Towards Cloudware Paradigm for Cloud Computing

Dong Guo[1, 2], Wei Wang[1, 2]*, Jingxuan Zhang[1, 2], Guosun Zeng[1, 2], Qiao Xiang[3], Zerong Wei[1, 2]

1. Department of Computer Science and Technology, Tongji University, Shanghai 200092, China
2. The Key Laboratory of Embedded System and Service Computing, Ministry of Education, Shanghai 200092, China
3. Department of Computer Science, Yale University

* corresponding author: wwang@tongji.edu.cn

***Abstract*—The rise of cloud computing and the Internet not only bring change on the data center, but also lead to transformation in software development, deployment, operation and maintenance. With the continuous improvement of the current cloud computing and the internet environment, how to make better use of cloud computing platform, and how to serve the users is a popular field of computer software is a big challenge. In recent years, with the further development of concepts like micro-services and containers, software will further step forward to the Cloudware. This paper discusses how to deploy Cloudware in cloud environment, and proposes a new method to construct the PaaS platform which can directly deploy software on the cloud without any modification, while achieving a new model by the browser services. By using micro-service architecture, we achieving good performance of extension, scalable deployment, faults tolerance and flexible configuration. Finally, we evaluate this method by constructing a complete framework and carrying out an interactive delay experiment that directly focuses on users' experience, which also shows the effectiveness of this method.**

Keywords—Cloud computing; Cloudware; Container; User experience

I. INTRODUCTION

The rapid development of cloud computing and Internet technology not only promotes the progress of computer software and hardware, but also makes people change the way of using software. The concept of cloud computing makes the Internet technology reach its climax, and a lot of traditional industries are in the transition to the Internet mode to improve social efficiency, and further affecting people's work and life [1]. In this environment, the traditional software deployment paradigm will also requires transition.

How such a transition should proceed is currently an important research direction of software evolution and software engineering. First, the idea of IT services as resources is becoming more and more popular, showing the trend of everything as a service (XaaS). Users can easily enjoy the different levels of software as a service through networks, and such services have become the essence and core concept of cloud computing. IaaS, PaaS and SaaS are some representative service models, which have been widely studied and deployed [2]. At the same time, with the continuous deepening of virtualization and container technology, e.g., Docker[3], on behalf of the containers, gradually infiltrated into cloud computing at all levels of the system from the development to the operation and maintenance of the whole process. The micro services architecture has become an architectural style and design patterns. It advocates the mode to split the application into a series of small services, in which each service focuses on a single business function, running in a separate process, thus making the service operating with clear boundaries. Light communication mechanisms can also be adopted, such as HTTP/RESTful, to meet the needs of business and users. Micro-service, as an architecture model of transformation, is not accidental. It is a reflection of the architecture model, development, operation, and the maintenance methodology.

On the other hand, with the continuous optimization of the network environment, especially the rising of 4G/LTE related wireless communication technology, it is easier for users to gain high-speed access to the Internet[4], making it easy for traditional software gradually migrating to the cloud. For users, the browser is the main entrance to the Internet and browser technology is also in rapid development, from the simple analysis of the HTML file to the new HTML 5, CSS 3[5] and web OS[6] etc. This provides a solid foundation for software in cloud migration and web access. Access to software services through the browser will be an important direction of future software development, and the cloudlization of software will also become one of the important forms of software in the future.

In summary, cloud computing provides environment for micro services to build a software and the runtime environment. Meanwhile, utilizing the advanced Internet technology to realize web-based software will be the direction of software development and the tendency. In the cloud, the software will no longer be a simple code entity, but a series of services delivered to the users through the Internet. In this paper, we refer this paradigm of Internet plus software as *Cloudware*. We believe it will become the main form of future software paradigm.

In order to verify the feasibility of Cloudware, this paper proposes and develops a PaaS platform based on the micro service architecture and container technology. In this platform, the traditional software can be directly deployed on the cloud. And users can connect to the cloud through the Internet technology, using the browser to interact, achieving the *Internet + software* completely.

Section 2 introduces the concept and features of Cloudware. Section 3 illustrates the details of our proposed Cloudware PaaS architecture. Section 4 describe a QoE testing framework for the Cloudware PaaS platform. Section5 describe the experiments and result analysis of our method, and Section 6 concludes this paper.

