Cloudware: An Emerging Software Paradigm for Cloud Computing

Dong Guo[+×] Wei Wang[+*] Jingxuan Zhang[+] Qiao Xiang[+#] Chenxi Huang[+]
Jinda Chang[+] Liqing Zhang[+]

+Tongji University #Yale University ×Cloudware Labs

* corresponding author: wwang@tongji.edu.cn

ABSTRACT

Software paradigm is a driving force for the evolution of software technology. With the continuous improvement in the current cloud computing and the Internet environment, software will develop further into Cloudware, which is emerging as a new software paradigm. This paper defines the concept of Cloudware, and discusses it in the context of software paradigm. Then, based on a loosely coupled von Neumann computing model, we propose a new method of constructing a Cloudware PaaS system which can directly deploy software into the cloud without any modification. By using micro-service architecture, we can achieve high performance, scalable deployment, faults tolerance and flexible configuration. Finally, we evaluate this method by carrying out an interactive delay experiment that directly focuses on users' experience, which shows the effectiveness of our method.

Keywords

Cloud computing; Cloudware; Software paradigm; Micro-service

1. INTRODUCTION

Software is a computer program that models the problem space of the real world as well as its solution. Software paradigm describes a software model and its construction theory about the perspective of software engineers or programmers [1, 2]. It acts as the core driven force the evolution of software technology, and always pursues to better utilize underlying hardware capabilities or runtime features.

With the rapid development of computer science and technology, the application domain and runtime environment evolve as well. Especially, 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. In this environment, the traditional software paradigm also requires a transition.

Internetware '16, September 18 2016, Beijing, China.

DOI: http://dx.doi.org/10.1145/2993717.2993718

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 are 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 [3, 4]. At the same time, with the continuous deepening of virtualization and container technology, e.g., Docker[5], 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 for 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 on 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 5G wireless communication technology, it is easier for users to gain high-speed access to the Internet, making it easy for traditional software gradually migrating to the cloud. For end 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 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. It will become the main form of future software paradigm [15].

In order to verify the feasibility of Cloudware paradigm, this paper first discusses the Cloudware in the context of software paradigm, and then proposes and develops a new Cloudware PaaS prototype based on the micro-service architecture and container technology. In this prototype, 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 paradigm completely.

Section 2 describe the related work. Section 3 introduces the concept and features of the Cloudware. Section 4 discusses the Cloudware in the context of software paradigm. Section 5 proposes the loosely coupled von Neumann computing model. Section 6 illustrates the details of our proposed Cloudware PaaS prototype architecture. Section 7 describes the experiments and result analysis of our method, and Section 8 concludes this paper.

2. RELATED WORK

The Internet, once a network of networks, has become not only the platform of choice for delivering services to increasingly mobile users, but also the connective tissue between people, information and things. Almost all new and popular computing and application paradigms were born in the Internet, or at least motivated by it, including Web 2.0, Social Networking, Mobile Internet, Cloud Computing [3], Internet of things, and big data. Software has played a central part in the evolution of the Internet. The open, dynamic, evolving environment of Internet computing continues to demand new software technologies in order to realize its rapid paradigm shifts.

To support Internet computing from a software engineering perspective, *Internetware* [1, 2] has been proposed as a software paradigm that enables developers to construct new applications or evolve legacy systems to include new characteristics. After 10 years of research and practice involving hundreds of researchers, professionals and students from China, the USA, Japan and Europe, the fundamental challenges to Internetware have been thoroughly investigated.

In addition, since the Web advent as an Internet portal, browsers have become a kind of indispensable tool to our lives. More and more people use browsers perform commercial activities and access business applications on the Internet. As working with a browser continues to gain acceptance, an emerging strategy for software providers is to publish their applications online, especially for those applications that were previously distributed in a desktop environment. This is so called *Desktop Software Virtualization* (DSV).

One straightforward solution to DSV is to directly port these desktop applications to Web applications. But porting requires substantial effort, and the results are sometimes below expectations. An alternative solution is to host a desktop application on a Web server and display its GUI in a browser. Users can manage desktop applications through standard Web protocols and interact with the applications using a browser.

