A Hybrid Cloud Testing System Based on Virtual Machines and Networks

Jing Chen[1,2*], Honghua Yan[3], Chunxiao Wang[1] and Xuyan Liu[4]
[1] Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences)
Jinan, 250014 - China
[e-mail: jingchen94@163.com]
[2] College of Computer Science and Engineering, Shandong University of Science and Technology
Qingdao, 266590 - China
[3] Development department, Shandong GreenPower Technology Co., Ltd
Jinan, 250010 - China
[e-mail: sdgrtech@163.com]
[4] Network Management Department, China Telecom Corporation Limited, Shandong Branch company
Jinan, 250101- China
[e-mail: liuxuyan77@163.com]
*Corresponding author: Jing Chen

Received April 15, 2019; revised October 14, 2019; accepted February 7, 2020; published April 30, 2020

Abstract

Traditional software testing typically uses many physical resources to manually build various test environments, resulting in high resource costs and long test time due to limited resources, especially for small enterprises. Cloud computing can provide sufficient low-cost virtual resources to alleviate these problems through the virtualization of physical resources. However, the provision of various test environments and services for implementing software testing rapidly and conveniently based on cloud computing is challenging. This paper proposes a multilayer cloud testing model based on cloud computing and implements a hybrid cloud testing system based on virtual machines (VMs) and networks. This system realizes the automatic and rapid creation of test environments and the remote use of test tools and test services. We conduct experiments on this system and evaluate its applicability in terms of the VM provision time, VM performance and virtual network performance. The experimental results demonstrate that the performance of the VMs and virtual networks is satisfactory and that this system can improve the test efficiency and reduce test costs through rapid virtual resource provision and convenient test services.

Keywords: cloud testing, cloud computing, TaaS, virtual machine, virtual network

A preliminary version of this paper appeared in 2017 Fifth International Conference on Advanced Cloud and Big Data (CBD), August 13-16, Shanghai, China. This version includes a more detailed introduction, implementation and analysis on a hybrid cloud testing system. This research was supported by the National Key Research and Development Program of China (No.2018YFB100360) and Shandong Provincial Natural Foundation (No.ZR2016FM41).

http://doi.org/10.3837/tiis.2020.04.007 **ISSN : 1976-7277**

1. Introduction

Rapid software development and testing are critical for maintaining the competitive advantage and seizing market opportunities for an enterprise. However, many problems, such as high test costs [1] and a low test efficiency, are encountered in traditional software testing. Most small enterprises have too few resources and test tools to build a diverse array of test environments because they have insufficient funds. Although welfare organizations can provide various free resources and test tools for small enterprises, most require these enterprises to use these resources and test tools in limited places and time ranges. Thus, it is not convenient for enterprises to use free resources and test tools.

Enterprises must spend a substantial amount of money on hardware and software resources. Moreover, many testers conduct highly repetitive and inefficient manual tests; for example, they build various test environments, install many test tools and manually implement tests, which results in the low efficiency of traditional software testing. Furthermore, existing test tools cannot simulate the real behaviors of user requests due to the limitations of single nodes and small local area networks (LANs) for the load testing of web applications. Due to the increasing number of test requirements, to perform comprehensive software testing quickly, it is becoming increasingly important to automatically deploy various test environments according to the test requirements and to provide various types of test tools and online services. The key strategy is to build an effective test system to provide various test resources, test tools and composite test services based on existing physical resources, test tools and systems. Cloud computing provides a satisfactory solution to this problem because it can integrate many physical computing, storage and network resources into a data center and transform these physical resources into virtual resources using virtualization technology. Users can realize virtual resources of all types on demand from cloud computing platforms anywhere and at any time, which reduces not only the resource cost but also the installation time for the operating system and software. Cloud computing technology can revolutionize traditional software testing. Enterprises require an automatic software testing platform that has rich test resources and services to improve test efficiency and reduce test costs.

Similar to software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS), testing as a service (TaaS) provides test services for handling test activities of all types on a cloud testing platform. The concept, needs and challenges of TaaS were proposed in some literatures [2-4]. Manveen Kaur [5] introduced the advantages of cloud testing, such as cost-effective, faster-testing, easy management and peak load handling, and new challenges, such as the problems of network congestion, data integrity, data security and privacy. A survey of TaaS in cloud computing is made to analyze the types, steps and forms of cloud testing, which mainly focuses on security testing and performance testing of data integrity, data confidentiality and data availability [6]. Mittal et. al [7] proposed cloud testing is the future norm of software testing and would provide TaaS for SaaS and clouds. TaaS platform can use various virtual resources to automatically provide test environments and support rapid resource provision for massive concurrent testing, inheriting the advantages of cloud computing technology. Moreover, TaaS can substantially reduce test costs through a pay-as-you-go model and improve the test efficiency through automatic and on-demand test services online. Sathe et al. [8] proposed a cost-effective framework for TaaS. Bertolino et al. [9] proposed an open-source platform ElasTest, which can support the testing for a large and

complex cloud applications, such as web, mobile, network.

Compared with traditional testing, TaaS based on cloud computing can support remote and distributed testing by test tools, various test configurations and test service semantics by a large number of resources and services, which can realize the automatic, rapid and cost-effective testing. Leah et al. [10] studied the adoption and use of cloud testing in various organizations and found that users made continuous demands for test resources and that the cloud testing infrastructure can provide various cost-effective solutions to meet users' demands. Previous studies of cloud testing focused mainly on the creation of test environments based on cloud resources [11] and automated testing [12]. Recently, the developed cloud testing platforms typically provide not only test environments but also various test services. Lo et al. [13] proposed a cloud test environment deployment and service model; this model can automatically generate test environments for performing tests and provide test results according to the test scenarios, and testers can define various test scenarios, including test clients, network and scripts. Cao et al. [14] proposed a hierarchical model and approach for constructing a software testing platform based on cloud computing. This platform employs IaaS and PaaS platforms to realize test environment creation and test project management, respectively. Chiang et al. [15] proposed a browser-based distributed testing service platform for web service testing. This platform provides elastic test environments for solving the slow-response problem of single-node testing and simulates large-scale user requests to test location-based web services using worldwide browsers from three countries in East Asia, thereby reducing the time and cost required for testing. Yan et al. [16] proposed a load testing platform of web services, namely, WS-TaaS, which can simulate massive concurrent requests from geographically distributed test nodes. Compared with the traditional web testing tool Jmeter on a single node, WS-TaaS achieves a shorter response time and a lower error ratio for the same number of concurrent requests; however, its test agents run on PaaS and cannot be fully customized, resulting in limited target services. Li et al. [17] introduced a cloud testing platform, namely, “Cloud Open Lab”, which can automatically test and evaluate the features, functionality and performance of cloud systems. Ding et al. [18] proposed a cloud testing platform that integrates the code coverage testing tool SAT. This platform can rapidly resize a VM to satisfy the coverage testing requirement when a massive number of test tasks occur concurrently, thereby reducing the time and cost required for testing. Although this platform can provide test resources and scale them rapidly when necessary, its service focuses only on coverage testing based on SAT. Lee et al. [19] proposed a system that integrates heterogeneous web testing tools into a composite service to reduce the tester's efforts in load and cross-browser testing. This system can automatically convert the inputs and outputs of various testing tools, initiate a testing process and send a compiled test report via email to reduce testers' workloads using the composite web testing service. The results of an experiment demonstrate that this system can substantially reduce the test time for load testing and cross-browser testing. Additionally, cloud testing has been applied in various applications, such as web applications [20], mobile applications [21,22], multimedia applications [23], bank applications [24], vessel traffic systems [25], Internet of Things (IoT) [26] and combinatorial testing [27]. Some researchers studied the infrastructure, needs and test process of mobile testing as a service (MTaaS). Zhang et al. [28] proposed a function test approach for mobile testing based on TaaS, where test scripts can be generated automatically to execute the testing. Tao et al. [29] proposed a mobile testing system MTaaS, compared and discussed mobile TaaS approaches and several industrial practices. They also analyzed the requirements, issues and challenges in mobile TaaS. Certainly, not all software testing is adaptable for TaaS. Ali et al. [30] proposed an

adoption assessment model based on a fuzzy multi-attribute decision making algorithm, which works in the prediction and assessment modes to support software development organization (SDO) to decide moving suitable software testing to the cloud. Eid et al. [31] proposed a mutation-based approach to evaluate the quality of TaaS including the provision ability of TaaS provider and the successful rating of TaaS using mutation scores. Oliveira et al. [32] proposed an approach to solve the lock-in problem caused by the lack of portability and interoperability among TaaS platforms, which uses design patterns and environment variables to avoid the lock-in problem in TaaS.

