International Journal of Informatics and Information System ISSN 2579-7069
Vol. 1, No. 2, December 2018, pp. 97-106

Server Structure Proposal and Automatic Verification Technology on IAAS Cloud of Plural Type Servers

Yoji Yamato [1,*]

[1] Software Innovation Center, NTT Corporation, Japan
[1] yamato.yoji@lab.ntt.co.jp*;
* corresponding author

Abstract

In this paper, we propose a server structure proposal and automatic performance verification technology which proposes and verifies an appropriate server structure on Infrastructure as a Service (IaaS) cloud with bare metal servers, container based virtual servers and virtual machines. Recently, cloud services have been progressed and providers provide not only virtual machines but also new metal servers and container based virtual servers. However, users need to design an appropriate server structure for their requirements based on 3 types quantitative performances and users need much technical knowledge to optimize their system performances. Therefore, we study a technology which satisfies users' performance requirements on these 3 types IaaS cloud. Firstly, we measure performances of a bare metal server, Docker containers, KVM (Kernel based Virtual Machine) virtual machines on OpenStack with virtual server number changing. Secondly, we propose a server structure proposal technology based on the measured quantitative data. A server structure proposal technology receives an abstract template of OpenStack Heat and function / performance requirements and then creates a concrete template with server specification information. Thirdly, we propose an automatic performance verification technology which executes necessary performance tests automatically on provisioned user environments according to the template.

Keywords: Performance; Cloud Computing; IaaS; Baremetal; Container; Hypervisor; OpenStack; Heat; Automatic Test;

1. Introduction

Infrastructure as a Service (IaaS) cloud services have advanced recently, and users can use virtual resources such as virtual servers, virtual networks and virtual routes on demand from IaaS service providers (for example, Rackspace public cloud [1]). Users can install OS and middleware such as DBMS, web servers, application servers and mail servers to virtual servers by themselves. And open source IaaS software also becomes major, adoptions of OpenStack [2] are increasing especially. Our company NTT group has also launched production IaaS services based on OpenStack since 2013 [3].

Most cloud services provide virtual computer resources for users by virtual machines on hypervisors such as Xen [4] and Kernel based Virtual Machine (KVM) [5]. However, hypervisors have demerits of much virtualization overhead. Therefore, some providers start to provide container based virtual servers (hereinafter, containers) which performance degradations are little and bare metal servers (hereinafter, baremetal) which does not virtualize a physical server.

Providing alternatives of bare metals, containers and virtual machines to users can enhance IaaS adoptions, we think. It is generally said that bare metals and containers show better performances than virtual machines but an appropriate usage is not mature based on 3 type servers quantitative performances. Therefore, when providers provide these 3 type servers naively, users need to design an appropriate server structure for their performance requirements and need much technical knowledge to optimize their system performances.

Therefore, we study a technology which satisfies users' performance requirements on these 3 types IaaS cloud in this paper. Firstly, we measure performances of a bare metal server provisioned by Ironic [6], Docker [7] containers, KVM virtual machines on OpenStack with virtual server number changing. Secondly, we propose a server structure proposal technology based on the measured quantitative data. In OpenStack, Heat [8] provisions virtual environments based on text format templates. A server structure proposal technology receives an abstract template of Heat and function/performance requirements and then creates a concrete template with server specification information.

Thirdly, we propose an automatic performance verification technology which executes necessary performance tests automatically on provisioned user environments according to the template.

The rest of this paper is organized as follows. In Section 2, we introduce an IaaS platform OpenStack, review bare metal, container and hypervisor technologies and clarify problems of providing these 3 type servers at the same time. In Section 3, we measure performances of these 3 type servers on OpenStack and discuss an appropriate usage. In Section 4, we propose a server structure proposal technology which satisfies users' requirements and an automatic performance verification technology which confirms performances on the provisioned environments. We compare our work to other related work in Section 5and summarize the paper in Section 6.

2. Overview of Existing technologies

2.1. Outline of OpenStack

OpenStack [2], CloudStack [9] and Amazon Web Services [10] are major IaaS platforms. The basic idea of our proposed technologies is independent from the IaaS platform. For the first step, however, we implement a prototype of the proposed technologies on OpenStack. Therefore, we use OpenStack as an example of an IaaS platform in this subsection. Note that functions of OpenStack are similar to other IaaS platforms.

