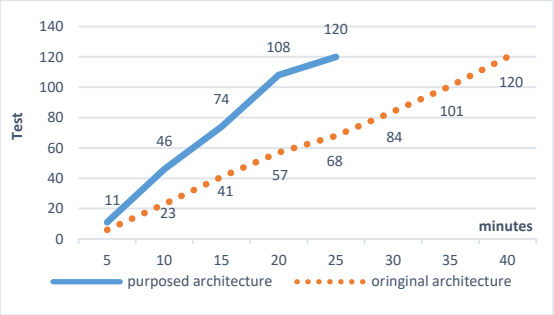

Fig. 16. Throughput of STF-CTP Test

The average waiting time of STF-CTP test is shown in **Fig. 15**. It can be observed that the proposed one is less than original one because the quantity of IS in proposed architecture is more than original architectures. Although the overhead in proposed architecture is to create new IS, it still takes less time than the original one.

The throughput test result of STF-CTP test is shown in **Fig. 16**. Multiple IS in one VM can execute more tests in parallel without waiting for other VM available. The two results show that the proposed architecture can reduce the average waiting time and enhance the throughput with the same OpenStack server, Android device and hardware resource.

After 2015, the scheduler technology has been transferred to the organization called Taiwan Testing and Certification Center (ETC) [29] where the technology is used and extended for commercial/business operations. The extension of the technology mainly focuses on the enhancement of system functionality to support security testing and the change of system architecture to improve the overall performance through parallelism by using additional servers. The core technology of the scheduler used in the framework of ETC has not been optimized or further improved. **Table 13** lists the comparison of the two frameworks.

Table 13. Comparison of two architectures

Approach	Purpose	Technologies
The proposed research	To improve the resource utilization that can reduce the execution time and average waiting time on the overall test.	Expand the FIFO-based monitoring, dynamic resource scheduling methods and dynamic IS management.
Taiwan Testing and Certification Center [29]	To support mobile application testing for commercial/business operations to improve bug discovery rate, and to lower the cost.	Extend the system architecture to enable parallel computing for improving the performance.

As can be seen from **Table 13**, this study has expanded the FIFO-based monitoring, dynamic resource scheduling methods and dynamic IS management. The Taiwan Testing and Certification Center was transformed into a commercial operation after the transfer of the technology bank. The work of focus is on how to use this technology to guide companies to efficiently test their mobile software application, and to create test cases based on test methods to lower the cost for commercial/business operations.

5. Conclusion

This paper proposes an effective architecture to change the quantity of IS in each OpenStack virtual machine and dynamically manage IS which can improve the resource utilization of

STF-CTP. In addition, we propose a scheduling method to improve the utilization of testing devices that can reduce the execution time and average waiting time on overall test. We decrease about 40% memory usage of OpenStack server, increase 3% to 10% Android device usage of STF-CTP, enhance about 80% test job throughput and reduces about 40% test job average waiting time.

In future, we hope that Robot Framework test job can be supported in parallel execution, and improve the test job average waiting time by predicting the execution time of each test job. It can reduce the test job average waiting time by scheduling the test job according to the completion of test job also let IS dynamic can support load balance on the virtual machine.

Acknowledgment

This research was supported by Ministry of Science and Technology, R.O.C. program MOST 106-2221-E-027-018-MY2.

References

[1] Yeong-Jun Kim, Jae-Wook Jeon, "Benchmarking Java application using JNI and native C application on Android," in *Proc. of The 12th International Conference on Control, Automation and Systems*, pp. 284-288, 2012.

[2] Lin Deng, Nariman Mirzaei, Paul Ammann, and Jeff Offutt, "Towards mutation analysis of Android Apps," in *Proc. of IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 1-10, 2015. Article (CrossRef Link).

[3] STF-CTP, "Cloud Testing," 2015. Available: http://www.openfoundry.org/of/projects/2193.

[4] Raiyani Kashyap, Sanjay Chaudhary, P. M. Jat, "Virtual machine migration for back-end mashup application deployed on OpenStack environment," in *Proc. of International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pp. 214-218, 2014. Article (CrossRef Link).

[5] Salami Suhas Sahasrabudhe, Shilpa S. Sonawani, "Comparing OpenStack and VMware," in *Proc. of International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, pp. 1-4, 2014. Article (CrossRef Link).

[6] Jose Teixeira, "Developing a Cloud Computing Platform for Big Data: The OpenStack Nova case," in *Proc. of International Conference on Big Data (Big Data)*, 2014. Article (CrossRef Link).

[7] A. Al-Shaikh, H. Khattab, A. Sharieh, and A. Sleit, "Resource Utilization in Cloud Computing as an Optimization Problem," *International Journal of Advanced Computer Science and Application (IJACSA)*, vol. 7, no. 6, 2016. Article (CrossRef Link).

[8] Z. Peng, W. Ke, M. Zhong, and A. M. Gates, "Deployment method of VM cluster based on graph theory for cloud resource management," *IET Communications.*, vol. 11, no. 5, pp. 622–627, 2017. Article (CrossRef Link).

[9] Rakesh Kumar, Neha Gupta, Shilpi Charu, Kanishk Jain, Sunil Kumar Jangir, "Open Source Solution for Cloud Computing Platform Using OpenStack," *International Journal of Computer Science and Mobile Computing*, vol. 3, no. 5, pp. 89-98, 2014. Article (CrossRef Link).

[10] Google, Inc., "Monkey," 2015. Available: http://developer.android.com/tools/help/monkey.html.

[11] Cloudmonkey LLC, "MonkeyTalk," 2015 Available: https://www.cloudmonkeymobile.com/monkeytalk.