2159-6190/16 $31.00 © 2016 IEEE
DOI 10.1109/CLOUD.2016.29

II. Basic concept of Cloudware

Software are everywhere nowadays. With the development of cloud computing and virtualization, more and more software tend to be deployed on cloud, making it free for the local resource. This is actually a form of service, and we call this diagram of software Cloudware. It is a specific form of SaaS, but the idea is different in that Cloudware relocates the local OS and runtime environment to the cloud without modification, and the users can communicate with it through a unified interacted platform.

Definition 1: *Cloudware* is a software paradigm suiting for scalability and on-demand service under cloud environment, which makes direct transportation of software in the cloud in a convenient way, and users can get access to it through any browsers.

A. Features of Cloudware

To be specific, Cloudware has the following features:

Running on the cloud. All the traditional software on desktops and the necessary resource are deployed on the cloud.

Flexible resource configuration. Cloudware can adjust its using of resource at any time based on the type of the service, like offering multi-cores for computing-intensive tasks.

Rendering on the cloud. Cloudware renders the graphic tasks on the cloud and send the outcome back to the local users. The process is independent on the local hardware.

Fast boot. The time for starting a Cloudware should be short enough, and it is possible to boot large software in seconds.

Interaction with internet. All the services are interacted by internet, making sure that all the data and runtime status are saved on cloud. The only thing for terminals to do is to set catch system.

Unified platform. Cloudware need a unified platform to communicate with each other on any local terminal, making sure that different hardware can run the same Cloudware.

The transparent communication. Because of all the data and status are saved on cloud, a transparent communication system should be built to connect different terminal file system.

B. Key techniques in Cloudware

In the early time of cloud computing, the characteristics we proposed above are almost impossible. However, with the rising of the container technology, GPU virtualization technology, 4G network technology and HTML 5 front-end interactive technology, achieving such functions becomes possible. In particular, the functionality of Cloudware depends on the following related technologies:

Virtualization. Virtualization can provide a virtual running environment on the cloud, offering a different platform for the cloud operating system, its dependent libraries and components services.

Cloud interactive rendering technology. Cloud interactive rendering technology is putting the rendering processes one the cloud, which will generate the RGBA image coding for streaming data format. The data then transmitted to the terminal through network and decoded directly after the process. Terminal interaction events such as mouse and keyboard events are sent to the cloud through network to realize the interaction between clouds rendering, which is shown as Figure 1.

Container. Container is a popular lightweight virtualization technology in recent years, which launches a mirrored instances within a few millimeters and occupies only a few additional resources. The container can be achieved by applying the rapid deployment with the local desktop software. At the same time, using containers such as Docker can easily realize micro service, and further improving the cloud deployment flexibility.

Media stream data compression. In order to improve users' experience, to reduce the delay of interaction as far as possible, and to ensure remote rendering output frame quality, the real-time interaction and streaming media data compression are necessary. Techniques such as widely used H.264[9], H.265[10] and Webm[11], is the key technology to improve the cloud interaction user experience.

Terminal interaction. The Cloudware mainly running on the cloud, thus the terminal only needs a unified interactive platform. The best choice to achieve this is to use the browser, which can adapt to the different platforms. At the same time, with the development of HTML5, CSS3 and other technologies, browser's processing and interactive capabilities has greatly expanded, laying a solid foundation for the construction of a unified interactive platform for Cloudware.

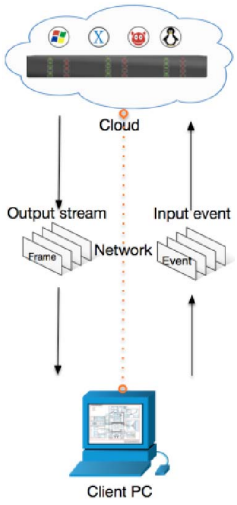

Fig. 1 Rendering interactive architecture

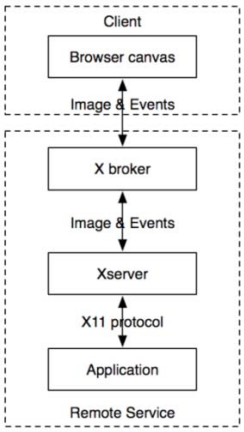

Fig. 2 Interactive principle of Cloudware

C. The development, deployment and operation mode of Cloudware

Development. In traditional software development process, developers need to build their own corresponding software development environment, such as IDE and compiler tool chain. With the rising of GIT and task management systems,

Cloudware development can follow cloud development processes, which using cloud based IDE and compiler to complete the entire development services. At the same time, the cloud collaborative software is used to carry out the tracking task, from the view of code writing and software engineering to cloudalize the software development process.