OpenStack is composed of function blocks that manage each virtual resource and function blocks that integrate other function blocks. Fig.1 shows a diagram of OpenStack function blocks. Neutron manages virtual networks. OVS (Open Virtual Switch) [11] and other software switches can be used as virtual switches. Nova manages virtual servers. Hypervisors such as KVM usages are major but containers such as Docker containers and bare metal servers provisioned by Ironic also can be controllable. OpenStack provides two storage management function blocks: Cinder for block storage and Swift for object storage. Glance manages image files for virtual servers. Heat orchestrates these function blocks and provisions multiple virtual resources according to a template text file. Ceilometer is a monitoring function of virtual resource usage. Keystone is a function block that enables single sign-on authentication among other OpenStack function blocks. The functions of OpenStack are used through REST (Representational State Transfer) APIs. There is also a Web GUI called Horizon that uses the functions of OpenStack.

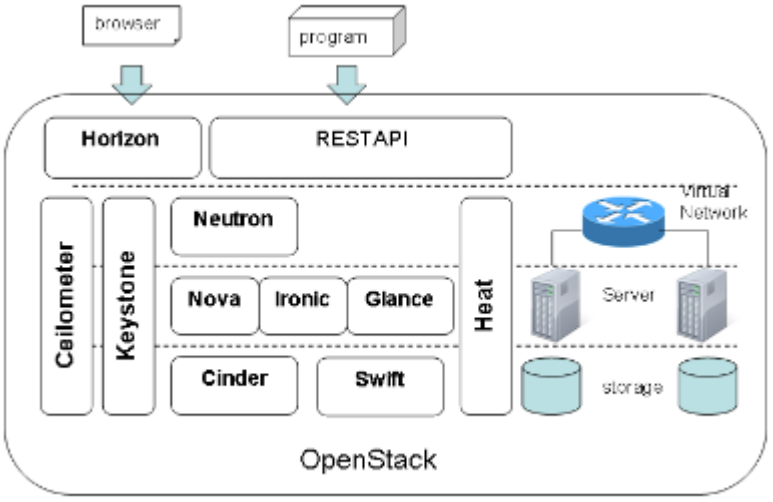


Figure. 1. OpenStack Architecture

2.2. Qualitative comparison of bare metal, container, hypervisor

Baremetal is a non-virtualized physical server and same as an existing dedicated hosting server. IBM SoftLayer provides bare metal cloud services adding characteristics of prompt provisioning and pay-per-use billing to dedicated servers. In OpenStack, Ironic component provides bare metal provisioning. Because baremetal is a dedicated server, flexibility and performance are high but provisioning and start-up time are long and it also cannot conduct live migrations.

Containers' technology is OS virtualization. OpenVZ [12] or FreeBSD jail were used for VPS (Virtual Private Server) [13] for many years. Computer resources are isolated with each unit called container but OS kernel is shared among all containers. Docker which uses LXC (Linux Container) appeared in 2013 and attracted many users because of its usability. Containers do not have kernel flexibility but a container creation only needs a process invocation and it takes a short time for start up. Virtualization overhead is also small. OpenVZ can conduct live migrations but Docker or LXC cannot conduct live migrations now.

Hypervisors' technology is hardware virtualization and virtual machines are behaved on emulated hardware, thus users can customize virtual machine OS flexibly. Major hypervisors are Xen, KVM and VMware ESX. Virtual machines have merits of flexible OS and live migrations but those have demerits of performances and start up time. Fig. 2 summarizes above descriptions. In section 3, we evaluate performance and start-up time quantitatively.

Table. 1. Qualitative comparison of bare metal, container and hypervisor

	Baremetal	Container	Hypervisor
Performance	Good	Fair	Bad
Start-up time	Bad	Good	Fair
Flexibility	Good	Fair	Good
Live migration	No	Open VZ: Yes, LXC/Docker: No	YES
Example	OpenStack Ironic, IBM SoftLayer	Docker, LXC, OpenVZ	Xen, KVM, Vmware ESX

2.3. Problems of plural types of IaaS server provisioning

Three type servers increase options of price and performance for users. It is generally said that new metals and containers show better performances than virtual machines on hypervisors. However, there are few works to compare performances and start up time of those three in the same conditions and appropriate usage discussions based on quantitative data are not mature. For example, [14] compared performances of baremetal, Docker and KVM but there is no data of performance with virtual server number changing. Therefore, when providers provide these 3 type servers naively, users need to select and design an appropriate server structure for their performance requirements and need much technical knowledge or performance evaluation efforts to optimize their system performances.

