2016 IEEE Symposium on Service-Oriented System Engineering

Microservices Architecture based Cloudware Deployment Platform for Service Computing

Dong Guo, Wei Wang*, Guosun Zeng, Zerong Wei
Department of Computer Science and Technology, Tongji University, Shanghai 200092, China
Tongji Branch National Engineering & Technology Center of High Performance, Shanghai 200092, China
The Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 200092, China
*corresponding author: wwang@tongji.edu.cn

***Abstract*—With the rising of Cloud computing, evolution have occurred not only in datacenter, but also in software development, deployment, maintain and usage. How to build cloud platform for traditional software, and how to deliver cloud service to users are central research fields which will have a huge impact. In recent years, the development of microservice and container technology make software paradigm evolve towards Cloudware in cloud environment. Cloudware, which is based on service and supported by cloud platform, is an important method to cloudalize traditional software. It is also a significant way for software development, deployment, maintenance and usage in future cloud environment. Furthermore, it creates a completely new thought for software in cloud platform. In this paper, we proposed a new Cloudware PaaS platform based on microservice architecture and light weighted container technology. We can directly deploy traditional software which provides services to users by browser in this platform without any modification. By utilizing the microservice architecture, this platform has the characteristics of scalability, auto-deployment, disaster recovery and elastic configuration.**

Keywords—Cloud computing; Cloudware; Container; Micro-service; PaaS

I. Introduction

Cloud computing gradually comes into our life because of its rapid development. The technology not only promotes the progress of computer systems, but also leads the reform of methods on using software. Meanwhile, the thinking that using IT resources as services becomes prevailing, which creates a trend of XaaS [1]. Services become a key concept of cloud computing. Cloud computing models like IaaS, PaaS, and SaaS have been largely used and widely spread. On the other hand, with the mature of virtualization and container technology like Docker [3], we can make the whole system, from developing to operation, into a microservices architecture, namely an architectural style or a design model. This model advocates that applications should be divided into a series of micro services and each of that should focus on a single function, running in separate process, which makes the boundary clear and makes light communication mechanism possible. It's not a coincidence for the emergence of microservices architecture. It's a profound thinking on architecture model, development and operation when the traditional services architecture faces challenges in the Internet era [2].

Cloud computing and the thinking that all are services change our traditional views of software. We can easily make it a service using cloud computing technologies. With the continuous optimization of Internet environment, traditional software gradually transports to the cloud side. For users, browsers are the entrance of Internet. From the simple HTML to the new HTML 5, CSS 3 and Web OS technologies, the advance of browser technology lays solid foundations for transportation of software and Web access. Getting services through browsers will be an important method in the future. The concept of software go to web and cloud will also be an important software deliver paradigm [4].

In conclusion, using microservice environment built on cloud to develop and operate software and utilizing advanced Web technology to make software on web, is the trend of software development. The software will not be an entity of code but a series of services, which can be delivered to users through Internet. In this article, we call this new form of software *Cloudware*, which will be the key form of software in cloud environment.

To promote the Cloudware well, we propose a new PaaS platform for Cloudware, which base on microservice and container technology for scalability and on-demand service. It makes direct transportation of software in the Cloud in a convenient way, and users can get access to it through any browsers.

Section 2 introduces challenges and some related works. Section 3 presents a new paradigm of software in the Cloud. Section 4 illustrates details of our proposed Cloudware PaaS architecture. Section5 describe the experiments and result analysis of CloudwareHub, and Section 6 concludes this paper.

II. Challenges and Related Works

As cloud computing gains traction with enterprises and terminal users, many conventional applications are moving to the cloud. Cloud applications have some advantages over traditional non-cloud desktop applications: they can be less expensive, simpler to manage, and easier to update and use. On the other hand, there still reasons to keep your application on your local machine, including OS high tightly dependence, network bandwidth, et al.