[12] Chien-Hung Liu, Chien-Yu Lu, Shan-Jen Cheng, Koan-Yuh Chang, Yung-Chia Hsiao, xeng-Ming Chu, "Capture-Replay Testing for Android Applications," in *Proc. of International Symposium on Computer, Consumer, and Control (IS3C)*, pp. 1129-1132. 2014. Article (CrossRef Link).

[13] Google, Inc., "UiAutomator," 2015. Available: http://developer.android.com/tools/help/uiautomator.

[14] Jian-Ping, Liu, Juan-Juan, Liu, Dong-Long Wang, "Application Analysis of Automated Testing Framework Based on Robot," in *Proc. of Third International Conference on Proceedings of Networking and Distributed Computing (ICNDC)*, pp. 194-197, 2012. Article (CrossRef Link).
[15] Maciej Rostanski, Krzysztof Grochla, Aleksander Seman, "Evaluation of highly available and fault-tolerant middleware clustered architectures using RabbitMQ," in *Proc. of Federated Conference on Computer Science and Information Systems*, pp. 879-884, 2014. Article (CrossRef Link).
[16] Mingzhe Xu, Weiqing Sun, Mansoor Alam, "Security Enhancement of Secure USB Debugging in Android System," in *Proc. of The 12th Annual IEEE Consumer Communications and Networking Conference*, 2015. Article (CrossRef Link).
[17] Kuan-Rong Lee, Meng-Hsuan Fu, Yau-Hwang Kuo, "A hierarchical scheduling strategy for the composition services architecture based on cloud computing," in *Proc. of The 2nd International Conference on Next Generation Information Technology (ICNIT)*, pp. 163-169, 2011.
[18] M. Hristakeva and D. Shrestha, Shrestha, "Different Approaches to Solve the 0/1 Knapsack Problem," in *Proc. of Midwest Instruction and Computing Symposium*, 2005.
[19] V. K. Patel and M. H. Pandya, "Learning of Scheduling Algorithm with Maximum Compatible Activity or Minimum Makespan," *International Journal of Engineering Development and Research (IJEDR)*, vol. 1, no. 2, pp. 121-124, 2014.
[20] S. Mittal and A. Katal, "An Optimized Task Scheduling Algorithm in Cloud Computing," in *Proc. of 2016 IEEE 6th International Conference on Advanced Computing (IACC)*, pp. 197–202, 2016. Article (CrossRef Link).
[21] Sharma, Arpita and Kumar Gupta, Amit and Goyal, Dinesh, "An Optimized Task Scheduling in Cloud Computing Using Priority," in *Proc. of 3rd International Conference on Internet of Things and Connected Technologies (ICIoTCT), 2018 held at Malaviya National Institute of Technology, Jaipur (India) on March 26-27*, 2018. Article (CrossRef Link).
[22] Hongyan Cui, Xiaofei Liu, Tao Yu, Honggang Zhang, Yajun Fang, Zongguo Xia, "Cloud Service Scheduling Algorithm Research and Optimization," *Security and Communication Networks*, vol. 2017, 2017. Article (CrossRef Link)
[23] L. Shi, Z. Zhang, and T. Robertazzi, "Energy-Aware Scheduling of Embarrassingly Parallel Jobs and Resource Allocation in Cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1607–1620, Jun. 2017. Article (CrossRef Link).
[24] N. Manikandan and A. Pravin Albert, "Hybrid - based novel approach for resource scheduling using MCFCM and PSO in cloud computing environment," *Concurrency and Computation Practice and Experience*, pp. 1-9, 2019. Article (CrossRef Link).
[25] Y. Pan, S. Wang, L. Wu, Y. Xia, W. Zheng, S. Pang, Z. Zeng, P. Chen and Y. Li, "A Novel Approach to Scheduling Workflows Upon Cloud Resources with Fluctuating Performance," *Mobile Networks and Applications*, vol. 25, pp. 690–700, 2020. Article (CrossRef Link).
[26] P. Veerendra, and K. Thirupathi Rao, "Nature-inspired cloud processing theory of optimization for adaptivetask schedule," *Materials Today: Proc.*, 2020. Article (CrossRef Link).
[27] Jong-Yih Kuo, T. Y. Chien, "A Novel Approach for Resource Monitoring and Scheduling on Cloud Testing Platform," in *Proc. of the Tenth Taiwan Conference on Software Engineering*, Nantou, Taiwan, 2014. Article (CrossRef Link)
[28] Jong-Yih Kuo, C. Liu and W. T. Yu, "The Study of Cloud-Based Testing Platform for Android," in *Proc. of IEEE International Conference on Mobile Services*, New York, NY, pp. 197-201, 2015. Article (CrossRef Link).
[29] Taiwan Testing and Certification Center. https://www.etc.org.tw/default.aspx

Jong-Yih Kuo received the Ph.D. degree in Computer Science and Information Engineering from National Central University in Taiwan in 2007. He is a Professor in the Computer Science and Information Engineering at Taipei University of Technology in Taiwan. His current research interests include software engineering and intelligent system.

Hui-Chi Lin is currently pursuing a master's degree in the School of Computer Science and Information Engineering at Taipei University of Technology in Taiwan. Her research interests include software engineering, and machine learning.

Chien-Hung Liu received his M.S. degree in Electrical Engineering from University of Southern California in 1994, and his Ph.D. degree in Computer Science and Engineering from University of Texas at Arlington in 2002. He is currently an associate professor of Computer Science and Information Engineering Department at National Taipei University of Technology, Taiwan. His research interests include software testing, vocal detection, and software engineering.