There are some works of resource arrangement on hosting / cloud services to use physical server resources effectively (for example, [15]), these technologies' targets are to reduce providers cost. In the other hand, a technology which selects appropriate type servers based on users' performance and cost requirements is not sufficient. Therefore, we study an appropriate type server proposal technology in Section 4 using quantitative performance data of Section 3.

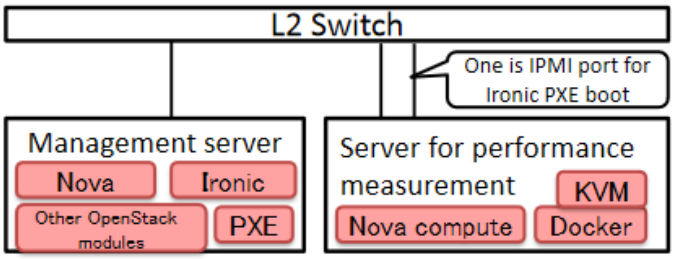
Note that a smooth migration among these 3 type servers is another problem. Live migrations cannot be done between different platforms, migrations need steps of image extraction and image deployment. For example, VMware provides a migration tool which helps a migration from other hypervisors to VMware ESX and it extracts images, converts images then deploys images [16]. In this paper, migrations are out of scope because we use these existing tools.

3. Comparison of Bare Metal, Docker and KVM Performance

Bagian ini mengukur kinerja dari 3 jenis server dengan kondisi yang sama. Kami menggunakan OpenStack versi Juno sebagai pengontrol cloud, server fisik yang disediakan oleh Ironic sebagai bare metal, Docker 1.4.1 sebagai teknologi container dan KVM / QEMU 2.0.0 sebagai hypervisor. Ironis, Docker dan KVM adalah perangkat lunak standar de facto di komunitas OpenStack. Contoh server adalah server Ubuntu 14.04Linux dan kami meminta 3 jenis contoh yang disediakan ke server fisik yang sama menggunakan OpenStack Nova.

3.1. Performance measurement items

- Measured servers: bare metal provisioned by Ironic, containers based on Docker, Virtual machines deployed on KVM.



Hardware	Main usage	CPU model name	CPU core#	RAM (GB)	HDD(GB)
MouseComputer H77MA-S40	Mamanegement server (OpenStack components and PXE server for	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz ×1	8	16	WD5000AAKX (SATA 500GB 7200rpm)
FUJITSU PRIMERGY RX200 S7/D3032-A1	Server for performance measurement	Intel(R) Xeon(R) CPU E5-2609 0 @ 2.40GHz ×2	8	64	MBE2147RC (SAS 147GB 15000rpm)

Figure. 3. Performance measurement servers' specifications

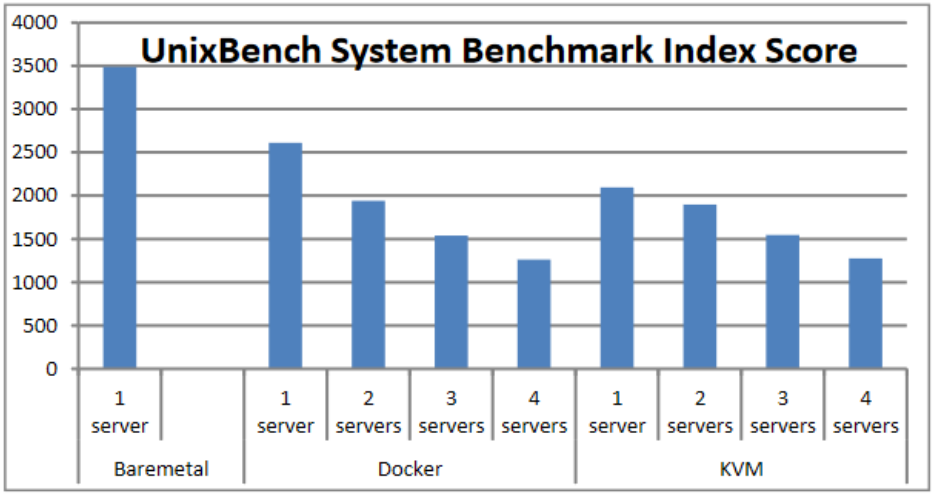


Figure. 4. UnixBench performance comparison

- Virtual server number: 1, 2, 3, 4

Only 1 for Bare Metal case, 1-4 containers for Docker case and 1-4 virtual machines for KVM case. When there are plural virtual servers, all physical resources are equally separated to these plural servers.