Many applications that were created for the desktop are being migrated to the cloud. In some cases, the same application runs in the cloud environment, while others' are different. And there are new applications developed specifically to run on cloud infrastructure to let you utilize the advantages provided by the cloud. Different from the method mentioned above, this paper proposed a novel PaaS platform which can directly deploy traditional desktop applications to it, and terminal users can use them only by browsers. Our proposed PaaS platform acknowledges the ongoing information technologies, such as light weighted virtualization, microservice architecture and interactive remote rendering systems.

A. *Container*

System Hypervisors allow Multiple Guest OS to run together on a single machine. Each Guest OS abstracts Computation, Storage and Network Components. Hypervisor itself could run on bare-metal (ESXi) or be part of an OS (KVM). Guest ISA is translated to Host ISA using multiple techniques like Hardware Virtualization or Binary Translation.

A container is a light weight operating system running inside the host system, running instructions native to the core CPU, eliminating the need for instruction level emulation or just in time compilation. Containers provide savings in resource consumption without the overhead of virtualization, while also providing isolation. Rajdeep et al. [15] compares virtual machines and containers on multiple factors like performance, isolation security, networking, storage, and conclude that the usage of containers in PaaS is becoming mainstream, and the Linux containers are becoming a defacto standard but have some way to go before widespread adoption. Scheepers [16] also make a comparison between virtualization and containerization of application infrastructure by a marco-benchmark, and conclude that both hypervisor-based and container-based virtualization can provide portability, isolation and optimize the utilization of hardware resources, but container is a better option for a PaaS environment as well as in private clouds.

B. *Microservice architecture*

For many years now, we have been finding better ways to build systems. Eric Evans's book Domain-Driven Design [11] helped us understand the importance of representing the real world in our code, and showed us better ways to model our systems. Alistair Cockburn's concept of hexagonal architecture [12] guided us away from layered architectures where business logic could hide. Virtualization platforms allowed us to provision and resize our machines at will, with infrastructure automation giving us a way to handle these machines at scale [13]. Some large, successful organizations like Amazon and Google espoused the view of small teams owning the full lifecycle of their services. And, more recently, Netflix has shared with us ways of building antifragile systems at a scale that would have been hard to comprehend just 10 years ago.

Many organizations have found that by embracing fine-grained, microservice architectures, they can deliver software faster and embrace newer technologies. Microservices give us significantly more freedom to react and make different decisions, allowing us to respond faster to the inevitable change that impacts all of us [14].

C. *Interactive Remote Rendering Systems*

Remote rendering has been receiving increasing attention from researchers in both academia and industry in recent years due to the proliferation of cloud computing and mobile devices.

Compared to the conventional approach that performs rendering locally, remote rendering has several advantages. First, it can provide rich rendering experiences to " thin" clients (e.g., mobile devices) with limited computing resources (i.e., GPU, memory, power). Second, the computing resources on a rendering server can be efficiently shared by multiple clients. Third, remote rendering is a simple but effective cross platform solution. After developing the client programs for different platforms, all applications need only be developed for the server and the same rendering experience will be achieved on all client platforms. Last but not least, a remote rendering system can be designed to prevent the source contents from leaking to malicious users by streaming only rendering results to clients. Shi and Hsu [17] survey the interactive remote rendering systems in an excellent way, analyze how to improve the state of the art, and summarize the related technologies. The readers of this article will understand the history of remote rendering systems and obtain some inspirations of the future research directions in this area.

III. CLOUDWARE: A NEW PARADIGM OF SOFTWARE ON THE CLOUD

With the mature of cloud computing, cloud platform like Amazon EC2 and Google App Engine are used more and more to cut down cost. Nowadays, developers are the majority of using cloud services. Furthermore, the majority of cloud services are IaaS and PaaS. Thus, users of SaaS can only get access to cloud service by utilizing software on cloud. With the introduction of microservice and so forth, the software on cloud is not a simple entity of code, but a series of services which offered online. Those software become more and more adjusted for the cloud and gradually become a new form which we called "Cloudware" in this article.

Cloudware is not a software, but a *software paradigm* under cloud environment. Traditional software is dependent of OS and the libraries. With the wide spread of cloud, however, software should be developed and run more adapted to the cloud, and it will be reformed with cloud. In next part, we will describe developing model, deployment and running of Cloudware in detail.