Cloud testing platforms provide cloud test services in industry. For example, SOAtest is Parasoft's testing platform, which provides UI, API, function, load, performance and security testing [33]. CloudTest is a performance testing tool for website and mobile applications [34]. Sauce Labs provides a platform that supports automated web testing and mobile app testing [35]. Some testing platforms, such as Xamarin Test Cloud [36], MQC from Alibaba [37], WeTest from Tecent [38], MTC from Baidu [39] and Testin [40], focus mainly on mobile testing. However, most of these platforms primarily provide various mobile devices for testing the function, performance, compatibility and security of mobile applications, but they do not construct a cloud testing platform based on the users' existing resources and test tools to automatically create test environments, remotely use test tools, and execute a series of test activities. And most cloud testing systems concentrate on test environment creation and some test services provision, which cannot support the whole life-cycle of online software testing. Two main challenges exist in designing such a cloud testing system. One is the multi-tenancy problem supporting the third-party independent software vendors (ISVs). A cloud testing system involves in not only test environments and test tools based on virtual machines, but also SaaS services from ISVs. The cloud testing system need to support a large number of users to implement test activities online. Once a user logs on this system, he can access all test services whatever they are from ISVs or this system. The key problem is to realize the single sign-on (SSO), which can help users access to a lot of SaaS services of ISVs. Another is to design a whole life-cycle test process online. A whole life-cycle test process basically includes test environment preparation, test tools deployment, test implementation and test management. They involve in different test processes, test environments and test requirements, which will cause more difficulties to move the offline test activities to the online cloud testing system and then set up a whole life-cycle test process. One key issue is to realize network isolation, network connectivity and data transformation.

This paper extends the findings of our previous study [41], in which we proposed a basic model and constructed an online software test platform based on cloud computing; additionally, we performed VM-scale testing to evaluate the capacity of this platform. However, our previously proposed test platform focuses only on providing rich VMs and test services to support test activities, whereas the ways in which VMs and virtual networks support various test scenarios and the performance of a VM and a private virtual network are not considered. Accordingly, this paper proposes a hybrid cloud testing system that not only implements TaaS based on VMs and virtual networks but also tests the performance of virtual resources. In contrast to [41], our new work improves the previous software online testing system in terms of virtual network management and SaaS access, presents a more detailed introduction and implementation to our proposed TaaS system, and analyzes the performance of VMs and private virtual networks. This TaaS system realizes the rapid creation of test environments and the remote use of test tools and online test services, thereby enabling users to implement their test activities online. The main components of this

system are discussed. Furthermore, VM provision-time testing, VM performance testing and virtual network testing are conducted to determine whether they are satisfactory.

2. Background

2.1 Virtual Machine

Virtualization technology can divide a physical computer, a network device and a storage device into multiple VMs, virtual networks and virtual storage devices, respectively. Two types of virtualization technologies (hardware virtualization and operating system-level virtualization) are used in cloud computing. Hardware virtualization can enable multiple VMs with different operating systems (OSs) to run on a physical server while preserving their isolation in environments via a hypervisor. A hypervisor, such as XEN, KVM, VMware EXSi or Hyper-V, is a VM monitor that enables multiple OSs and applications running on VMs to share the same hardware. A hypervisor also controls the interactions between the OS of a VM and the hardware of a physical server. Operating system-level virtualization can enable multiple containers to share the kernel of a physical server but use different user spaces via kernel partitioning. A container is a lightweight virtualization technology; it uses fewer resources to serve an application than a VM, which uses more resources to run an entire operating system. Although containers have more advantages than VMs [42], they are weaker in terms of security. If a superuser right is illegally obtained from a container, the underlying operating system may be cracked. Additionally, computer viruses infecting a container can affect all containers due to the shared OS. A VM is a simulated environment based on the computer architecture of a physical server, which includes a series of hardware (CPU, memory, hard disk and network) and software (OS and application) components. VMs separate applications and OSs from hardware, thereby eliminating the interference not only between applications of VMs but also between applications and physical hardware. A VM is less vulnerable than a container because of its better isolation at software level, and its behavior and functionality are almost the same as those of a physical computer. Different types of VMs with different OSs can be automatically created and rapidly provided to end-users from the cloud platform.

2.2 Virtual Network

Network virtualization can establish connections between different applications, services and users. It can also simulate networks of all types to construct test environments for software testing. In cloud computing, VMs are combined in a virtual network obtained via network virtualization. Bridge mode and network address translation (NAT) mode are used in early network virtualization. Bridge mode connects the virtual network interface card (NIC) to the physical NIC. Under NAT mode, a VM accesses the public network through the host network using the NAT service. The OpenStack cloud platform uses Neutron and virtual switches, such as Open vSwitch (OVS), to generate virtual networks and switches. Neutron, an OpenStack component, provides "networking as a service" by creating virtual networks, subnets, ports, switches and routers. OVS is a multilayer virtual software switch that provides network connections between VMs. **Fig. 1** illustrates the traffic flow between two VMs from different compute nodes on the same virtual network. VM2, which runs on compute node 1, forwards a packet to the Linux bridge, namely, qbr, and subsequently to the OVS integration bridge, namely, br-int, and the tunnel bridge, namely, br-tun. Then, the packet is transmitted from compute node 1 to compute node 2 via the GRE tunnel, and it

reaches VM1 via br-tun, br-int and qbr on compute node 2. Compared with the VMs on the same network, a packet from VM1 on compute node 1 and virtual network 1 cannot be forwarded to VM1 on compute node 2 and virtual network 2 until the route between two interfaces, namely, qr-1 and qr-2, is set up in the router namespace, namely, qrouter, on the network node, as shown in **Fig. 2**, where the qr-1 and qr-2 interfaces separately contain the virtual network 1 and 2 gateway IP addresses. **Fig. 3** illustrates the traffic flow that a VM accesses internet websites. A VM forwards a packet to compute node NIC via the Linux bridge, namely, qbr, the OVS integration bridge, namely, br-int, and the tunnel bridge br-tun. Then, the packet is transmitted to network node NIC via the GRE tunnel. Finally, it accesses the internet website via OVS, qrouter, OVS external bridge, namely, br-ext and another NIC on the network node.