Cloud development should also follow the concept of micro-service software style, which is to divide the software into different components, and to make it convenient for testing process of continuous integration. The container, like Docker, can provide a reusable operating environment, flexible resource allocation and convenient integration test method especially for the teamwork. In the development process of Cloudware, calls of the function is no longer like traditional software as the library operating system calls, but to the micro service. Cloudware development should be oriented to micro service component form, and should not depend on the specific operating system and hardware architecture.

Deployment. The deployment of Cloudware is actually micro service deployment. Currently Docker, as the representative of the micro service container technology, is becoming more and more mature. Docker provides a series of container deployment tools for developers to carry out a novel and convenient software integration test method.

The cloud deployment should be carried out as the form of service, which means that different component can be deployed separately, providing downward compatible service. This ensures the continuous operation which is a basic idea of cloud computing.

Operation mode. Cloudware reflects as micro service in the design process. The difference between Cloudware and traditional software is that the body of Cloudware runs on the cloud, and that of traditional software on the client. Thus for the Cloudware, how to interact with users is a key problem here, especially for the desktop software. Because the main body of the cloud is running on the cloud, with computing and storage on the cloud server, the client only needs an interactive environment. In recent years, software Weblization is a kind of trend of software evolution. Cloudware can be regarded as a browser to provide interactive services components, but its operation is on the client side. Thus the client is only dependent on the browser, and it does not need to install GL, JDK, .Net, and other similar runtime.

Browser interaction is actually an input and output visualization process, which only need things like mouse and keyboard. Input is sent to the cloud server and return to the client for rendering [5], which can achieve the same effect as the local software. The specific interaction process is shown in Figure 2.

Cloud services mainly consists of three separate micro-services constitute: X broker, Xserver and Application. Application can also be constituted by a plurality of additional micro-services. Application and Xserver communicate through X11 protocol, which will render the request to Xserver. Then Xserver will send mouse and keyboard events to the application, maintaining a window state synchronized to the client's browser and achieving browser and Xserver event and rendering synchronization.

In this way, users can use the same software as using local Cloudware, but they do it with the browser window manager. At anytime, anywhere it can return to the last state, without depending on the client's operating system and running state.

III. CLOUDWARE PAAS PLATFORM ARCHITECTURE

In order to verify the technical conditions on the current cloud, this paper presents a Cloudware of PaaS platform based on micro-service architecture. It is a platform for the Cloudware development, testing, deployment and maintenance, both for developers and users. The Cloudware PaaS platform overview architecture is shown in Figure 3.

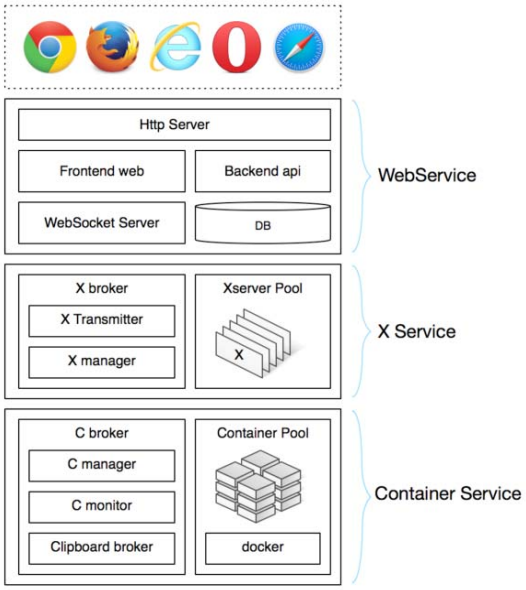

Fig. 3 The overall architecture of Cloudware

A. The architecture of Cloudware PaaS

From Figure 3, we can see that three main parts make up the whole Cloudware PaaS platform architecture: 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 plane and data plane

The communication process of different services of Cloudware PaaS platform is shown as Figure 4.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 Cloudware PaaS platform, 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.