A. *The developing model of Cloudware*

When developing traditional software, developers should build the corresponding environment like IDE or Compilers. After the emergence of Git and Task managing system, development of Cloudware should be on the cloud. Developers can utilize cooperated software to trace the task and "cloudize" the process, both from the coding and software engineering aspects.

Meanwhile, developing Cloudware should obey the microservice thinking, which indicates that software should be divided into distinct constructions as far as possible, and be encapsulated into service form. The deconstruction can be done by multiplex through API interface. And it is also convenient for testing. Especially for the teamwork, container like Docker can offer multiplexed environment, flexible resources configuration and convenient methods for testing. In the process of development, the call of function differs from traditional way which call from libraries of OS. Instead, it call from microservice. Cloudware development should be faced to microservice constructions, but not to specific OS or hardware.

B. *The deployment of Cloudware*

The deployment of Cloudware is the deployment of microservice. Nowadays, container technology like Docker becomes more and more developed. Docker provides a series container deployment tools which offer developers a new, convenient methods for arranging tests [3].

The deployment of Cloudware should be released as services. We can deploy different constructions both in separate or integrated ways to provide the compatible service model and to keep Cloudware running. This is the fundamental need of cloud services.

C. *The running of Cloudware*

Cloudware are designed as service form. In fact, running a Cloudware is running integrated microservice. The difference between Cloudware and traditional software is that the main body of Cloudware is running on the cloud while that on the local. Thus the interaction between Cloudware and users is the key problem of Cloudware, especially for the desktop one.

Because of its running on cloud, which means that computation and storage are also on cloud, users only need an environment for interaction. In recent years, the "weblization" of software is a trend. Cloudware can use local explorers as constructions that provide interaction services. Users can get services through one explorer and they do not need to build the environment which is necessary for running software like GL, JDK and so forth, namely, a client independent model.

The essence of explorer interaction is an input-output visible process. The only thing we need to do is to send the input from mouse or keyboard to the servers on cloud and send the outcome back to the local [5], which can get almost the same effect of local software. The details are shown as Figure1. The cloud service is mainly constructed by three independent microservices: X broker, X Server and Application which can be further divided. Application and Xserver communicate by X11 protocol and send request for rendering to Xserver. It will send mouse and keyboard event to Application while Xserver keep the status of Application and make it synchronous on the local explorers. In this way, users can utilize Cloudware like local software. The difference is that they use explorers as windows manager. And the status of working can be immediately recovered, which is independent of local OS and status.

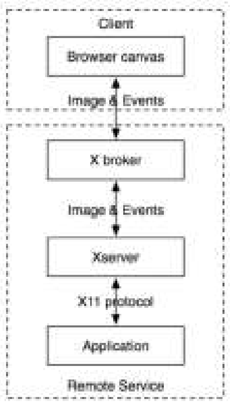

Fig. 1 Interactive style for Cloudware

IV. MICROSERVICES ARCHITECTURE BASED CLOUDWARE PAAS PLATFORM

After we make the concept of Cloudware clear, we develop a PaaS platform for Cloudware deployment and application: CloudwareHub. It is an integrated platform for developing, testing, deploying and running of Cloudware. It offer developing tools and environment to developers and services for users. CloudwareHub has realized main functions for testing Cloudware and offering usually developing tools.

A. *The architecture of Cloudware platform*

The architecture of Cloudware PaaS platform is shown as Figure.2.