- Performance measurement

UnixBench [17] is conducted to acquire UnixBench performance indexes. Note that UnixBench is a major system performance benchmark.

3.2. Performance measurement environment

For a performance measurement environment, we prepared 1 physical server on which 3 types of servers were provisioned and 1 physical server which had OpenStack components (Nova, Ironic, PXE server for Ironic PXE boot and so on). These servers were connected with Gigabit Ethernet and Layer 2 switch. Fig. 3 shows each server specification.

3.3. Performance of Bare Metal, Docker and KVM

Fig. 4 shows a performance comparison of 3 type servers. Vertical axis shows UnixBench performance index value and horizon axis shows each server with virtual server number changing.

Based on Fig.4results, it is clear that Docker containers performance degradation is about 75% performance compared to Bare Metal performance. And it is also said that Docker performance is degraded when we change virtual server number but it is not inverse proportion. Meanwhile, virtual machines on KVM performance degradation is more larger and only 60% performance compared to Bare Metal performance and KVM performance degradation tendency with virtual server number change is as same as Docker.

3.4. Discussion

Here, we discuss appropriate usages of IaaS servers based on quantitative data. Because bare metal shows better performances than other 2 type servers, it is suitable to use large scale DB processing or real time processing which have performance problems when we use virtual machines. Containers lack flexibility of kernel but performance degradation is small and start up time is short. Thus, it is suitable for auto scaling for existing servers or shared usages of basic services such as Web or mail. Hypervisors are suitable to use for areas which need system flexibility such as business applications on specific OS.

4. Proposal of Performance Aware Server Structure Proposal and Automatic Performance Verification Technology

We propose a technology which enables a provider proposes an appropriate server structure and verifies it based on a users performance requirements in this section. In 4.A, we explain the steps of server structure proposal and automatic performance verification. The figure shows OpenStack, but OpenStack is not a precondition of the proposed method. In 4.B, we explain the process of server structure proposal using Section 3 performance data, which is one of core process of these steps. In 4.C, we explain the process of performance test extraction for each user environment, which is another core process of these steps.

4.1. Processing steps

Our proposed system is composed of Server structure proposals and Automatic Verification Functions (hereinafter SAFs), a test case DB, Jenkins and an IaaS controller such as OpenStack. Fig.5shows the processing steps of server structure proposal and automatic verification. All steps are 8.

1. A user specifies an abstract template and requirements to SAFs. A template is a JSON text file with virtual resource structure information and is used by OpenStack Heat [8] or Amazon CloudFormation [18] to provision virtual resources in one batch process. Although Heat template needs server flavor (= specification) information, an abstract template does not include flavor information. A template also describes image files for server deployments. Both providers' images and user original images can be used. A user also specifies each server function and performance requirements. Function requirements are that OS are normal Linux or non-Linux or customized Linux, and are used to judge if a container satisfies requirements. Performance requirements are server throughput or latency requirements. Note that if a user would like to replicate an existing virtual environment, we can use an technology of [19] to extract a template of the existing environment.