Researchers have attempted to realize this DSV concept in several ways. One solution is to create a Web user interface (UI) for a desktop application and providing a wrapper to mediate the interaction between the UI and the application [7, 8, 9]. However, software companies would need to implement a dedicated Web UI and wrapper for each desktop application. A simpler solution is to design a library for desktop applications to generate Web UIs dynamically [10, 11]. This provides a generic way to publish desktop applications on the Web, but there's little support for managing the execution of applications. Other research has deployed applications in a desktop environment and adopted a Web-based remote desktop method to support users in interacting with the applications through a browser [12, 13]. Although users can manage applications directly in a desktop environment, sharing the whole desktop is inefficient, especially in a network environment with low bandwidth. In recently, Chen and Hsu [14] introduced a desktop application service (DAS) framework that lets a Web server provide services based on existing desktop applications. The DAS framework augments desktop applications with browser-access capability, allowing an application to display its window in a browser without any modification. But, in essential, this method is still a virtual desktop based methodology, and cannot utilize the resource efficiently, such as underline computing and storage. It cannot deploy the specific software application in a massive and scalable way, which cannot utilize the current Cloud computing infrastructure efficiently.

To solve the problems above, this paper proposed a Cloudware paradigm based method which can directly deploy software on the cloud. With the emerging of the Cloudware, software paradigm will further evolve to its next generation which will be described in this paper in detail.

3. BASIC CONCEPT OF CLOUDWARE

Software are everywhere nowadays. With the development of cloud computing and virtualization, more and more software tends 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.

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 end users can get access to it through any browsers [15, 16].

Obviously, Cloudware distinguishes from traditional software in terms of forms, structures and behaviors. To be specific, Cloudware has the following features:

Running on the cloud. All the software on desktops and the necessary resource are deployed on the cloud, including native cloud software and traditional on-premises software.

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 the Internet. All the services are interacted by the 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.

Unify platform. Cloudware needs 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.

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, 5G network technology and HTML 5 front-end interactive technology, achieving such characteristics becomes possible. In particular, the characteristics of Cloudware depend 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.

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-services , 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 [17], H.265 [18] and Webm [19], 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.

Table 1 Differences between Cloudware and Web application

	Cloudware	Web application
Computing	Completely on cloud	Partialy on terminal
Output	Windows interactive images	HTML, CSS, Image
Input	Mouse and keyboard event	HTTP request
Storage	Copletely on cloud	Partialy on terminal
Recovery	Yes	No

Table 2 Differences between Cloudware and cloud desktop

	Cloudware	Cloud desktop
Computing	Completely on cloud	Partialy on terminal
Service type	Only sofware service (SaaS)	Desktop system service (DaaS)
Install	Immediately used	Download and install
Output	Windows interactive images	VRAM rendering information

At the same time, we can also compare the differences between the Cloudware and desktop application as well as the cloud desktop, as shown in Table 1 and Table 2 separately.

4. CLOUDWARE AS A PARADIGM

This section discusses the Cloudware in the context of software paradigm. A software paradigm usually concerns four aspects [2]: what is to be constructed and executed; how to develop the resulted software artifacts or entities (development techniques); how to run the artifacts or entities (runtime system supports); and how well the constructed and executed software can perform (the promised software qualities).

4.1 Cloudware software model

The Cloudware model should specify the form, structure and behavior of the software entity as well as the user interaction. These will determine the principles and features of the corresponding software technologies (programming languages, development approaches and runtime mechanisms). The Cloudware software model should leverage both legacy software and new features. The basic Cloudware can be built upon current popular technologies, such as object-oriented technologies or services computing technologies. But new capabilities should also be provided.

The Cloudware software model concerns three aspects: client, cloud and interaction channel, as well as their relationships, as shown in Figure 1.

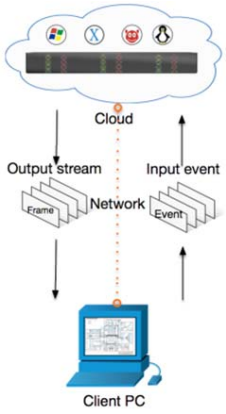

Fig. 1 Cloudware interactive architecture

4.2 Cloudware operating platform

Software operating platform realizes the elements and their relationships of the software model. Cloudware operating platform should provide a runtime space to operate Cloudware entities and their interaction. It should equip legacy software systems with Cloudware features, and should also manage the applications and itself in a more intelligent and automatic manner.