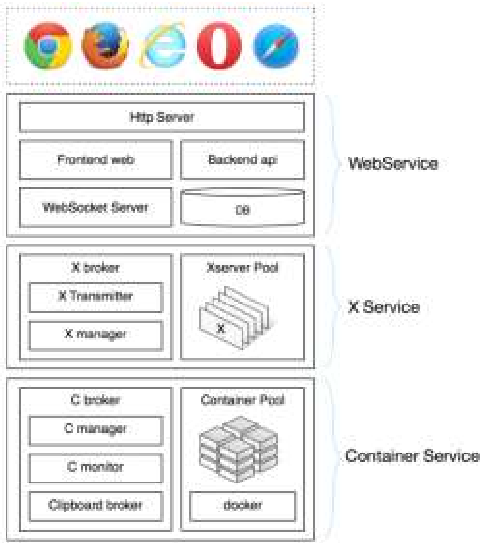

Fig. 2 Overall architecture of Cloudware PaaS platform

From Figure 2 we can see that three main parts make up the whole CloudwareHub: Container Service, X Service and Web Service. Container Service provide environment of applications. Each application is encapsulated as an image of

Docker, which will get storage in Container Pool. C broker is responsible for managing the lifetime of different containers and for communicating with X Service; X Service provides Xserver services, and communicates with both Container Service upward through X11 protocol and Web Service downward through TCP protocol. X broker is responsible for Xserver's lifetime; Web Service provides interaction and data service , which can be divided into two independent parts: Frontend web and Backend API. The Frontend web will call interface which is in the form of Restful, supported by the Backend API. Data interaction is done by Backend API and database. Moreover, because interactions of applications have its own status while Http does not, the WebSocket [7] Server in Web Service realizes the communications between X Service and explorers.

B. *The control plain and data plain of Cloudware PaaS*

The communication process of different services of CloudwareHub is shown as Figure 3.

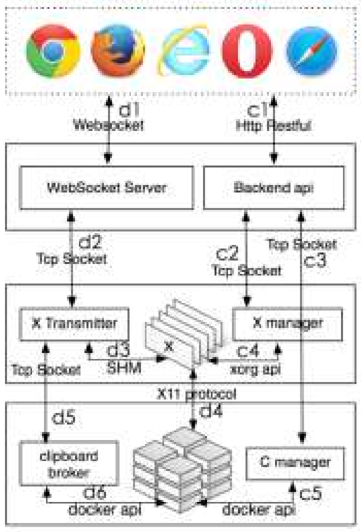

Fig. 3 Control plane and data plane

As the figure shown, c1-c5 is the tunnel of control plain, and d1-d6 is that of data plain. The main goal of control plain is to control the lifetime of Cloudware such as running and restart. C1 is responsible for calling Backend API for users which is based on Http Restful API [6]. For CloudwareHub, the main request includes X Service requests (events like adjust resolution or keyboard) and Container Service requests (events like running application and so forth). It communicates with X manager and C manager through c2 and c3 that is based on TCP. X manager controls Xorg [8] programs through libraries like Xlib or XCB, while C manager controls Docker by using API offered by Docker and corresponding language binding.

The data plain mainly focus on data flow in interaction level. D1 is responsible for communications between explorers and Websocket Server. It sends input from keyboard and mouse to Websocket Server through Websocket protocol and Websocket Server send output from Cloudware to the display. D2, which is based on TCP protocol, is the interaction between Websocket Server and X Service. It sends events and images to X Transmitter for further process.

D3 is interaction between X Transmitter and Xorg. X Transmitter sends events form mouse and keyboard to Xorg, using libraries like XCB and Xtest extension, meanwhile send changed images from caches which have been compressed to Websocket Server. And Websocket Server send it to the terminal display. For faster getting images, X Transmitter shares its memory With Xorg to get corresponding bit images. D4 is the data tunnel between X11 client and Xorg, which is realized by X11 protocol. D5 is the data tunnel of clipboard extension, which is used to realize the data copy between client and Cloudware. Data of client can be sent through d5 to clipboard broker, finally send to the clipboard caches of corresponding Cloudware through d6.

C. *The microservice running environment of Cloudware*

CloudwareHub offers microservice running environment. Thus, to realize the large scale of deployment, disaster recovery and flexible configuration, we should deploy and run the system as the form of microservice on IaaS cloud system. The original system of CloudwareHub runs on two cloud hosts, which is configured with one cores, 2GB memory and 20GB hard disk. Web Service runs on one of the cloud hosts which can be accessed by domain name “cloudwarehub.com”. Its configuration is shown as Figure 4.