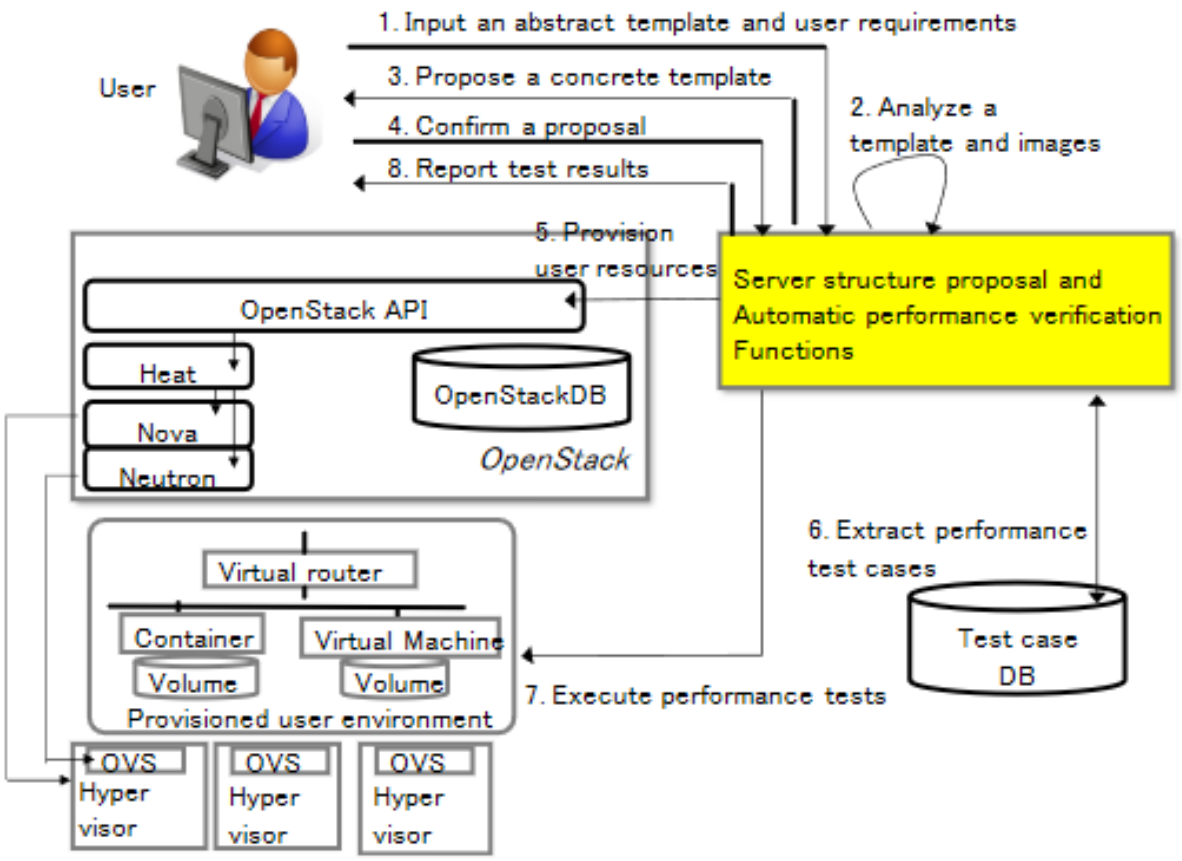


Figure. 5. Processing steps of the proposed method

2. SAFs understand server connection pattern and installed software from a template and image files specified by a user. If there is a user original image file, SAFs need to get information from a volume which is deployed by the image to understand what software is installed. In this case, a user needs to input login information in step 1. After analyzing a template and images, SAFs judge a system structure such as Fig. 6.
3. SAFs select server types and propose a server structure using user requirements specified in step 1. Because this is a first core step of the proposed method, we explain it in detail in 4.2. When SAFs propose a server structure, SAFs add a specific flavor for each server to Heat template. Thus, a user can distinct each server type as bare metal or container or virtual machine by flavor descriptions.
4. A user confirms the proposal and replies an acknowledgment to SAFs. After acknowledgment, SAFs fix a concrete template with each server flavor.
5. SAFs request an IaaS controller to deploy the concrete template with the target tenant. An IaaS controller provisions for virtual resources of the user environment on the specified tenant.
6. SAFs select appropriate performance verification test cases from the test case DB to show a sufficient performance of user environment provisioned based on the template. SAFs select test cases not only each individual server performance but also plural servers' performance such as transaction processing of Web 3-tier model. Because this is a second core step of the proposed method, we explain it in detail in 4.3.
7. SAFs execute performance test cases selected in Step 6. We use an existing tool, Jenkins [20], to execute test cases selected from the test case DB. Although performance verification is targeted for servers, verification test cases are executed for all virtual resources in a user environment. In a case where virtual machines with web servers are under one virtual load balancer, web server performances need to be tested via the virtual load balancer.