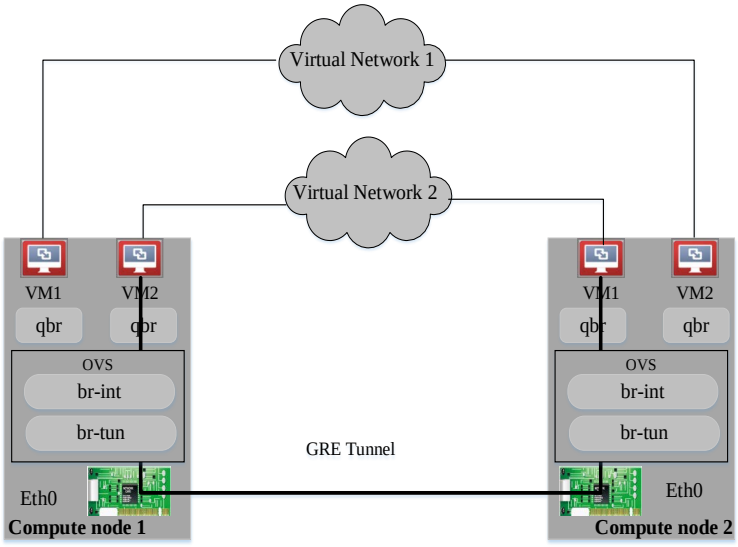


Fig. 1. Traffic flow between two VMs on the same virtual network

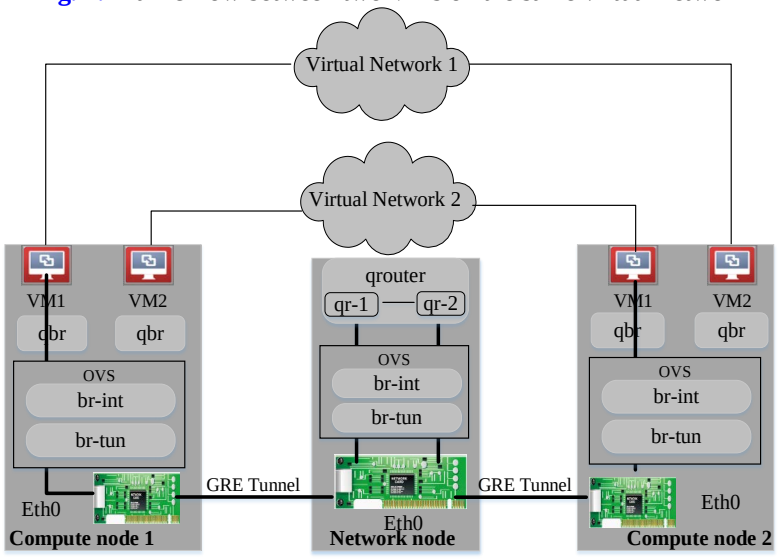


Fig. 2. Traffic flow between two VMs on different virtual networks

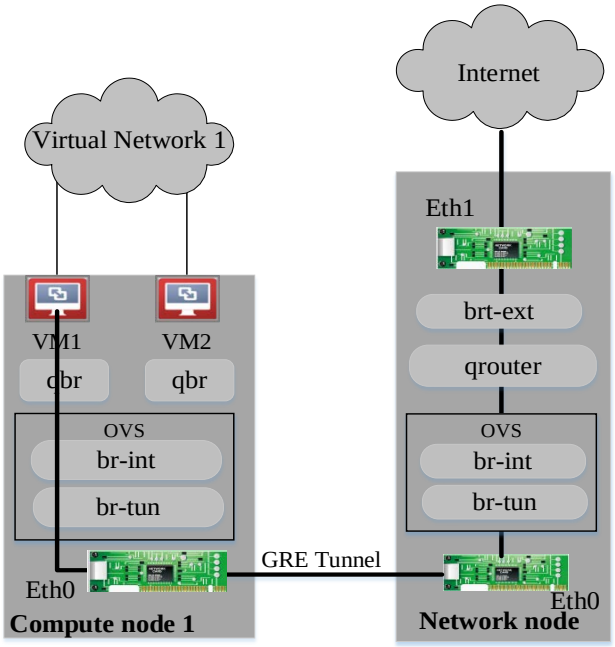


Fig. 3. Traffic flow from a VM to the internet

3. A Hybrid TaaS System based on a Multilayer Model

3.1 Architecture and Model of a Hybrid TaaS System

The basic life cycle of a TaaS system includes test plan setup, test preparation, test implementation and test result management. A test plan including the information and scheduling of a test project, is first setup. Then, various test preparations, such as test environment creation, test tool deployment and test service ordering, are executed. Furthermore, various test activities, such as code analysis, function testing and performance testing, are implemented. Finally, bugs are submitted to the management module of the TaaS system. Our hybrid TaaS system illustrated in **Fig. 4** provides three main functions based on VMs and virtual networks: automatic creation of various test environments, remote use of test tools, and online test services. These components are independent and can be easily assembled according to a user's existing resources. Test targets can be deployed on either VMs provided by this TaaS system or users' own machines. The test environments refer to the runtime environments of test clients or test targets. For example, "Windows 7.0" and "Microsoft Office 2013" are basic running environments of test clients. "Tomcat 7.0" is installed on a VM with OS "Ubuntu Server 14.04" to run a website. The runtime software can be integrated into an image. Thus, a test environment can be built automatically when a VM is created. The remote use of a test tool is realized similarly by integrating a test tool into an image and using this image to create a VM with the test tool. An enterprise customer can create various VMs with different virtual networks, test environments and test tools and further deploy test cases and test scripts on these VMs to implement test activities. Additionally, the user can remotely use test services, such as code analysis and performance testing services, to test the application regardless of whether it runs on this TaaS system or on a personal machine. Test bugs can be saved and managed using the bug management

service. Finally, test environments can be removed automatically or manually, and test reports and data can be collected and downloaded from this system after the test has finished.

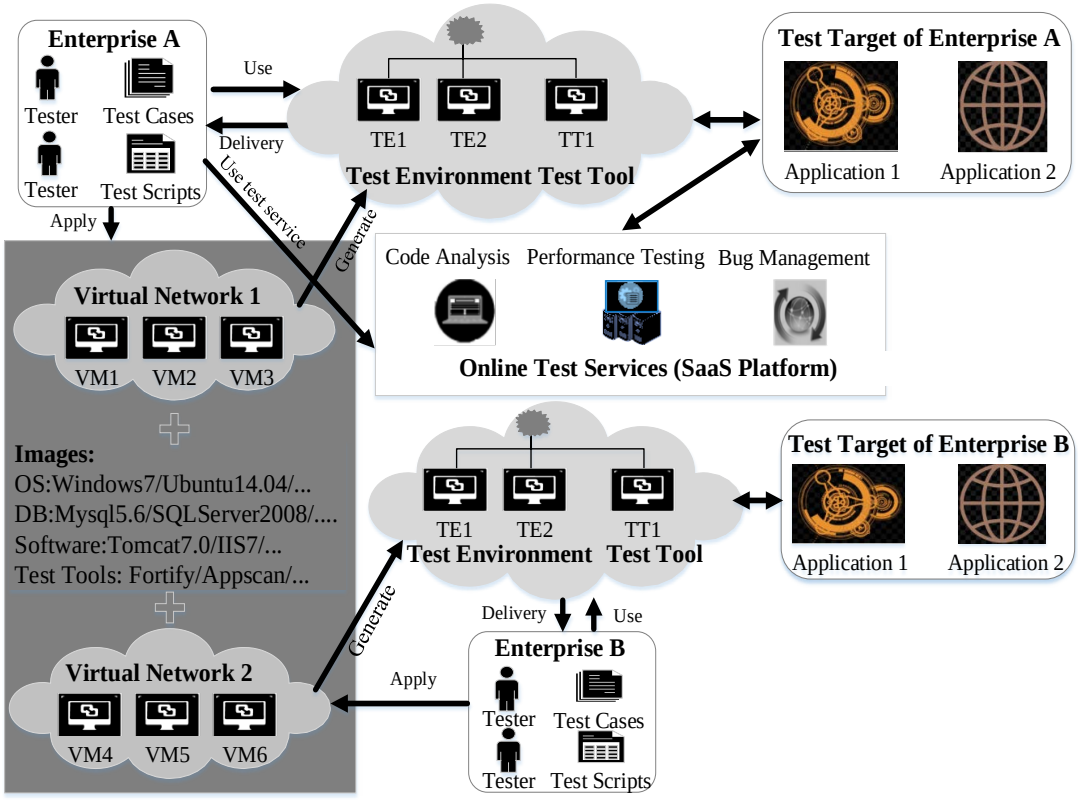


Fig. 4. Architecture of a hybrid TaaS system

Fig. 5 illustrates the multilayer model of a TaaS system. There are four layers: physical layer, IaaS layer, SaaS layer and portal layer. The physical layer integrates various compute, storage and network resources from different places into a pool to support virtual resource provision on an IaaS platform. The IaaS platform virtualizes physical servers and networks using virtualization techniques and manage all physical and virtual resources. The controller node manages all other physical nodes. A storage node stores the images and disks of VMs. A compute node runs various VMs. A network node sets up the communication between two VMs from different compute nodes. Various VMs and virtual networks can be provided based on this IaaS platform. The SaaS platform provides the access specifications of SaaS services. These SaaS services can be deployed on either the servers of the ISVs or the VMs of the IaaS platform. ISVs need to modify their SaaS services according to these access specifications. The modified SaaS services are released on this platform by submitting their information and launching them as SaaS service products. The service portal provides the services of test environment, test tool, performance testing, code analysis and bug management. These services are supported by the resources of the IaaS and SaaS platforms. The management portal provides resource management, order management, service management, enterprise management and ISV management services to help administrators maintain the TaaS system. We use this model to build an online software testing platform based on cloud computing to realize basic test activities in traditional software testing, such

as the automatic creation of test environments, the remote use of test tools and the use of online test services.

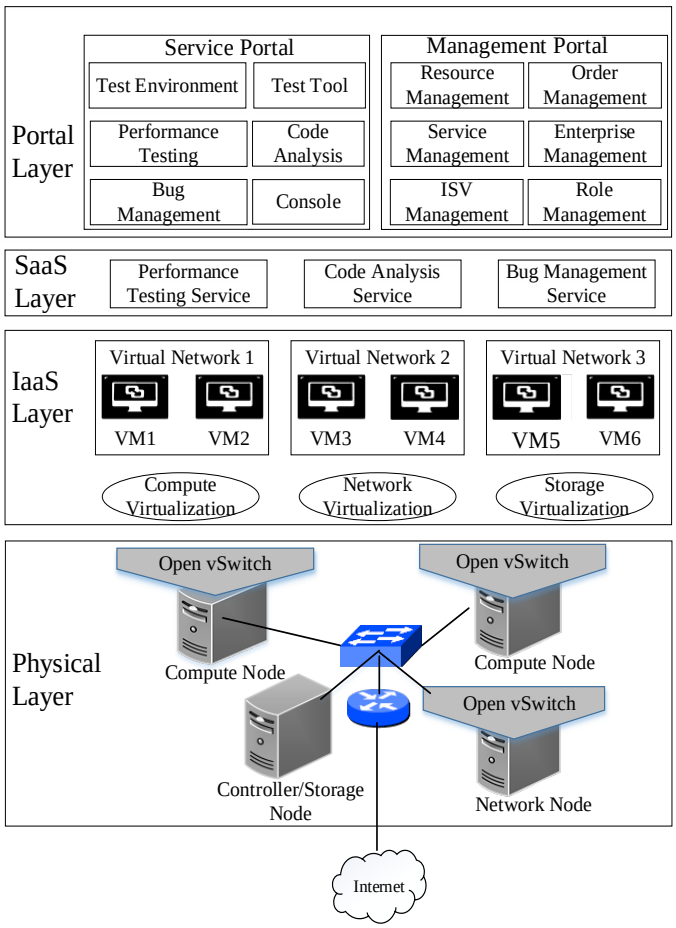


Fig. 5. A multilayer model of a TaaS system

3.2 Main Components of a Hybrid TaaS System

(1) IaaS Platform

We use an open source software, namely, OpenStack [43], and the KVM virtualization technique to build an IaaS platform. This platform provides various resources and services based on different components of OpenStack, such as keystone, nova, glance, neutron and cinder. Keystone provides identity, authentication and access management; it authenticates users and determines which resources users have the authority to access. Nova provides VMs and manages their lifecycles by calling the libvirt application programming interface (API). Glance provides an image service that can discover, register, and retrieve VM images. Cinder provides a block storage service that provides volumes to VMs. Neutron provides a network connectivity service between interface devices (e.g., virtual network interface cards). These components provide VMs and virtual networks that support the TaaS system by interacting with each other. VMs with different operating systems are created based on images of various types, such as Ubuntu, CentOS, and Windows. In addition, private virtual networks are provided to ensure the isolation of test environments for different enterprises.

(2) SaaS Platform

Many test services are developed as SaaS services that can be integrated into the TaaS system. SaaS service access involves service deployment, configuration and management. SaaS services from ISVs can be deployed on VMs of the IaaS platform, whose resources can be rapidly increased or reduced in response to load changes in the testing process. We develop a service management function to provide the configuration and management of SaaS services and solutions. A SaaS service must satisfy the following conditions before it can be accessed by the TaaS system:

(a) It must be a B/S structure. Users can access this service via browsers.

(b) It must support multiple tenants and separate applications and data for different tenants.