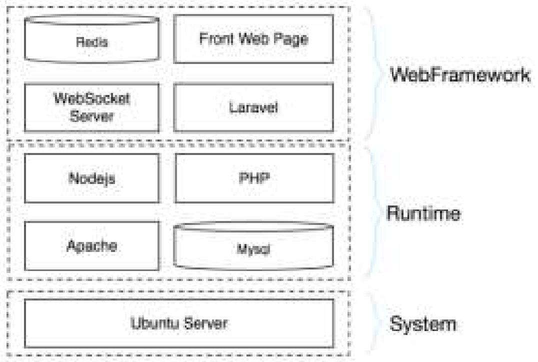

Fig. 4 The architecture of the Web service runtime environment

X Service and Container Service run on the other cloud host. The reason is that we want to make the communication more effective and to reduce the delay of internet. Its main architecture is shown as Figure 5.

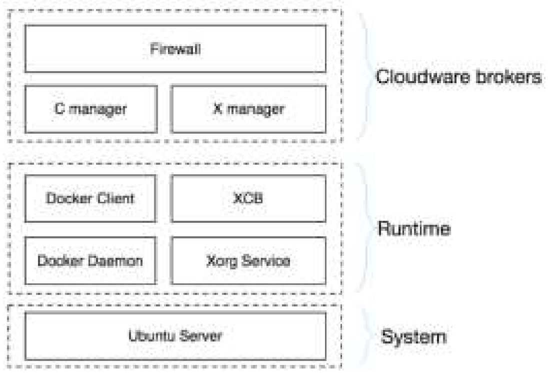

Fig. 5 X Service and Container Service runtime environment

Container Service is built under Linux, Ubuntu Server 14.04, also configured with Docker tools, Xorg drives and programming libraries. C manager and X manager make up the Cloudware brokers. It manages Xorg and Docker downward, and provides TCP interface outward.

D. *Advantages of Cloudware PaaS*

From above, we can see that because of using the microservice structure, Cloudware PaaS decomposes the application which is originally single to a series of single and focusing microservice by function boundary. Every microservice is corresponding to a component in traditional application, which can be separately compiled, deployed and extended. Compared with single structure, Cloudware PaaS has some advantages.

1) Complexity Controlling: When decompose every application, we delaminate the great complexity. Every microservice focuses on single function such as single Cloudware example, and define clearly the boundary by well-designed interfaces. Because of low complexity, every microservice can be controlled by a small team, which is easy to maintain and development. And through this we can support developer to develop more and more Cloudware examples.

2) Deployment independence: Because microservice has its own process, it can be deployed independently. There is no need for compiling or deploying the whole application when it changed. The application composed by microservice is equal as a series of distributed process, which led to more efficient deployment and less risk of the developing environment, finally to the less period of delivering.

3) Flexibility: Under microservice structure, technology is anti-centralized. Every team can choose their own technology type corresponding to their demand and requirement. Because each Cloudware example is relatively easy, it is possible for completely reconstructing a Cloudware when its risk is on a relatively low level.

4) Failures tolerance: When failures occurs on some components, they may become larger in the single structure, which will led to the failures of the whole system. Under the microservice structure, failures will only separately occurs in single Cloudware service. If with a good design, other Cloudware can tolerant failures by some restart and degeneration mechanism.

5) Extendibility: Single structure application can be extended transverse. Namely we can copy it to other nodes. When difference exist in distinct components, every Cloudware can extend separately according to the requirement because of the flexibility of microservice structure.

Although with many advantages, we must admit that it is not easy to construct, deploy and maintain the distributed microservice system. The Cloudware PaaS platform based on microservice utilizes light and application intended virtualized environment supported by container and offers ideal environment. Similarly, the cloud service which is based on container will largely make it easy for constructing, deploying and maintaining the whole process of containerize cloud service, and for putting forward the large scale of practices of Cloudware.