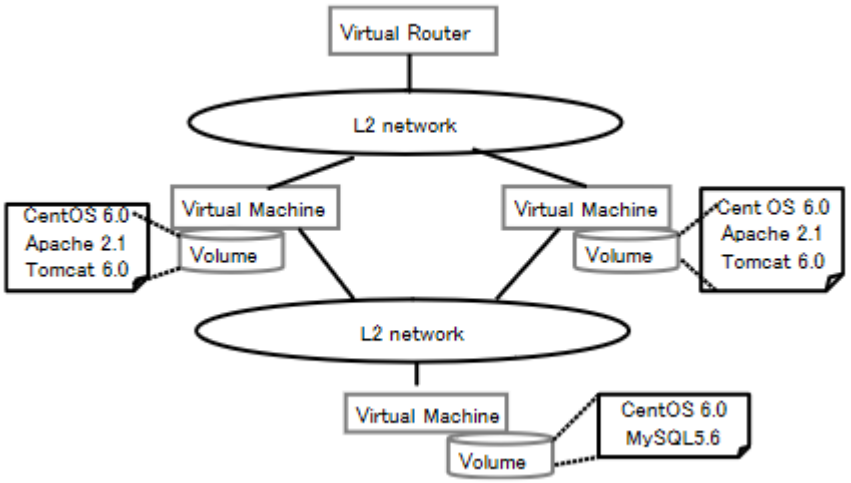


Figure. 6. Example of web 3-tier connection pattern

8. SAFs collect the results of test cases for each user environment using Jenkins functions. Collected data are sent to users via mail or Web. Users evaluate system performances by these data and start to use the IaaS cloud.

4.2. Server structure proposal technology

In this subsection, we explain in detail step 3 of the server structure proposal, which is a first core step of our proposal. SAFs understand server connection pattern and installed software from a template and image files specified by a user, then select server type from user function and performance requirements and propose an appropriate server structure.

Generally, server prices are container <virtual machine <bare metal. Therefore, the selection logic is that SAFs only select virtual machines or bare metals if containers cannot satisfy user requirements.

Firstly, SAFs select bare metals for servers which need high throughput and low latency. Throughput and latency thresholds are determined by Section 3 performance results. If user performance requirements specified in step 1 exceed thresholds, SAFs select bare metals. For example, because order management DB of Web shopping system needs strong consistency and is difficult for parallel processing, baremetal is appropriate when a system is above a certain scale. If a system does not require strong consistency and allows Eventual Consistency [21], a container or virtual machine become alternatives for a DB server because distributed Key-Values store such as memcached [22] can be adopted to enhance throughput.

Next, SAFs are narrow down server type by OS requirements. SAFs check function requirements whether a server OS is normal Linux or server OS is non-Linux / customized Linux, and select a virtual machine for latter case.

Lastly, SAFs select containers for servers which OS are normal Linux and are not uniform management servers. Fig. 7 shows a server selection logic flow of the proposed method.

4.3. Automatic performance verification technology

In this subsection, we explain in detail step 6 of performance test case extraction, which is a second core step of our proposal.

Authors developed an automatic patch verification technology for virtual machine patches previously [23]. A key idea of test case extractions of [23] is 2-tier software abstracting to reduce prepared test cases. [23] stores relations of software and software group which is a concept grouping different versions of software and function group which is a concept grouping same functions software, and it extracts test cases corresponding to the upper tier concept. For example, in case of MySQL 5.6 is installed on virtual machines, [23] method executes DB function group test cases and MySQL software group test cases. This idea has a merit for operators not to prepare each software regression test cases.

However, [23] can extract only unit regression tests because it selects test cases corresponding to each virtual server software. The problem is it cannot extract performance tests with plural virtual servers.

To enable performance tests with plural servers, we propose a performance test extraction method for each connection pattern of servers using information of Heat template connection relation and installed software.

Firstly, the proposed method stores software information in test case DB is not only [23] 's software relation information of Fig.8 (a) but also connection pattern information of Fig.8 (b). Here, Fig. 8 (b) second row shows that "connection pattern" is Web 3-tier and "deployment config" is {Web, AP} {DB}. A deployment config of {Web, AP} {DB} means one server has a Web server and an Application server and another server has a DB server. For example, connection relations like Fig. 6 can be analyzed by parsing a Heat JSON template description in step 2. Using connection relations of templates, installed software and Fig. 9 (a) software relation data, user server deployment configurations can be judged as { Web, AP} {DB}. Adding Fig. 8 (b) connection pattern information, a connection pattern can also be judged as Web 3-tier model.