(c) It should be chargeable, and its charging strategy should be submitted to this TaaS system.

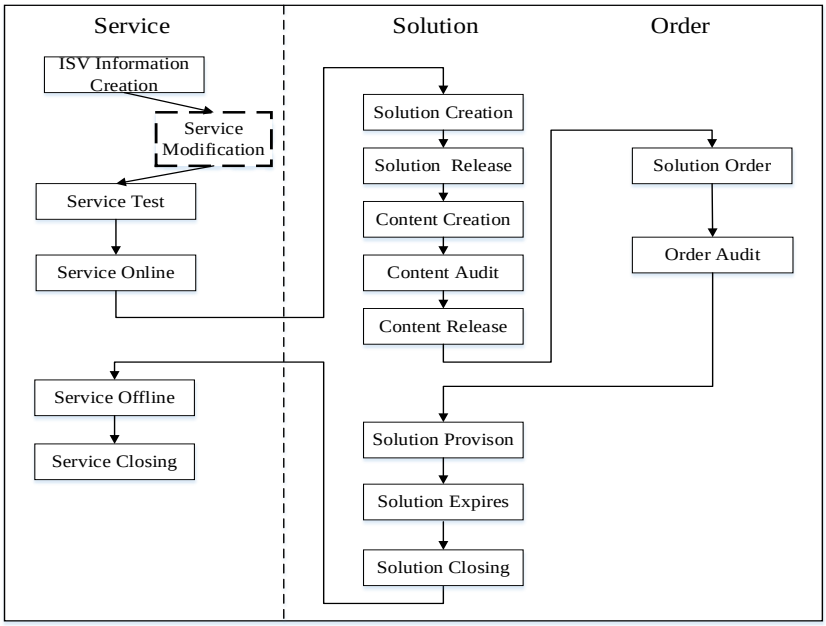


Fig. 6. Process of a SaaS service accessing a TaaS System

Fig. 6 illustrates the process of integrating a SaaS service from an ISV into the TaaS system. First, the ISV and SaaS service information is created, which involves “ISV ID”, “Log Account”, “Key”, and “SaaS Service Number”, etc. Second, the SaaS service must be modified according to the service access specifications of this TaaS system, including the modification of functions and interfaces. Function modification mainly focuses on deleting the overlapping functions between the TaaS system and this SaaS service, such as user management, role management, SSO and password modification. Interface modification mainly includes the service ordering, user authorization, role synchronization, SSO authentication and SSO heartbeat keeping interfaces. These interfaces must comply with the specifications of the web service and the TaaS system. The simple object access protocol (SOAP) and post page are used to transform messages and transfer data between the interfaces of the TaaS system and the ISV's SaaS services. Third, the SaaS service is tested in a sandbox after its modification has been completed. Fourth, the SaaS service is integrated into the TaaS system and is online. Fifth, one or more services are combined into a solution

for release. Sixth, the content of the solution is created, audited and released. Thus, the solution with this SaaS service can be online, ordered and used. If the solution expires, it will be closed. The corresponding SaaS services can be offline and closed.

(3) Service and Management Portals

The service and management portals are developed by using .NET. Some of their services are presented in **Fig. 5**. The service portal mainly provides online test environments, test tools and test services such as code analysis, performance testing and bug management. Users can use and manage their resources and services via their consoles. All resources and services can be obtained anywhere and at any time if users log into this TaaS system. A test environment is presented as one or more VMs with operating systems, clients, network topologies, and test scripts. In contrast to development environments, test environments must remain independent and isolated for different enterprises, which has an advantage in terms of the creation of independent and isolated test environments based on cloud computing. Various test environments can be constructed in batches using different VM images and virtual networks. Each enterprise has its own virtual network, thereby guaranteeing the isolation and privacy of the test environment. Enterprise data can also be isolated by separating different users and projects in OpenStack. If two VMs from different enterprises (tenants) communicate with each other, their private virtual networks should connect with the external network via their own routers, as illustrated in **Fig. 7**. For example, VM1 of tenant A forwards packets to VM1 of tenant B via tenant A's private virtual network and router, the external network, and tenant B's virtual router and network, sequentially. A test tool is also provided in VM mode and used remotely. Additional VMs can be rapidly provided when the resource utilization of a test environment or a test tool is overloaded. Test services from ISVs are integrated into this TaaS system and provided in the SaaS platform. Among these test services, the performance testing service can test a web application in parallel to shorten the test time regardless of whether this application is deployed on a user's server or on a VM of the TaaS system.

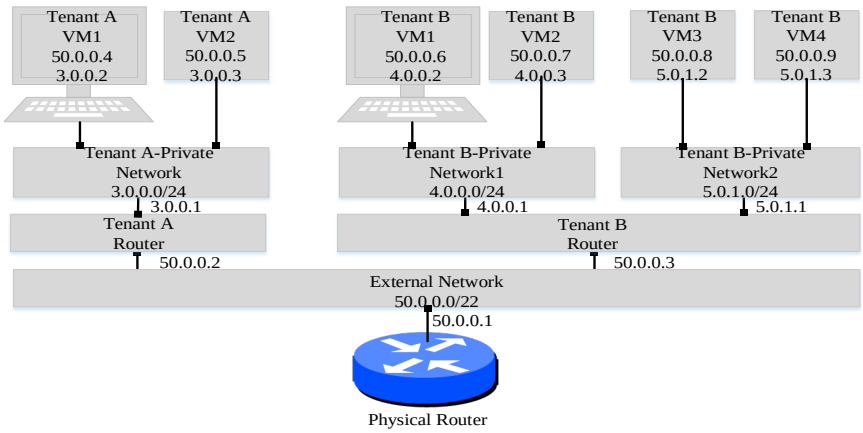


Fig. 7. Communication between different private networks

The management portal provides many management functions, such as enterprise management, resource management, service management, order management, and ISV management, as illustrated in **Fig. 8**. Enterprise management refers to the management of the information of enterprises and enterprise employees. Resource management refers to the management of various physical and virtual resources, including physical hosts, VMs, images, flavors, virtual networks and routers. Service management mainly refers to the

management of solutions, contents and SaaS services, such as code analysis, performance testing and bug management. These SaaS services and relevant supporting systems, such as Jmeter and Sonar, are deployed on VMs. Order management refers to the management and audit of all orders and the handling of abnormal orders. Users cannot obtain or use resources or services until the order is approved.

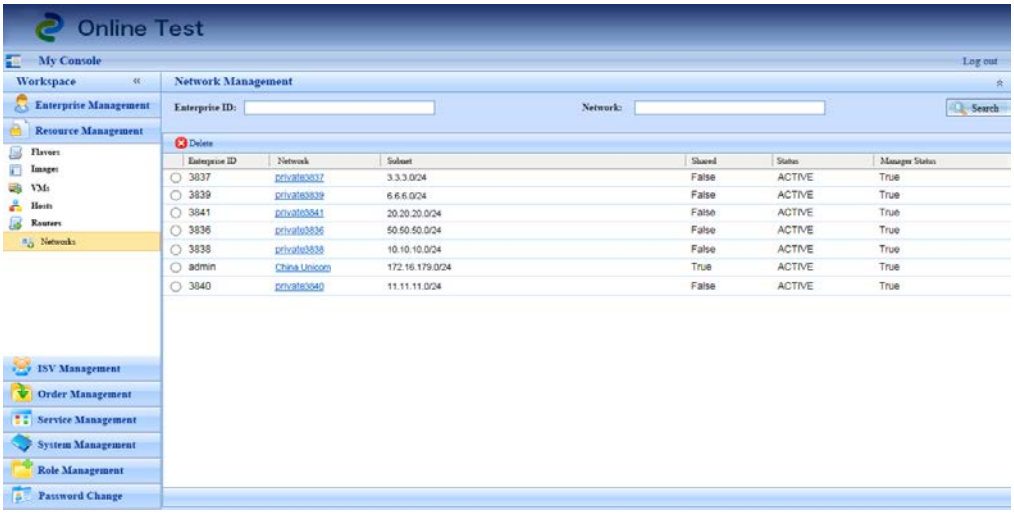


Fig. 8. Management portal

4. Experimental Setup and Results

A key objective of a TaaS system is to provide test resources and test services in an automatic and convenient manner. We implement a TaaS system based on a cloud platform, which includes 4 physical nodes (1 controller node, 1 network node and 2 compute nodes). This TaaS system provides not only the automatic creation and remote use of test environments and test tools based on VMs and virtual networks but also online test services, such as performance testing based on open source software Jmeter, code analysis based on open source software Sonar and bug management, as illustrated in **Fig. 9**.