V. Experience of the CloudwareHub

Because Cloudware is based on internet, its performance largely related with the internet environment. The instability influents the experience of Cloudware more on the WAN than the LAN. To test the experience of Cloudware, we deploy CloudwareHub [10] on UCloud [9] and offer service like Eclipse.

The interface of the CloudwareHub system can be shown as Figure 6.

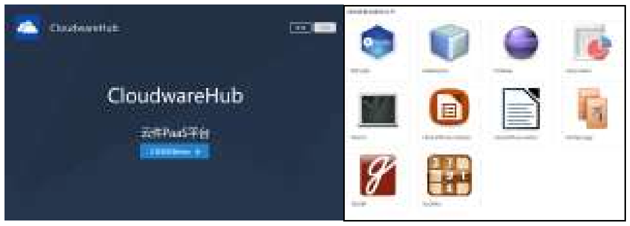

Fig.6 The main interface of CloudwareHub

Furthermore, we will evaluate the experience performance which is supported by CloudwareHub through some experiments.

We firstly test the average delay from typing the character to its showing on display in distinct internet environment, the rendered data is a 9x10 pix map which has been compressed in PNG format with size about 80Byte, the data size is small, and the delay is the key press event delay, we use this test case for testing the event delay. The outcome is shown as Figure 7.

We can get conclusion from the Figure 7 that with a very small data size, the delay is almost fixed. The data size and bandwidth do not contribute a lot to the delay. The main reason is the process of TCP communication and that of memory copy operations.

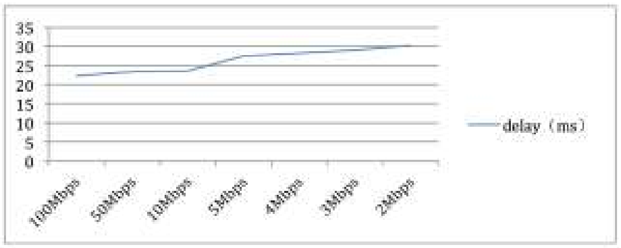

Fig. 7 The network latency(ms) of event of the CloudwareHub

Secondly, to test the effect of data size with different server bandwidth on experience, we choose some area of Eclipse, which has typical size of damaged PNG data, and measures the delay from mouse clicking on it to rendering image in browser canvas. To get the average delay, we tested 20 times for each size of data to get the average loopback latency. The outcome is shown as Figure 8. These result shows that the data size has some impact on the delay, but not varies with its size linearly. Since the main time consuming operations is memory copy operations and data compressing as well as TCP stack processing.

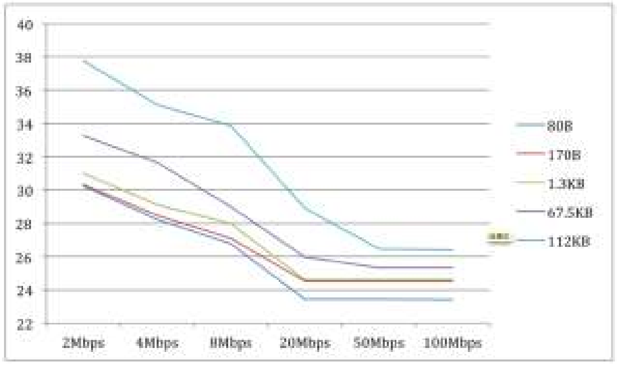

Fig. 8 The network latency (ms) of different data size with different server bandwidth of the CloudwareHub

Thirdly, to test the latency when client has a very low network speed, we make a limitation on our client router with different bandwidth limit from 10Kbps to 1Mbps, and we set the server bandwidth to 10Mbps to reduce the impact of server bandwidth. The result is shown as Figure.9. From Figure 9 we can see that when client has a very poor network, the network will has main impact on the experience, even data, which size larger than 10KB, will lead to timeout exception, so it is not shown in Figure 9.