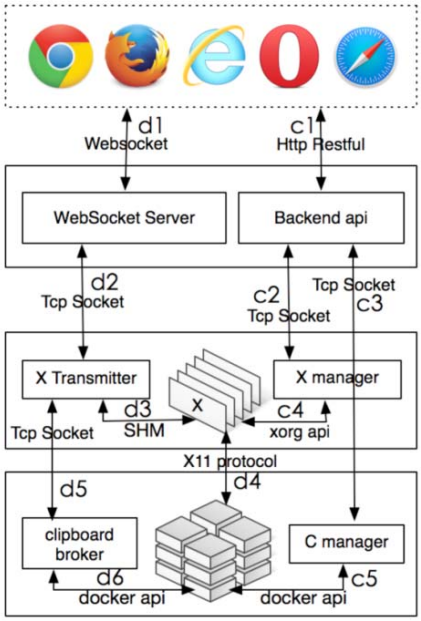

Fig. 4 Control plain and data plain

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. Internal protocols

Since Cloudware Hub itself has taken the design of micro-service architecture, the communication problems between micro-services is the key to the achieve service portfolio. To achieve high efficiency under loosely coupled communication, this paper design many sets of communication protocols between Cloudware components, including CDP (Cloudware Display Protocol) and Webftp agreement as shown in Figure 5.

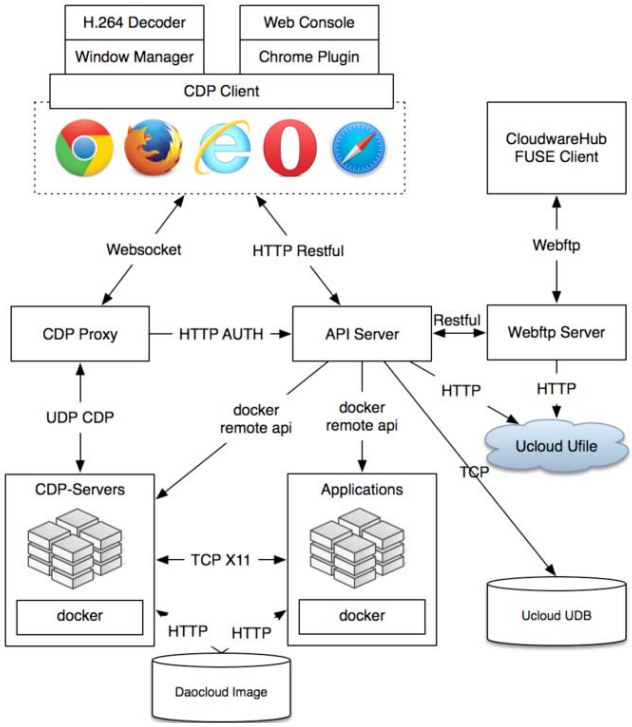

Fig. 5 Cloudware micro-service communication protocol

Different services communicate with each other mainly through TCP, UDP, HTTP, and Websocket communication protocol. TCP protocol is mainly used for two-way communication services internally; UDP protocol is used to transmit H.264 compression; HTTP protocol is used to access the API and external files; Finally Websocket agreement is used to implement two-way communication process between browser and server.

In the whole protocol architecture, CDP protocol and Webftp agreement is special for the Cloudware, where CDP-compatible binary protocol is a set of cloud-piece display protocol X11 protocol, whose main task is to carry out UDP encapsulation, and Webftp agreement is set of file transfer protocol running on the Websocket protocol, achieving transparent access to the cloud and local system [14].

D. The advantages of Cloudware PaaS

Due to the adoption of the micro-service architecture, the Cloudware decomposes a single application into a set of focused services, each of which is according to a component

with the function of independent compilation, deployment and extension. Cloudware PaaS platform has some advantages when compared with the single architecture:

The limited complexity. The decomposition in the application reduces the complexity of the original endless accumulation. Each micro-services focus on a single function, and articulate the service boundary through well-defined interfaces. Due to small size and low complexity, each micro-services can be controlled by a small development team, which is easy to maintain.

Independent deployment. Since the micro-services have an independent running processes, each micro-services can also be deployed independently. When a cloud service are changed without compiling, there is no need to deploy the entire application. Composed by the micro-service application making it more efficient and with less risks, ultimately shorten the delivery cycles.