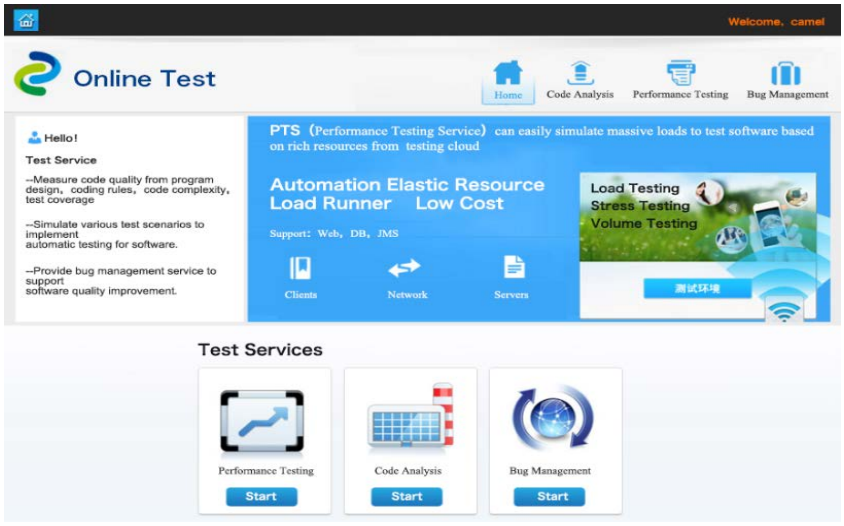
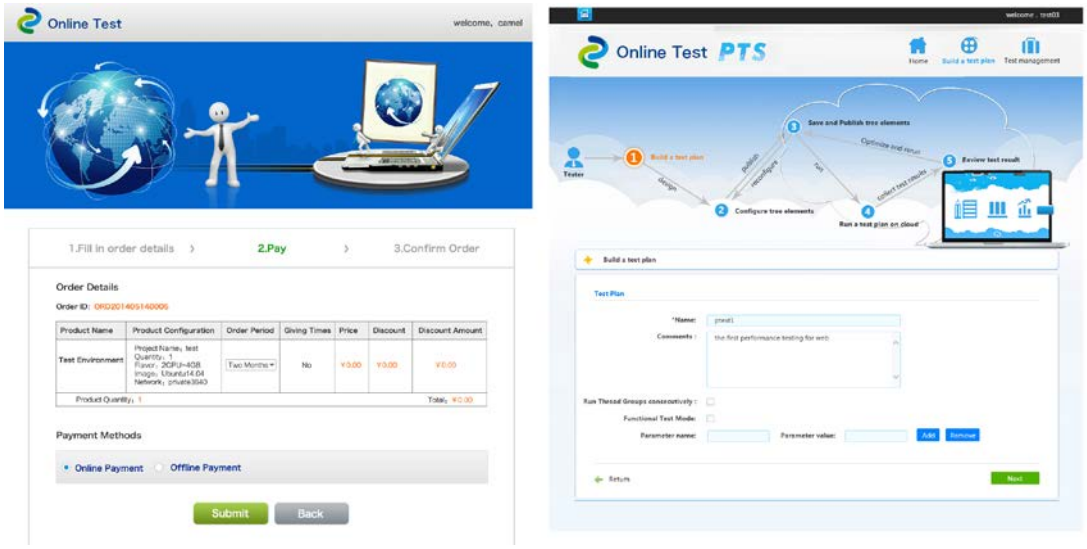


Fig. 9. Test services

In addition, tenants and ISVs in this system have two roles: ISVs provide and release their test services on the TaaS system, and tenants can order test resources and use these test services to implement their test tasks. The TaaS system automatically provides various test resources and test services. For instance, tenants can order test environments in which to deploy the systems being tested, as illustrated in **Fig. 10(a)**, and apply test tools, such as Fortify and Appscan, or test services, such as performance testing and code analysis, to test these systems, as illustrated in **Fig. 10(b)-10(d)**. Tenants can also install their own test agents or test scripts on test environments to test the systems deployed on their local machines. These features provide convenient test services to enterprises and accelerate their test processes. Furthermore, smooth operation and management are realized, thereby improving the maintenance efficiency.



(a) Test environment ordering (b) Test plan for performance testing

(c) Results of performance testing (d) Results of code analysis

Fig. 10. Partial functions of our TaaS system

Another objective is to rapidly provide test resources and ensure that they have good performance. To evaluate the performance of our TaaS system in practice, we implement three experiments: VM provision-time testing, VM performance testing and virtual network testing.

4.1 VM Provision-Time Testing

A TaaS system must provide many VMs to support various test activities quickly. We perform large-scale VM testing to evaluate the resource provision performance of our system. This testing approach adopts a VM image, namely, "CentOS Server 6.5", and selects a 1CPU2G20G flavor with 1 core CPU, 2 GB memory and 20 GB disk. Because each compute node has only a 600 GB disk, the maximum number of VM requests is only 45 in theory, excluding the occupation volume of the operating systems, software and VM files of the two compute nodes. We perform testing for 10, 20, 30, and 45 VM requests and present the results in terms of the VM provision time in **Fig. 11**. A VM is typically provided within the range [50 s, 120 s]. Moreover, the provision time increases gradually with an increase in the number of VM requests. For instance, the average provision time of a VM is only 54.4 s if there are 10 VM requests; however, it increases up to 61.45 s, 85.7 s, and 101.2 s when there are 20, 30, and 45 VM requests, respectively. The same phenomenon is observed for the maximum and minimum provision times. The standard deviation also exhibits this trend because VM provision becomes less stable with an increase in the number of VM requests. Although the provision time of a VM increases with the number of VM requests, it is substantially reduced compared with the traditional approach, which often requires tens of minutes to set up test environments manually.

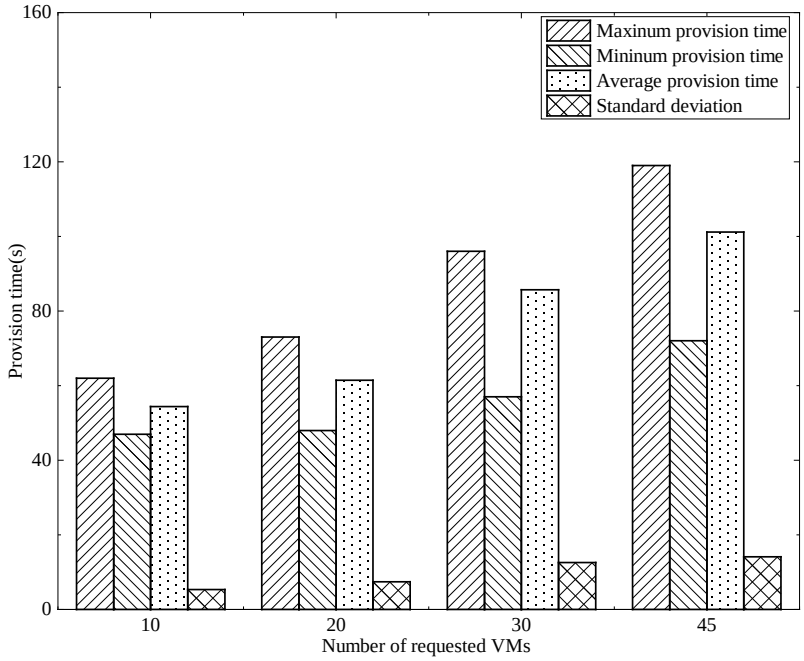


Fig. 11. Provision times of VMs

4.2 VM Performance Testing

We use an open source software, namely, "UnixBench" [44], to test the performance of the VMs from our TaaS system, Alibaba Cloud and QingCloud platforms. These VMs have the same image of OS "Ubuntu Server 14.04.5" and different flavors with 2CPU4G20G and 4CPU8G50G. We test each VM three times. Although the results of each test differ for each VM, the fluctuation is within a small range and does not affect the overall comparison. The system benchmark index scores of all VMs are listed in **Table 1**. Our VMs obtain scores that are near or over 1000; hence, the VMs perform well in this test. However, the performance

of most VMs from Alibaba Cloud and QingCloud are approximately 1.5 times those of ours. We further compare the index scores of individual test items for these VMs, as shown in **Table 2**, where all index scores are relative values obtained by comparing the test results with the baseline values. The VMs from our TaaS system outperform those from other cloud platforms on test items "Dhrystone" and "Whetstone", which indicates our VMs have a certain advantage in string handling and floating-point operations at that time. However, our VMs fall behind other VMs on other test items, such as file copying, execl throughput, pipe throughout, pipe-based context switching, shell scripts, process creation and system call overhead.

Table 1. VM performance of various cloud platforms

VMs	**2CPU4G20G**			**4CPU8G50G**		
Source	VM flavor type	2 CPUs, 1 parallel process	2 CPUs, 2 parallel processes	VM flavor type	4 CPUs,1 parallel processes	4 CPUs, 4 parallel processes
TaaS system	general type	1004	1561	general type	934	2396
Alibaba Cloud	ecs.n1.medium	1007	2052	ecs.n1.large	1075	3216
Alibaba Cloud	ecs.sn1.medium	1452	2600	ecs.c5.xlarge	1444	3343
QingCloud	basic type	1405	2316	basic type	1511	3315
QingCloud	enterprise type	1487	2331	enterprise type	1914	3631

Table 2. Index scores of individual test items

	2CPU4G20G (2 CPUs; 2 parallel processes)					**4CPU8G50G (4 CPUs; 4 parallel processes)**				
Cloud platform	TaaS system	Alibaba Cloud		QingCloud		TaaS system	Alibaba Cloud		QingCloud	
VM type	general type	ecs.n1. medium	ecs.sn1. medium	basic type	enterprise type	general type	ecs.n1 .large	ecs.c5. xlarge	basic type	enterprise type
Dhrystone 2 using register variables	5054	4151	5794	4966	4803	9884	8798	7015	9731	11854
Double-precisi on whetstone	1631	1253	1464	1355	1582	3205	2560	2505	2684	2903
Execl throughput	1313	1685	2102	1582	1643	2175	2821	3015	2913	3398
File copy 1024 bufsize 2000 maxblocks	1465	2204	2480	2008	2036	1361	1949	2250	1986	1691
File copy 256 bufsize 500 maxblocks	966	1410	1673	1512	1492	951	1417	1440	1199	1228
File copy 4096 bufsize 8000 maxblocks	2961	3953	4894	3970	3660	2903	4133	4587	3476	3514
Pipe throughput	1356	2312	2917	2990	2917	2618	4929	4117	5834	7088
Pipe-based context switching	648	1138	1491	1605	1625	1156	2161	2111	2538	3289
Process creation	1072	1462	1757	1393	1323	1576	1861	1911	2229	2474
Shell scripts (1 concurrent)	1873	2886	4095	3243	3362	3536	5577	6089	5532	6932
Shell scripts (8 concurrent)	2495	2570	3660	2941	2996	4488	5055	5433	5274	6774

System call overhead	1045	1815	2302	2905	2932	1831	3023	4499	2898	2269
System benchmarks index score	1561	2052	2600	2316	2331	2396	3216	3343	3315	3631

Additionally, we implement a benchmark test to obtain the "copy" memory bandwidth available to these VMs. The test results are listed in **Fig. 12(a)-12(b)**. The memory bandwidths of our VMs are larger than those of other VMs except in the MEMCPY test for the flavor 2CPU4G20G. The larger the memory bandwidth is, the higher the VM performance at that time.