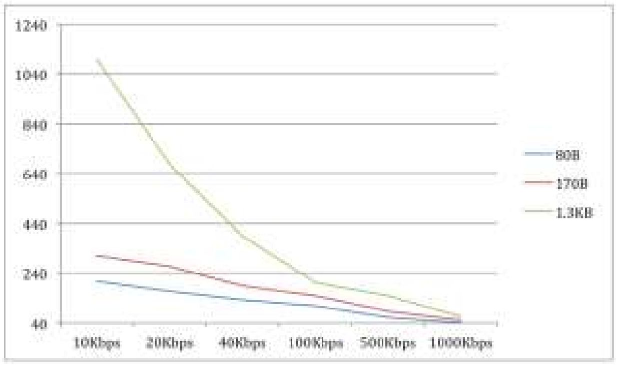

Fig. 9 The network latency (ms) of different data size with different client bandwidth of the CloudwareHub

As we can see from the results above, the experience of Cloudware is influenced by many conditions, and we can learn from the result that data compressing and bandwidth are key points to reduce latency, and try to make a cache system for CloudwareHub will improve user experience greatly. But in most cases, client has an unstable network, how to fit the network environment dynamically is important. We can learn that keeping stable experience under circumstance like fixed network delay and unstable network environment is a key potential factor for future research.

VI. Conclusion

The development of cloud computing not only leads the reform of data center, but also impacts the traditional methods on developing, deploying and running software, and further on the way of using it. Under this circumstance, deploying software as Cloudware will be the main stream. The thinking that cloud is service will promote the microservice design model and make it easy to be adjusted for cloud. From the users' perspective, Cloudware is the trend, which can directly offer services to them. We believe that Cloudware will be the main application and developing model in the future.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61272107, 61202173), the Program of Shanghai Academic/Technology Research Leader (15XD1503600), the Open Project Program of the State Key Laboratory of Computer Architecture (Institute of Computing Technology, Chinese Academy of Sciences, CARCH201408).

References

[1] Serrano N., G. Gorka, H. Josune, Infrastructure as a Service and Cloud Technologies, IEEE Software, 2015, 32(2): 30-36.

[2] Stefan W., Eddy T., Wouter J., Comparing PaaS offerings in light of SaaS development, Computing, 2014, 96(8): 669-724.

[3] Boettiger C. An introduction to Docker for reproducible research[J]. ACM SIGOPS Operating Systems Review, 2015, 49:71-79.

[4] Mei H., Huang G., and Xie T., Internetware: A Software Paradigm for Internet Computing, IEEE Computer, 2012, 45(6): 42-47.

[5] Quax P., Liesenborgs J., Barzan A., et al. Remote rendering solutions using web technologies[J]. Multimedia Tools and Applications, 2015: 1-28.

[6] Richardson L, Ruby S. RESTful Web Services[B]. O'reilly Media Inc, 2007.

[7] Bansal N, Harchol-Balter M, Schroeder B. Size-Based Scheduling to Improve Web Performance.[J]. ACM Transactions on Computer Systems, 2003, 21(2): 207-233.

[8] https://wiki.archlinux.org/index.php/Xorg

[9] Ucloud. http://www.ucloud.cn [OL], 2015.

[10] CloudwareHub. http://www.cloudwarehub.com [OL], 2015

[11] Eric E., Martin F., Domain-Driven Design: Tackling Complexity in the Heart of Software, Addison-Wesley, 2003.

[12] http://alistair.cockburn.us/Hexagonal+architecture

[13] Tseitlin A., The antifragile organization, Communications of the ACM, 2013, 56(8): 40-44.

[14] Newman S., Building Microservices, O'Reilly Media, 2015.

[15] D.Rajdeep, R. Reddy, R.Reddy, Virtualization vs Containerization to support PaaS, In Procedings of the IEEE International Conference on Cloud Engineering (IC2E'14), Boston, MA, 2014.

[16] M. J. Scheepers, Virtualization and Containerization of Application Infrastructure A Comparison, In Procedings of the 21st Twente Student Conference on IT June 23rd, 2014, Enschede, The Netherlands, 2014.

[17] S. Shi C. H. Hsu, A Survey of Interactive Remote Rendering Systems, Computing Surveys, 2015, 47(4): 57-85.