Flexible technology selection. By micro-services architecture, technology selection is decentralized. Each team can make their decision according to the status and development of its own service industry, having the freedom to choose the most suitable tools for the development. Since each Cloudware is relatively simple and low-risk when you need to upgrade the face, or even completely reconstructing it is also feasible.

Faults tolerance. When a failure occurs in the traditional architecture of a single process, it is likely result in diffusion, making the global application crashed. In the micro-services architecture, the fault will be isolated in a single Cloudware service. If designed well, other members can retry to achieve application-level fault tolerance.

Scalability. Single application architecture can achieve scale by copying the entire application to a different node. When different components of Cloudware elements are different in expansion needs, micro-service architecture will reflect its flexibility because each Cloudware can be expanded independently

While the micro-services architecture brings many advantages, it must be admitted that to build, to deploy and to maintain a distributed micro-services system is not easy. Based on the proposed pieces of PaaS cloud micro-services, the light containers provided by application-oriented virtualized runtime environment provide an ideal environment for the micro-services. Similarly, the cloud services based on container technology will greatly simplify the integration, deployment, operation and maintenance of the entire process, so as to promote large-scale Cloudware element in the cloud-based practice.

IV. QoE Test Framework and User Experience Model

In order to evaluate and optimize the Quality of Experience (QoE) of the Cloudware systems, we must be able to measure the performance of Cloudware elements. Performance analysis for Cloudware systems mainly have following challenges: 1) objectivity, some of the traditional performance measurement methods only concerned part of the system, not the performance of the overall objective response; 2) accuracy, some measurements are usually performed to measure system performance, but not considering the user's sensory experiences.

Similar performance evaluation method focused on the study of virtual desktop infrastructure (VDI) system [16], but it is always poor at the users' experience because of too many factors affecting its performance. Previous studies focus on thin client [17, 18]. It cannot be applied to the proposed cloud member systems because of differences in application scenarios.

To solve this problem, and to get more objective and more accurate reflection from the users, this section starts with the Cloudware client, modeling the user experience and putting forward a kind of new test methods for the cloud element system performance. The method uses means of an user behavior control language, simulating directly on the client. This paper also use this method to build a Cloudware performance testing framework and to test the Cloudware client performance. Through a number of qualitative and quantitative analysis, the method will be proved to be valid and feasible.

A. Simulation of user behaviors and collection of system status

This paper presents a User Motion Controlling Language (UMCL) reference by the concept of distributed computing load simulation, generating Desktop Input controlling stream to simulate user behavior. This is divided into four parts: the application task definition, mouse smoothing, random thinking delay and operation repeatability. The user's behavior consists of a sequence of application tasks, and each event may be a mouse movement, keyboard input, etc. Mouse events are the most important Cloudware system events. The definition of mouse events are of two kinds: moving operation and key operation. Moving operation sends screen coordinate point to the mouse device file, so the mouse can move to the specified location on the screen. According to *Fitt's Law*, the users' moving mouse, the target object, the time it takes to move from the T and D and the target's size S has the following relationship:

$$T = A \log(\frac{D}{S} + 1) \quad (1)$$

A is a constant number. Thus we have the relationship between D and T:

$$D = \mathrm{e}^{\frac{T}{A}} - 1 \quad (2)$$

On the other hand, during the execution of the application, this article also monitors the network and the resource utilization of Cloudware system. These data reflect the real-time status of Cloudware systems, and they will indirectly reflect the client performance. For the information, the test framework can examine the following three parts: network monitoring, system monitoring, system resource utilization and hardware related information monitoring.

Network Monitoring is the main method for measuring Cloudware system performance. In this test, the network traffic

is monitored by a proxy agent. The flow agent monitors two things through socket proxy: monitoring the flow generated by the protocol transmission, and the operation type of analytical protocol.

Monitor system resource utilization is mainly responsible for the collection of present status, which is a commonly used measurement of system performance. Resource utilization monitoring the following three points: CPU utilization, memory utilization, and I / O access request (including disk read and write conditions and network literacy situation). This article also records the system resource utilization of both client and server hosts. These information requested by the timing controller program to a shorter interval periodic feedback from the host to the controller. The controller behavior of data recording is to ensure the consistency of the clock interval. This ensures that the state information on different hosts can be simultaneously requested. Resource monitoring service uses a Restful architecture on the client operating system and server operating system, which are open to provide resources to monitor the XML-RPC service, and then timed XML-RPC requests issued by the monitoring thread controller, obtaining system resources utilization of data. This design has a good scalability because developer can easily monitor service through RPC resources.