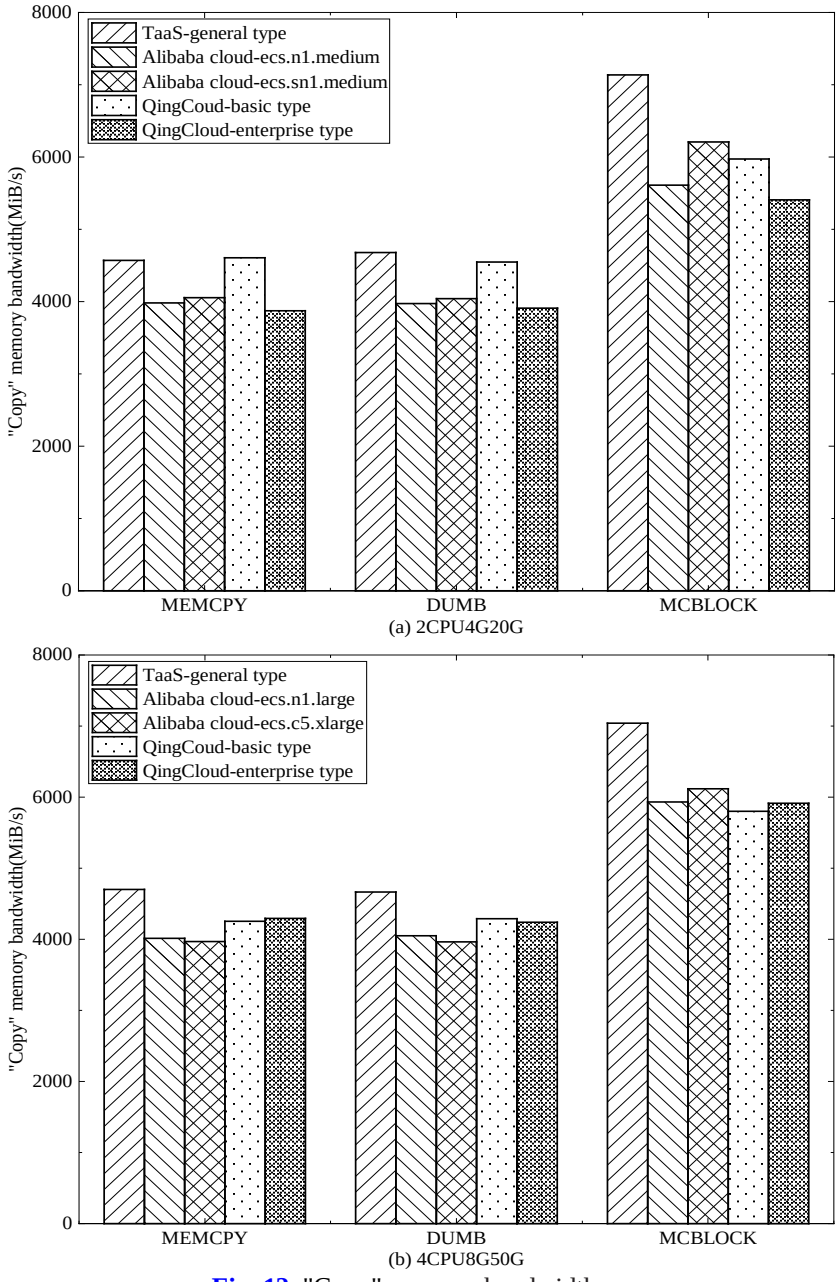


Fig. 12. "Copy" memory bandwidth

4.3 Virtual Network Testing

In this experiment, we measure the real-time bandwidths of the virtual network and the physical network. Then, we measure the average round-trip times (RRTs) from a VM and a physical node of our cloud platform to domestic and foreign websites. Finally, we create two groups of VMs according to their network topologies on which to perform RRT tests.

Fig. 13 presents the real-time bandwidths of the virtual and physical networks in the internal network and internet connection tests. We use Linux "iperf" and "speedtest" tools to implement the internal network bandwidth and the internet connection bandwidth measurements, respectively. The uplink and downlink network bandwidths for the virtual network are lower than those for the physical network. The uplink and downlink network bandwidths between two VMs based on the same virtual network are slightly lower than those between two physical nodes (PNs) based on the same physical network. The uplink and downlink network bandwidths between two VMs based on the same virtual network are 937 Mb/s and 921 Mb/s, respectively, while those between two PNs based on the same physical network are 941 Mb/s and 933 Mb/s, respectively; the differences between them are very small. However, their differences increase in the internet connection bandwidth tests. We use "speedtest" tool to test the network bandwidth from one client to the nearest speedtest server on internet. The virtual network has an uplink network bandwidth of only 85.58 Mb/s, which is slightly lower than the bandwidth of 92.89 Mb/s of the physical network; however, it has a downlink network bandwidth of only 62.40 Mb/s, which is far lower than the bandwidth of 91.82 Mb/s of the physical network in the internet connection tests. Although the virtual network performance is slightly lower than the physical network performance, it is acceptable.

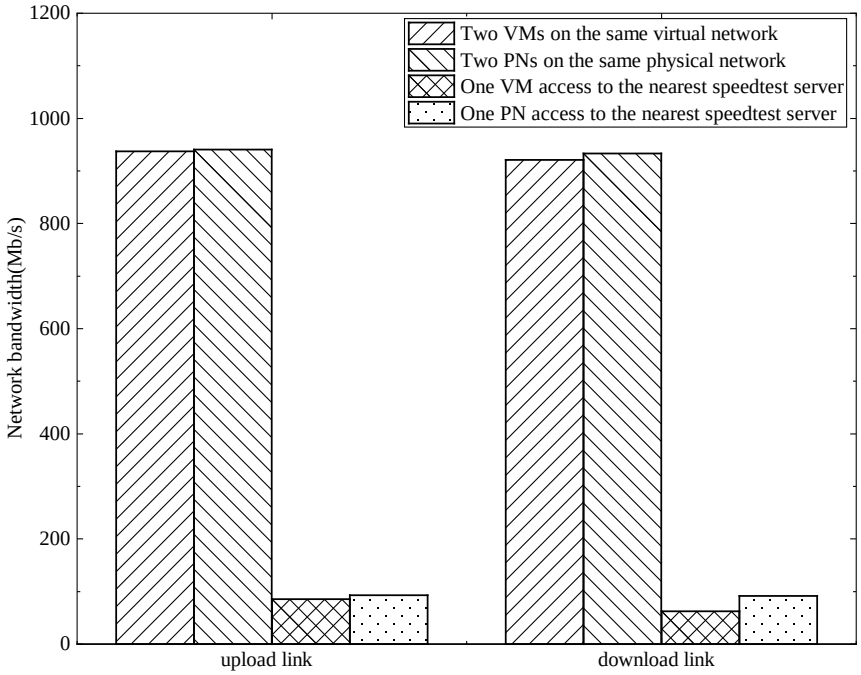


Fig. 13. Network bandwidth testing

We use the Linux "ping" command to measure the round-trip time (RTT) from a VM to an internet website. **Fig. 14** presents the average round-trip times (RTTs) that a packet travels

from a VM to websites and back again 100 times. The average RTTs to the domestic websites, such as baidu.com and 163.com, are within 21 ms. However, those to foreign websites all exceed 48 ms, and some even exceed 200 ms, e.g., oracle.com and github.com. In addition, the average RTTs from the VM to internet websites all exceed those from a PN to websites, mainly because a packet from a VM travels multiple bridges to the physical network interface card (NIC) of the compute node running this VM and subsequently travels to the network node and accesses internet websites through its physical NIC, as illustrated in **Fig. 3**. Therefore, the access to internet websites by a VM is always slower than that by a PN; however, the difference is small.