(a)

Function group	Software group	Software
OS	Windows	Windows Server 2012
OS	Windows	Windows 8.1
OS	RHEL	RHEL 7.0
OS	RHEL	RHEL 6.1
DB	Oracle	Oracle11g
DB	Oracle	Oracle 10g
DB	MySQL	MySQL 5.0
DB	MySQL	MySQL 4.0
Web	Apache	Apache 2.1
Web	Apache	Apache 2.2
AP	Tomcat	Tomcat 6.0
AP	Tomcat	Tomcat 7.0

(b)

Connection Pattern	deployment config
Web 3-tier	[Web][AP][DB]
Web 3-tier	[Web, AP][DB]
Web 3-tier	[Web][AP, DB]
Web 3-tier	[Web, AP, DB]

Figure 8. (a) Software relation data, (b) Connection pattern data

Connection Pattern	Function group	Software group	Software
	DB		
	DB		
	DB	MySQL	
Web 3-tier			

Test case	Test case class	target subject
Table CRUD	DB function group	function
character garbling check	DB function group	data
Access by phpMyAdmin	MySQL software group	function
TPC-C benchmark test	Web 3-tier connection pattern	function

Figure. 9. Test case data

Next, the proposed method adds a "connection pattern" column to [23] 's test case information of Fig. 9 and enables to define test cases corresponding to each connection pattern. For example, Fig. 9fourth row shows that the TPC-C (Transaction Processing Performance Council benchmark) benchmark [24] test can be used for regression tests for the Web 3-tier connection pattern.

By these improvements, SAFs judge connection patterns by templates created in step 3 and installed software extracted in step 2. For example of Fig.6, Fig.8and Fig.9case, SAFs judge a connection pattern as Web 3-tier. Then, when SAFs extract test cases in step 6, those extract not only each Web server or DB performance test cases but also TPC-Ctest for Web 3-tier connection pattern.

5. Related Works

Like OpenStack, OpenNebula [25], Eucalyptus [26] and CloudStack [9] are open source Cloud software. OpenNebula is a virtual infrastructure manager of IaaS building. OpenNebula manages VM, storage, network of company and virtualizes system resources to provide Cloud services. Eucalyptus characteristic is an interoperability of Amazon EC2, and Xen, KVM or many hypervisors can be used on Eucalyptus. Our group also contributes to developments of OpenStack itself. Some bug fixes and enhancements of OpenStack are our group contributions.

The paper [27] is a research of dynamic resource allocation on OpenStack. There are some works of resource arrangement on hosting services to use physical server resources effectively [15] [28]. As same as [27], our work is also a resource arrangement technology on OpenStack but our work targets to resolve problems of appropriate server type selection from 3 type servers. There is no similar technology to propose an appropriate server structure on IaaS cloud with bare metals, containers and virtual machines.

The work of [14] compared performances of bare metal, Docker and KVM. However, there is no data of performance with virtual server number changing and appropriate usages discussions of 3 type servers are not mature. We measured performances of a new metal provisioned by Ironic, Docker containers and KVM virtual machines with same conditions and evaluated quantitatively.

Amazon CloudFormation [18] and OpenStack Heat [8] are major template deployment technologies on the IaaS Cloud. However, there is no work using these template deployment technologies for automatic performance verification of virtual servers because each user environment is different. We use Heat to provision user virtual environments by a concrete template and execute performance test cases automatically to show a guarantee of performance to users.

Some tools enable automatic tests, for example, Jenkins [20] and Selenium [29]. However, these tools are aimed at executing automatic regression tests during the software development life cycle, and there is no tool to extract performance test cases dynamically based on each user environment. The method proposed by Willmor and Embury is intended to generate automatic test cases of DB [30]. It needs the specifications of preconditions and postconditions for each DB test case. However, collecting user system specifications is impossible for IaaS virtual machine users. Our technology can select and execute performance tests automatically based on installed software and connection patterns of templates. For example, it selects and executes TPC-C benchmark when a user system structure is 3-tier Web.

6. Conclusion

In this paper, we proposed a server structure proposal and automatic performance verification technology which proposes and verifies an appropriate server structure on Infrastructure as a Service cloud with bare metals, containers and virtual machines. It receives an abstract template of Heat and function / performance requirements from users and selects appropriate servers.

Firstly, we measured UnixBench performances of a bare metal, Docker containers, KVM virtual machines controlled by OpenStack Nova to collect the necessary data of appropriate proposal. In the results, a Docker container showed about 75% performance compared to a bare metal but a KVM virtual machine shows about 60% performance. Secondly, we proposed a server structure proposal technology based on the measured data. It selected appropriate server types and created a concrete template using server OS flexibility requirements and performance requirements of uniform management servers. Thirdly, we proposed an automatic performance verification technology which executed necessary performance tests automatically on provisioned user environments according to the template. It selected a performance test case using information of connection patterns and installed software.