Cloudware reflects as micro-services 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 trend of software evolution. Cloudware can be regarded as a browser to provide interactive services components, but its operation is on the client side.

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, which can achieve the same effect as the local software. The specific interaction process is shown in Figure 2.

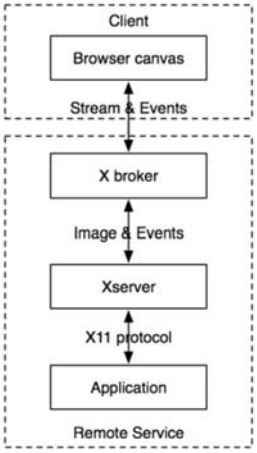

Fig. 2 Interactive principle of Cloudware

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.

4.3 Cloudware engineering approach

The engineering approach should systematically control the whole lifecycle of Cloudware development, including requirements specification, design, implementation, deployment, maintenance and evolution. The Cloudware engineering approach follows the core principle of "Software Architecture of the Whole Lifecycle". Software architecture acts as the blueprint and controls every stage of Cloudware development. To support the online development of Cloudware applications, the Cloudware and their on-demand interaction are implemented and governed based on software architecture.

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 systems (such as GitHub) and task management systems, Cloudware development can follow the 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-services . Cloudware development should be oriented to micro-services component form, and should not depend on the specific operating system and hardware architecture.

Deployment. The deployment of Cloudware is actually micro-services 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.

4.4 Cloudware quality assurance

Software on the Cloud usually serves a large number of users in an online and simultaneous style via Internet. Cloudware quality framework should not only define quantitative and qualitative measurement methods for various quality attributes such as performance, reliability and usability, but also make comprehensive tradeoffs among these attributes. To promise the Cloudware quality, it requires the quality assurance mechanisms by both engineering approach at development time (e.g., testing, verification and validation) and software running at runtime (e.g., online evolution, autonomic system management).

As Cloudware paradigm emerges at it's very early age, we mainly focus on its performance aspect of the quality, such as Cloudware boot time, networking low latency, and user experience in this paper. The research of the whole quality assurance mechanism for Cloudware will be carried out in the near future.

5. LOOSELY COUPLED VON NEUMANN MODEL FOR CLOUDWARE

Since the computer science appeared, the scientists represented by John von Neumann dominates the main stream of computing paradigm, and these theories are called von Neumann Architecture which is used by the majority of computer architecture researchers. It indicates that computer is made by five components, namely storage, operator, controller, input and output device, as shown in Figure 3. We can see a tight relationship between these five components, and all of these parts are necessary.

The traditional von Neumann architecture can be represented by the following tuple:

$$VC = (C, A, M, I, O) \quad C, A, M, I, O \neq \phi \tag{1}$$

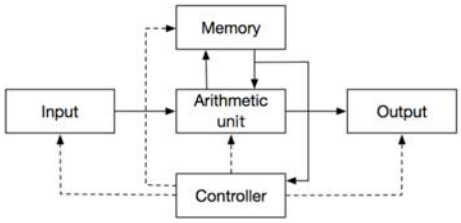

Fig. 3 von Neumann Architecture

in which *C* represents for the controller, *A* for the operator, *M* for storage, *I* for input, and O for output.

On the other side, in the Cloudware computing model, the input, output, storage and computing do not locate in a single computer system. They may be in different areas and be connected to the Internet. For examples, the input and output for cloud are deployed on the terminal while storage, controlling and computing on the cloud, and they communicate with each other through networks. And this article calls this kind of model the Loosely Coupled von Neumann computing model, which is the basic theory of Cloudware computing model.

5.1 Loosely coupled von Neumann model

The key Loosely Coupled von Neumann computing model is to decouple the five components and to make them into complete systems which can run separately and can communicate with each other through networks. A typical loosely coupled von Neumann computing model is shown in Figure 4 (solid line represents for data stream while the dotted line for controlling stream).