B. Interactive influence model

In order to modeling user interaction events, this paper models the Cloudware systems based on task driven and respond time. Here are the explanation of these two basic model.

Task driven modeling. This paper argues that in the computer interactive system, user's behavior is driven by tasks. It is also applicable in Cloudware systems. To build this model, we first introduce the following concepts:

Application: we consider user's behavior is always made up by a series of application session. We call the complete session, namely from the start to the end of a program, an Application.

Task: a logical related action series in a session is called Task. Application is always made up by tasks that can be divided into simpler task.

Motion: The smallest task is made up by actions that cannot be divided, which is according to a direct event, including mouse moving, mouse clicking and keyboard event, what we call a motion.

Thinking delay: the time used by user on thinking is called thinking delay.

Reaction delay: when doing motion, the intervals between these motions are called reaction delay.

The model should satisfied the following two principle: 1) Operations of different Applications should be logical and time independent. 2) The reaction time of users are always fixed. We can see that Application is always self-closed. After a session of an Application, the desktop status should be the same as before so that the operation independence can be ensured. Every Application forms a tree in task context, whose root is the session of Application and the node is a task. We call it an Application Task Tree. Figure 6 show such tree of a web task.

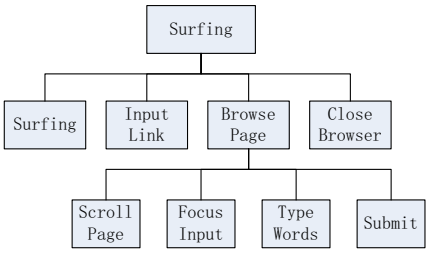

Fig. 6 Web task tree

This tree shows a session called Surfing which do a series of simple task: starting the browser, opening the link, browsing the web, submitting the table and finally closing the browser.

Reaction time modeling. This need to record the time point of interactive event. The whole process is shown as Figure 7.

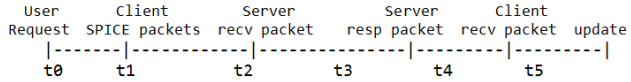

Fig. 7 User interaction process

$t_0 \sim t_1$ and t_4, t_5 is the dealing time of client; $t_1 \sim t_2$ and $t_3 \sim t_4$ is the transmission time; t_2 and t_3 is computing time of servers. The reaction time is a key factor affecting users' experience, which can be calculated as follow:

$$Latency = t_5 - t_0 \tag{3}$$

The model above can only describe a simple user interaction, like mouse double-clicking and so on. In fact, a user's event request will generate a request series on server, as shown in Figure 8.

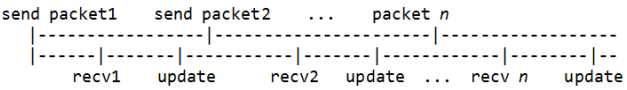

Fig. 8 Server interaction process

In this case, only calculating Latency is meaningless. We should monitor the following three indexes: The frequency of screen updates (FPS) that reflect the smoothing, the longest delay between two images(LatencyMax) that reflect the waiting time and the total update latency (LatencyTotal) that reflect the total waiting time. The model above is not good enough in all the cases, but it is satisfied in performance testing.

V. Performance Evaluation

This section uses the testing framework built on the above section to test the Cloudware platform for the actual user behavior simulation and we carry out statistical analysis and modeling based on these results, the cloud file system delay interactive performance analysis. Finally we analyze the delay performance of interactive events.

A. Running environment of Cloudware PaaS

Cloudware PaaS platform itself provide micro services environment. To achieve scalable deployment, faults tolerance and flexible configuration, we base our system on IaaS. To test the validity, we make our Cloudware PaaS Platform prototype system, namely CloudwareHub, run on two cloud hosts with two cores, 4GB memory and 20GB disk. Web Service runs on one of them and can be accessed by www.cloudwarehub.com.

X Service and Container Service runs on another hosts to improve the efficiency of communication between Container and Xorg, as shown in Figure 9.