In the future, we will implement our method not only for OpenStack but also for other IaaS platforms such as CloudStack. We will also prepare sufficient number of performance test cases for actual use cases of IaaS virtual servers. Then, we will cooperate with IaaS Cloud service providers to provide managed services in which service providers propose appropriate server structures and guarantee performances.

References

[1] Rackspace public cloud powered by OpenStack web site, http://www.rackspace.com/cloud/

[2] OpenStack web site, http://www.openstack.org/

[3] Y. Yamato, Y. Nishizawa, M. Muroi and K. Tanaka, "Development of Resource Management Server for Carrier IaaS Services Based on OpenStack," Journal of Information Processing, Vol. 23, No.1, pp. 58-66, Jan. 2015.

[4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," In proceedings of the 19th ACM symposium on Operating Systems Principles (SOSP'03), pp. 164 -177, Oct. 2003.

[5] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori "kvm: the Linux virtual machine monitor," In OLS '07: The 2007 Ottawa Linux Symposium, pp. 225-230, July 2007.

[6] Ironic web site, https://wiki.openstack.org/wiki/Ironic

[7] Docker web site, https://www.docker.com/

[8] OpenStack Heat web site, https://wiki.openstack.org/wiki/Heat.

[9] CloudStack web site, http://CloudStack.apache.org/

[10] Amazon Elastic Compute Cloud web site, http://aws.amazon.com/ec2

[11] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado and S. Shenker, "Extending Networking into the Virtualization Layer," In Proceedings of the 8th ACM Workshop on Hot Topics in Networks (HotNets-VIII), Oct . 2009.

[12] OpenVZ web site. http://openvz.org/Main_Page

[13] P.-H. Kamp, and R.N.M. Watson, "Jails: Confining the Omnipotent root," In Proceedings of the 2nd International SANE Conference, May 2000.

[14] W. Fester, A. Ferreria, R. Rajamony and J. Rubio, "An Updated Performance Comparison of Virtual Machines and Linux Containers," IBM Research Report, July 2014.

[15] X. Liu, X. Zhu, P. Padala, Z. Wang, and S. Singhal, "Optimal Multivariate Control for Differentiated Services on a Shared Hosting Platform," In Proceedings of the IEEE Conference on Decision and Control, pp. 3792- 3799, 2007.

[16] VMware vCenter Converter web site, http://www.vmware.com/products/converter

[17] UnixBench web site, https://code.google.com/p/byte-unixbench/

[18] Amazon CloudFormation web site, http://aws.amazon.com/cloudformation/

[19] Y. Yamato, M. Muroi, K. Tanaka and M. Uchimura, "Development of Template Management Technology for Easy Deployment of Virtual Resources on OpenStack," Journal of Cloud Computing, Springer, DOI: 10.1186 / s13677-014-0007-3 , June 2014.

[20] Jenkins web site, http://jenkins-ci.org/

[21] W. Vogels, "Eventually Consistent," ACM Queue Vol. 6, No.6, pp. 14-19, Oct. 2008.

[22] B. Fitzpatrick, "Distributed caching with memcached," Linux Journal, Vol. 2004, Issue.124, pp.5, Aug. 2004.

[23] Y. Yamato, "Automatic verification technology of software patches for user virtual environments on IaaS cloud," Journal of Cloud Computing, Springer, Feb. 2015.

[24] TPC-Cweb site, http://www.tpc.org/tpcc/

[25] D. Milojicic, Ignacio M. Llorente, Ruben S. Montero, "OpenNebula: A Cloud Management Tool," IEEE Internet Computing, v.15 n.2, pp.11-14, Mar. 2011.

[26] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, D. Zagorodnov, "The Eucalyptus Open-source Cloud-computing System," In Proceedings of Cloud Computing and Its Applications, Oct . 2008.

[27] F. Wuhib, R. Stadler, and H. Lindgren, "Dynamic resource allocation with management objectives -Implementation for an OpenStack cloud," In Proceedings of Network and service management, 2012 8th international conference and 2012 workshop on systems virtualization management, pp.309 -315, Oct. 2012.

[28] B. Urgaonkar, P. Shenoy, and T. Roscoe, "Resource overbooking and application profiling in shared hosting platforms," Symp on Operating Systems Design and Implementation, pp. 239-254. ACM Press, 2002.

[29] Selenium web site, http://www.seleniumhq.org/

[30] D. Willmor and S. M. Embury, "An intentional approach to the specification of test cases for database applications," In proceedings of the 28th international conference on Software engineering, pp.102-111. ACM, 2006.