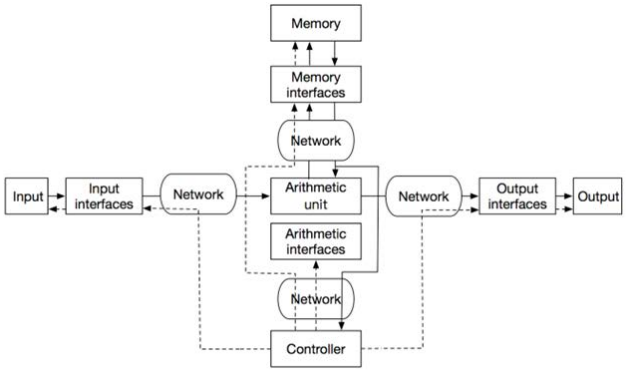

Fig. 4 Loosely Coupled von Neumann computing model

We can see that input, output, computing, storage and controlling are all connected by networks. Their external communication is achieved by the correspondent interface. This makes it possible for those components to extend themselves.

The Loosely Coupled von Neumann computing model can be represented by the following tuple:

$$LVC = (C, A, Ai, M, Mi, I, Ii, O, Oi, N) \quad C, A, Ai, M, Mi, N \neq \phi \quad (2)$$

in which *C* represents for the controller, *A* for operator, *Ai* for the calculating interface, *M* for storage, *Mi* for storage interface, *I* for input, *Ii* for Input interface, *O* for output, *Oi* for output interface, *N* for the network. We can see that input and output are not necessary, which can be immediately achieved through networks.

In addition, the network devices include those physics connection or wireless connection devices which support the Internet or other bus architecture, not only IP network. N can be represented as the following tuple:

$$N = (Wire, Wireless, Routed) \quad (3)$$

in which Wire represents the physics connection; Wireless for the wireless connection like LAN; Routed for those devices who can communicate with indirectly passed route like WAN.

In the Loosely Coupled von Neumann computing model, Interface is the key for the communication among different components to achieve the protocol in distinct level. In the Cloudware system, different components are abstracted into services while Interface becomes the interface of the external services.

The module features of Loosely Coupled von Neumann computing model are very typical, endowing every component with high flexibility, which are listed as below:

Infinite extension of the component through the network. Because the network introduced in every component of Loosely Coupled von Neumann computing model, it is easy for the model to extend to the whole network.

The independent component system. For the single component in Loosely Coupled von Neumann computing model, it must be able to run separately and to provide an external interface, ensuring its validity even in the case that external component failed to work.

The hot plug components. The loosely coupled von Neumann computing model supports hot plug. For instance, any display devices with different resolution can be immediately replaced with no break during the process.

Infinite storage. Because the storage is deployed behind the interface, the physics storage is not transparent to the computer system. Therefore the logical storage information can only be accessed by the storage interface. It is easy to establish the backend distributed storage in this sense, namely the infinite storage space.

Networking self-adaption. The networking in Loosely Coupled von Neumann computing model does not refer to the present IP network, but the network devices of physics or wireless connection. The protocols in the lower level dependent on the unified controller.

5.2 Application of loosely coupled von Neumann computing model in Cloudware

The typical features of Cloudware reflect the loosely coupled von Neumann computing model, the computing, storage and controlling are deployed on the cloud while the input and output on the terminal. The communications between the components are achieved by network， and the input and output are hot-pluggable, as shown in Figure 5.

Cloudware model use the terminal environment to achieve the independent system of input and output, and to interact with each other through TCP(UDP)/IP protocol. The programs on cloud can utilize the storage resource, and they can also use the system call of OS to achieve storage function. That of the computing component is supported by the CPU of the Physics machine, and that of the controller is supported by the correspondent with the related Cloudware application. In Cloudware computing model, every module is abstracted as service, and the interactive service of Cloudware is deployed on the terminal while the computing and storage on the cloud.

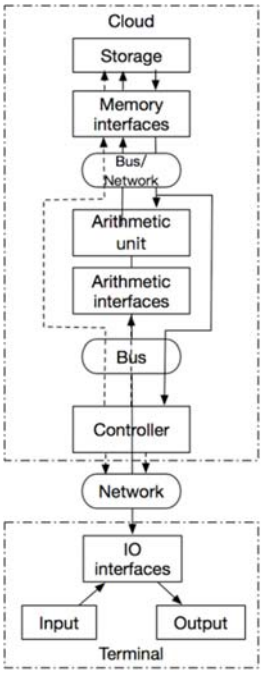

Fig. 5 Loosely Coupled von Neumann model of Cloudware

6. CLOUDWARE PAAS ARCHITECTURE

6.1 Architecture

In order to verify the technical conditions on the current cloud, this paper presents a Cloudware of PaaS platform prototype based on micro-service architecture: CloudwareHub. 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 6.