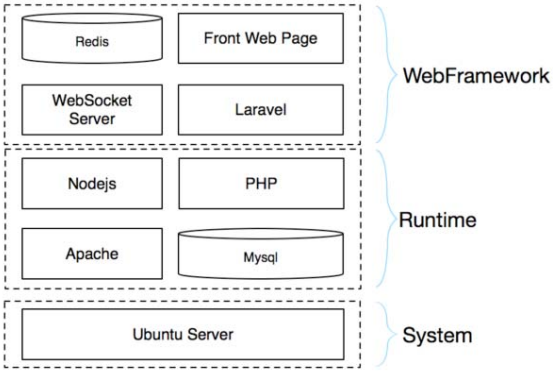

Fig. 9 Web Service runtime architecture

Container Service is based on Linux with Ubuntu Server14.04, Docker tools, Xorg devices and programming libraries. Parts of Cloudware brokers consist of C manager and X manager, managing the Xorg and Docker internally and providing TCP externally. All the systems above are deployed on Ucloud[15], as shown in Figure 10.

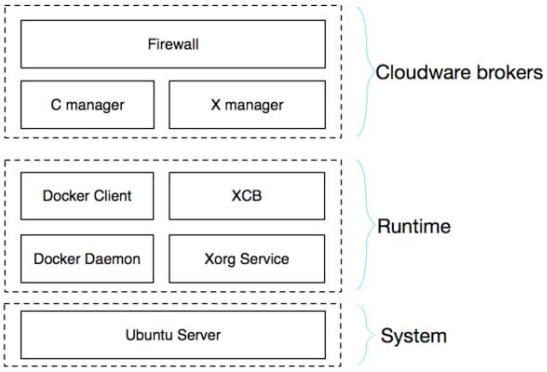

Fig. 10 X Service and **Container** Service runtime

B. Experiments Evaluation Framework based on UMCL

The test process based on the above ideas can be divided into three parts: user behavior modeling, experiment data collecting and quantity analysis of user experience.

In a conventional Cloudware system, the client's behavior can be abstracted into independent application tasks sequence. Due to mutual independence between applications, we can model user behaviors separately for different types of application and can characterize the performance of the main Metrics. This section examines several common scenarios of Cloudware: Web browsing Cloudware, video player Cloudware and text editor Cloudware. These three application are of the largest proportion among users choices. Therefore, we test them to get description of user behaviors. We use UMCL method proposed in Section.4 to describe different types of applications, as shown in Figure 11 which represents a text edit task tree structure.

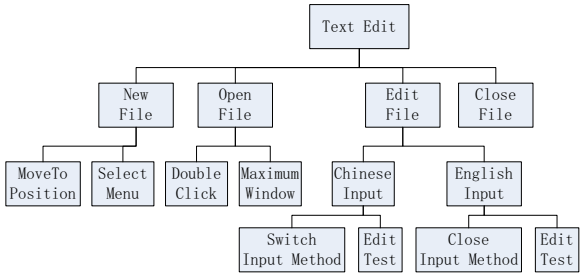

Fig. 11 Text edit task tree structure

With the UMCL tools, after testing user behavior, the paper collected for incident response time for statistics proposal. The main elements of the statistical process is to record the response time. By analyzing it, we may learn the response time variety made by the Cloudware system.

C. Experiments Study of Interactive delay

To test the experience of Cloudware, we carry out the interaction delay experiment in this section. Because of the fact that Cloudware is based on the internet, which means that the internet environment is highly related with the experience. Bad internet environment will lead in poor user experience.

We record the average delay from the click of keyboard to the complete rendering to browser, as shown in Figure 12.

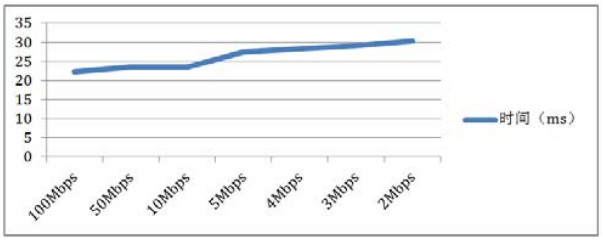

Fig. 12 Delay in different bandwidth

We can see that the delay is fixed when the amount of data is not big and TCP contribute most to the delay because of the memory copy. The delay can be ignored when the bandwidth is larger than 5Mbps.