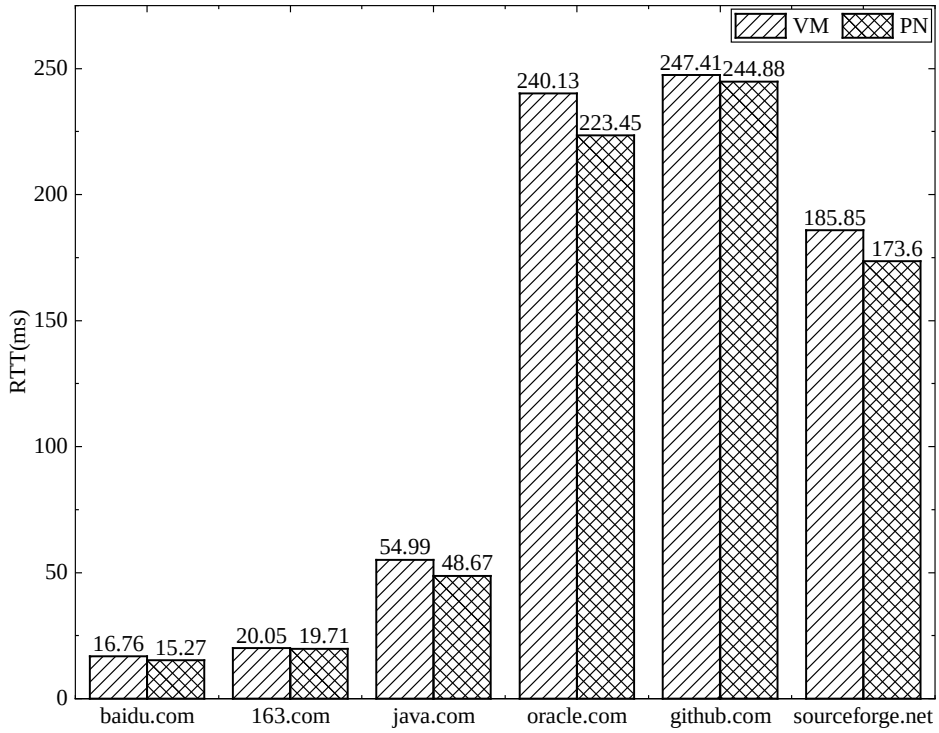


Fig. 14. RTTs of pinged websites

To analyze the performance of a virtual network, we construct virtual and physical networks of various topologies and prepare two groups of VMs and two groups of PNs, respectively, for the RTT tests. One group has two VMs on the same private virtual network; another group has two VMs on different private virtual networks. Similarly, we prepare two groups of PNs on the same physical network and different physical networks. **Fig. 15** shows the average RTTs between VMs and those between PNs for various packet sizes. The average RTTs between two PNs are all lower than those between VMs. Hence, the virtual network has a transmission latency due to the bridges between the virtual NIC and the physical NIC. The PNs on different networks show lower transmission performance than those on the same network, which may be due to the transmission latency between networks through a router. Their difference increases very slowly with an increase in the packet size. However, a different trend is observed in the RTT tests between VMs. The average RTTs between VMs on different networks are higher than those between VMs on the same network, and their difference increases with an increase in the packet size. For instance,

when we use the 64-byte packet from one VM to ping another VM, the average RTT between them is 1.841 ms, which corresponds to a 31% performance drop compared with the average RTT of 1.406 ms between two VMs on the same network. Furthermore, the average RTT between VMs on the same network increases from 1.406 ms to 4.597 ms if the packet size is increased from 64 bytes to 40960 bytes, whereas the average RTT between VMs on different networks increases from 1.841 ms to 10.359 ms. The data transmission latency increases as expected when the packet size increases. However, the performance degrades substantially with an increase in the packet size. The performance drops by 125.34% for a 40960-byte packet. These phenomena may correspond to issues in the transmission of big data on different private virtual networks.

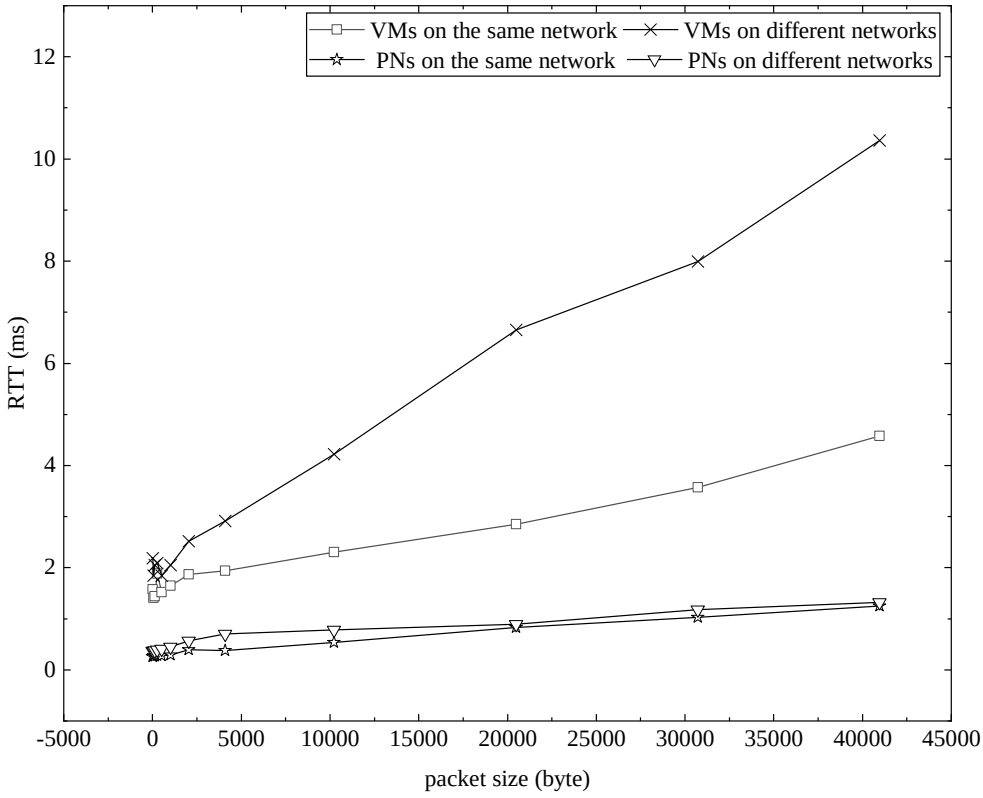


Fig. 15. RTT test between two VMs

5. Validity

The threat to validity is mainly related to VMs. In this paper, the VMs are from different cloud platforms, such as Alibaba Cloud, QingCloud and our private TaaS system. These cloud platforms use physical servers that differ in terms of their properties and performance. VMs with different resource configurations differ in terms of performance. VMs from different cloud platforms also differ in terms of performance, even though they have the same resource configuration. Moreover, the performance of a VM may vary over time. We create two types of VMs for the same resource configuration from Alibaba Cloud and QingCloud. For example, we create two VMs that have the same resource configuration (i.e., 2CPU4G20G) and different flavors (i.e., “ecs.n1.medium” and “ ecs.sn1.medium”) on the

Alibaba Cloud. Two VMs that have the same resource configuration (i.e., 2CPU4G20G) and different types of flavors (i.e., “basic type” and “enterprise type”) are created on the QingCloud platform. Our TaaS system provides only a general type of VM. These VMs are created at different moments. The performance testing for these VMs demonstrates only their performance at that time, and the test results reflect the real-time performance of the VMs at that time. Although the performance of a VM is not fixed, the results of performance testing reflect the real-time characteristics of the VM and can be compared with the results of other VMs. The test results can differ slightly over time and among batches of VMs. Additionally, we only compare four specific types of VMs from Alibaba Cloud and QingCloud with those from our TaaS system, respectively. More different VMs from these platforms need to be further tested in the future.

6. Conclusion

Traditional software testing suffers from high resource costs and long test times due to limited resources and manual test activities. Cloud computing can rapidly provide many low-cost resources to alleviate these problems. In this paper, we propose a multilayer cloud testing model based on cloud computing and implement a hybrid TaaS system based on VMs and virtual networks. This TaaS system integrates the IaaS platform and the SaaS platform to support the automatic and rapid creation of test environments and the remote use of test tools and online test services; in addition, this system provides service and management portals for easily managing test resources and test services. In addition to function demonstration, we evaluate this TaaS system experimentally in terms of the VM provision time, VM performance and virtual network performance and compare the experimental results with those from Alibaba Cloud and QingCloud platforms. The results demonstrate that our VMs and virtual networks perform well and can satisfy testers' basic demands. This TaaS system can provide low-cost resources and convenient test services and accelerate the testing process, thereby reducing testing costs and improving the testing efficiency. Furthermore, the proposed system can realize smooth operation and management, improving the maintenance efficiency. Therefore, this system is cost effective in practical software testing. However, our VM performance is not as good as those from Alibaba Cloud and QingCloud in most test items; hence, we must improve our system in future work.

References

[1] Q. Li, D. L. Ke, and X. L. Wang, "Brief survey on cloud testing," *Application Research of Computers*, vol. 29, no. 12, pp. 4401-4406, December, 2012. Article (CrossRef Link)

[2] J. Gao, X. Bai, W.-T. Tsai, "Testing as a service (TaaS) on clouds," in *Proc. of IEEE 7th International Symposium on Service-Oriented System Engineering (SOSE)*, pp. 212-223, March 25-28, 2013. Article (CrossRef Link)

[3] S. Jain, D. Srivastava, "Testing as a service (TaaS) on cloud: needs and challenges," International *Journal of Advanced Research in Computer Science & Technology*, vol. 2, no. 2, pp. 335-340, April-June, 2014. Article (CrossRef Link)

[4] A. Bertolino, L. Nautiyal and P. Malik, "Annotated buzzwords and key references for software testing in the cloud," in *Proc. of IEEE International Conference on Computing, Communication and Automation*, pp. 893-900, May 5-6, 2017. Article (CrossRef Link)

[5] M. Kaur, "Testing in the cloud: new challenges," in *Proc. of IEEE International* Conference *on Computing, Communication and Automation*, pp. 742-746, April 29-30, 2016. Article (CrossRef Link)

[6] P. Harikrishna and A. Amuthan, "A survey of testing as a service in cloud computing," in *Proc. of 2016 international conference on computer communication and informatics (ICCCI)*, pp. 1-5, January 7-9, 2016. Article (CrossRef Link)
[7] V. Mittal, L. Nautiyal and M. Mittal, "Cloud testing-the future of contemporary software testing," in *Proc. of International Conference on Next Generation Computing and Information Systems (ICNGCIS)*, pp. 131-136, December 11-12, 2017. Article (CrossRef Link)
[8] A. S. Sathe, R. Kulkarni, "Study of testing as a service (Taas) –cost effective framework for taas in cloud environment," *International Journal of Application or Innovation in Engineering & Management*, vol. 2, no. 5, pp. 240-243, May 2013. Article (CrossRef Link)
[9] A. Bertolino, A. Calabro, D. A. Guglielmo, et al, "When the testing gets tough, the tough get ElasTest," in *Proc. of 40th ACM/IEEE International Conference on Software Engineering (ICSE)*, pp. 17-20, May 27-June 3, 2018. Article (CrossRef Link)
[10] L. Riungu-Kalliosaari, O. Taipale, K. Smolander and I. Richardson, "Adoption and use of cloud-based testing in practice," *Software Quality Journal*, vol. 24, no. 2, pp. 337-364, October, 2016. Article (CrossRef Link)
[11] T. Banzai, H. Koizumi, R. Kanbayashi, T. Imada, T. Hanawa and M. Sato, "D-cloud: design of a software testing environment for reliable distributed systems using cloud computing technology," in *Proc. of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pp. 631-636, May 17-20, 2010. Article (CrossRef Link)
[12] A. Gambi, W. Hummer and S. Dustdar, "Automated testing of cloud-based elastic systems with AUToCLES," in *Proc. of 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 714-717, November 11-15, 2013. Article (CrossRef Link)
[13] W. T. Lo, R. K. Sheu, S. M. Yuan and G.-H. Luo, "Automatic test environment deployment and service testing model in the cloud," *Journal of Internet Technology*, vol. 17, no. 3, pp. 599-607, May, 2016. Article (CrossRef Link)
[14] L. Cao, Y. Jiang, C. M. Gan, Y. C. Zhang and G. Q. Chen, "Construction of software testing platform on cloud computing," *New Technology of Library and Information Service*, vol. 11, pp. 34-39, November, 2012. Article (CrossRef Link)
[15] C. Chiang, C. Chang, H. Chen, Y. Chen, S. Yuan and C. Wang, "ATP: a browser-based distributed testing service platform," *International Computer Symposium*, pp. 192-197, December 15-17, 2016. Article (CrossRef Link)
[16] M. Yan, H. Sun, X. Wang and X. Liu, "Building a TaaS platform for web service load testing," in *Proc. of IEEE International Conference on Cluster Computing*, pp. 576-579, September 24-28, 2012. Article (CrossRef Link)
[17] C. Li and H. Shih, "A cloud testing platform and its methods based on essential cloud characteristics," in *Proc. of International Conference on Machine Learning and Cybernetics (ICMLC)*, pp. 163-169, July 12-15, 2015. Article (CrossRef Link)
[18] X. P. Ding, H. Hou, S. He, Z. G. Chen and D. H. Guo, "Performance study of cloud testing platform based on openstack," *Software*, vol. 36, no. 1, pp. 6-10, March, 2015. Article (CrossRef Link)
[19] S. Lee, Y. Lin, K. Lin and J. You, "Composing and delivering heterogeneous web testing software as a composite web testing service," *International Computer Symposium (ICS)*, pp. 605-610, December 15-17, 2016. Article (CrossRef Link)
[20] X. Xu, H. Jin, S. Wu, L. Tang and Y. Wang, "URMG: enhanced CBMG-based method for automatically testing web applications in the cloud," *Tsinghua Science and Technology*, vol. 19, no. 1, pp. 65-75, February, 2014. Article (CrossRef Link)
[21] C. Tao and J. Gao, "Cloud-based mobile testing as a service," *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 1, pp. 147-152, 2016. Article (CrossRef Link)
[22] D. Tao, Z. Lin and C. Lu, "Cloud platform based automated security testing system for mobile internet," *Tsinghua Science and Technology*, vol. 20, no. 6, pp. 537-544, December, 2015. Article (CrossRef Link)

[23] C.-H. Liu, "A cloud platform for compatibility testing of android multimedia applications," in *Proc. of International Conference on Frontier Computing*, pp. 169-178, July 12-14, 2017. Article (CrossRef Link)
[24] C. Guo, S. Zhu, T. Wang and H. Wang, "FeT: hybrid cloud-based mobile bank application testing," in *Proc. of IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 21-26, July 16-20, 2018. Article (CrossRef Link)
[25] M. Ficco, R. Pietrantuono and S. Russo, "Hybrid simulation and test of vessel traffic systems on the cloud," *IEEE Access*, vol. 6, pp. 47273-47287, August, 2018. Article (CrossRef Link)
[26] H. Kim, A. Ahmad, J. Hwang and et al, "IoT-TaaS:towards a prospective IoT testing framework," *IEEE Access*, vol. 6, pp. 15480-15493, 2018. Article (CrossRef Link)
[27] W.-T. Tsai and G. Qi, "Integrated fault detection and test algebra for combinatorial testing in TaaS (Testing-as-a-Service)," *Simulation Modeling Practice and Theory*, vol. 68, pp. 108-124, November, 2016. Article (CrossRef Link)
[28] S. Zhang and B. Pi, "Mobile functional test on TaaS environment," in *Proc. of 9th IEEE International Symposium on Service-Oriented System Engineering (SOSE)*, pp. 315-320, March 30-April 3, 2015. Article (CrossRef Link)
[29] C. Tao and J. Gao, "On building a cloud-based mobile testing infrastructure service system," *Journal of System and Software*, vol. 124, pp. 39-55, February, 2017. Article (CrossRef Link)
[30] S. Ali and H. Li, "Moving Software Testing to the Cloud: An Adoption Assessment Model Based on Fuzzy Multi-Attribute Decision Making Algorithm," in *Proc. of 2019 IEEE 6th IEEE International Conference on Industrial Engineering and Applications (ICIEA)*, pp. 382-386, April 12-15, 2019. Article (CrossRef Link)
[31] A.-R. Eid, H. Aloran, M. Salah and et. al., "A mutation-based model to rank testing as a service (TaaS) providers in cloud computing," in *Proc. of the International Conference on Internet of things and Cloud Computing*, pp. 1-5, March 22-23, 2016. Article (CrossRef Link)
[32] R. R. Oliveira, R. M. Oliveira, A. S. Oliveira, "Impact of the Vendor Lock-in Problem on Testing as a Service (TaaS)," in *Proc. of 2017 IEEE International Conference on Cloud Engineering (ICCE)*, pp. 190-196, April 4-8, 2017. Article (CrossRef Link)
[33] Parasoft, "SOAtest," https://www.parasoft.com/products/soatest.
[34] Akamai, "CloudTest," https://www.akamai.com/cn/zh/products/performance/cloudtest.jsp.
[35] Sauce Labs, "Continuous Testing Cloud," https://saucelabs.com.
[36] Microsoft, "Xamarin Test Cloud," https://testcloud.xamarin.com.
[37] Alibaba, "MQC," http://mqc.aliyun.com.
[38] Tecent, "WeTest," https://wetest.qq.com.
[39] Baidu, "Baidu MTC," http://mtc.baidu.com.
[40] Testin, "cloud testing service platform," http://www.testin.cn.
[41] J. Chen, C. Wang, F. Liu and Y. Wang, "Research and implementation of a software online testing platform model based on cloud computing," in *Proc. of 5th International Conference on Advanced Cloud and Big Data*, pp. 87-93, August 13-16, 2017. Article (CrossRef Link)
[42] B. Kavitha and P. Varalakshmi, "Performance analysis of virtual machines and docker containers," in *Proc. of International Conference on Data Science Analytics and Applications*, pp. 99-113, January 4-6, 2018. Article (CrossRef Link)
[43] Openstack, "Open source software for creating private and public clouds," https://www.openstack.org.
[44] UnixBench, "A benchmark suite for unix-like systems," https://www.ostechnix.com/unixbench-benchmark-suite-unix-like-systems.

Jing Chen received the M.S. degree from Beijing Institute of Technology, and is currently working toward her Ph.D. degree in Shandong University of Science and Technology. She is also an Associate Research Fellow at Shandong Computer Science Center (National Supercomputer Center in Jinan). Her research interests include cloud computing and software testing.

Honghua Yan received the M.S. degree from Shandong University in 2014. He is currently a senior engineer at Shandong GreenPower Technology Co., Ltd. His current research interest focuses on automation of electric power systems.

Chunxiao Wang received the M.S. degree from Shandong University in 2004. She is currently an Associate Research Fellow at Shandong Computer Science Center (National Supercomputer Center in Jinan). Her Her research interests include big data analysis and artificial intelligence.

Xuyan Liu received the M.S. degree from Shandong University in 2003. He is currently an Engineer at China Telecom Corporation Limited, Shandong Branch company. His research interests include network transmission and network optimization.