Meanwhile, we choose five Cloudware with different rendering complexity to test the average delay including *Gedit*, *Eclipse*, *Rstudio*, *Libreoffice Impress* and *Supertuxkart*, getting the average frame size in one minute, as shown in Figure 13. Figure 13 shows that rendering complexity contributes to the delay, but not the lineal relations primarily because the rendering time and the delay caused by data compression on the cloud do not lineally related with the final frames. But in a

bigger context, delay will be larger with more complex render process.

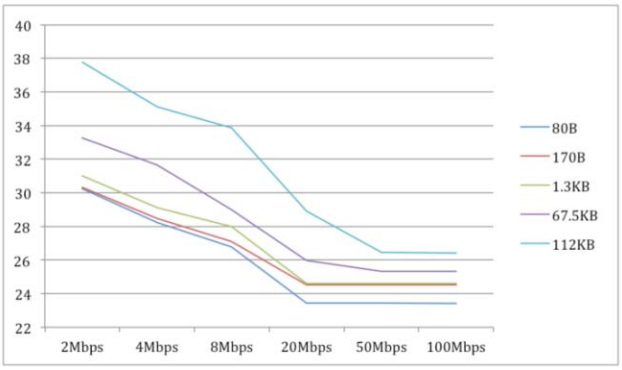

Fig. 13 Interactive delay(ms) in different rendering complexity and different server network

Therefore, we further test the delay of rendering in different terminal internet bandwidth. We choose three Cloudware as Gedit, Eclipse and Rstudio, shown as in Figure 14.

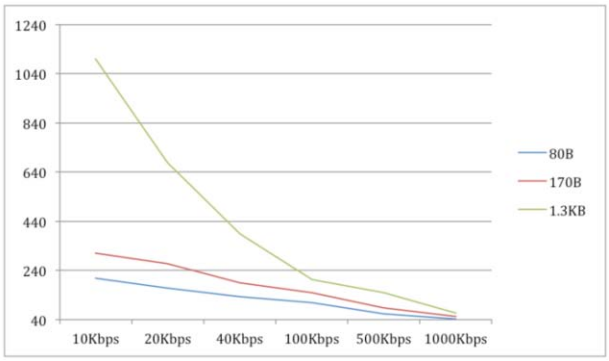

Fig. 14 Interactive delay(ms) in different rendering complexity and different terminal network

Figure 14 shows that the internet environment of terminal affect the user experience a lot. A bad internet environment and large amount of data will lead in TCP delay.

We can see that the user experience is decided by multiple factors, and the key is to improve internet quality and to reduce rendering data transmission. How to solve the original delay and keep a satisfied user experience is the key direction of future research.

VI. Conclusion

The development of cloud computing not only leads the reformation of data center, but also affects 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 micro-service 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.

VII. Acknowledgments

This work was supported by 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] Mei H., Huang G., Xie C. T. Internetware: A Software Paradigm for Internet Computing, IEEE Computer[J]. 2012, 45(6): 26-31.

[6] Vahdat A., Anderson T., Dahlin M., et al. Webos: Operating System Services For Wide Area Applications[J]. Proceedings of the Seventh IEEE Symposium on High Performance Distributed Systems, 1997:52 - 63.

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

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

[9] Gary J. Sullivan, , Jens-Rainer Ohm, , Woo-Jin Han, et al. Overview of the High Efficiency Video Coding (HEVC) Standard [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2012, 22(12): 1649 - 1668.

[10] Lu Y, Zhang Q, Wei B. Real-time CPU based H.265/HEVC encoding solution with x86 platform technology[C]// Computing, Networking and Communications (ICNC), 2015 International Conference on. IEEE, 2015: 418 - 421.

[11] Bankoski J. Intro to WebM.[C]// International Workshop on Network & Operating Systems Support for Digital Audio & Video. ACM, 2011: 1-2.

[12] 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.

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

[14] FUSE. https://github.com/libfuse/main [OL], 2015. 9.

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

[16] Wang J, Liang. Survey of Virtual Desktop Infrastructure System, IEFT, 2011.

[17] Albert M. Lai , Jason Nieh, On the Performance of Wide-Area Thin-Client Computing[J], ACM Transactions on Computer Systems, 2006, 24(2): 175–209.

[18] Nieh J, Yang S J, and Novik N. Measuring thin-client performance using slow-motion benchmarking[J]. ACM Transactions on Computer Systems, 2003, 21: 87–115.