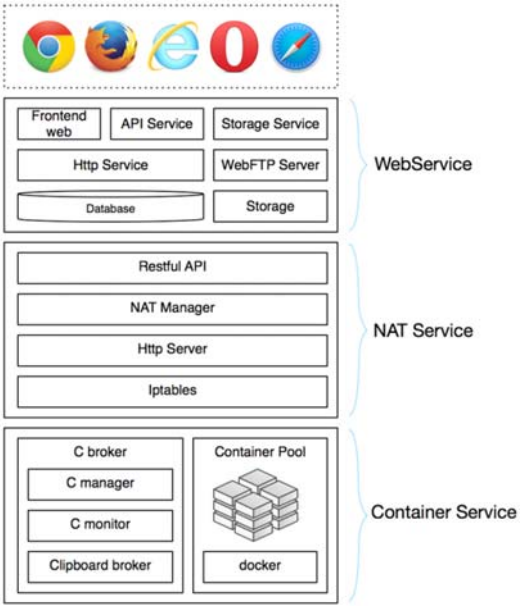

Fig. 6 The overall architecture of Cloudware

From Figure 6, we can see that three main parts make up the whole Cloudware PaaS platform architecture: Container Service, NAT 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; NAT Service provides Network Address Translation services, and achieves reverse proxy poll. NAT Manager is responsible for managing NAT rules of Iptables and exports restful APIs; 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 WebFTP Service use WebSocket [7] protocol in Web Service realizes the communications between files in cloud with client.

6.2 The control plane and data plane

The communication process of different services of Cloudware PaaS platform is shown in Figure 7, which shows that *c1* to *c5* is the tunnel of control plane, and *d1* to *d6* is the tunnel of data plane. The main goal of control plane 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. 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 programs through libraries like Xlib or XCB, while C manager controls Docker by using API offered by Docker and corresponding language binding.

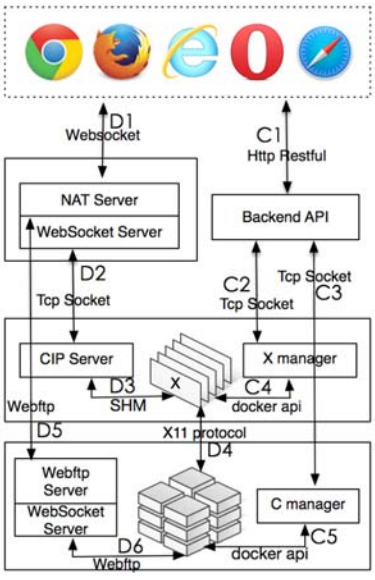

Fig. 7 Control layer and data layer

The data plane mainly focus on data flow in interaction level. *D1* is responsible for communications between browsers and Websocket Server. It sends input from keyboard and mouse to Websocket Server through Websocket protocol and Websocket

Server send output from CIP Server to the browser. *D2*, which is based on TCP protocol, is the interaction between Websocket Server and CIP Server. It sends client input events and to CIP Server for further process.

D3 is interaction between CIP Server and Xserver. CIP Server sends mouse and keyboard events to client to Xserver, using libraries like XCB and Xtest extension, meanwhile send changed images from Xserver which have been compressed to H.264 stream to Websocket Server, using Xdamage extension to get image change notify. And Websocket Server send it to the terminal display. For faster getting images, CIP Server shares its memory With Xserver to get corresponding bit images. *D4* is the data tunnel between software and Xserver, which is realized by X11 protocol. *D5* is the data tunnel of Webftp, which is used to realize the data share between client and Cloudware. Data of client can be sent through d5 to share storage, finally shared storage is mounted to Cloudware container and do I/O through *D6*.

6.3 Internal protocol architecture

Since Cloudware PaaS prototype 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 CIP (Cloudware Interacting Protocol) and Webftp protocol as shown in Figure 8.

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, and Websocket protocol is used to implement two-way communication process between browser and server.

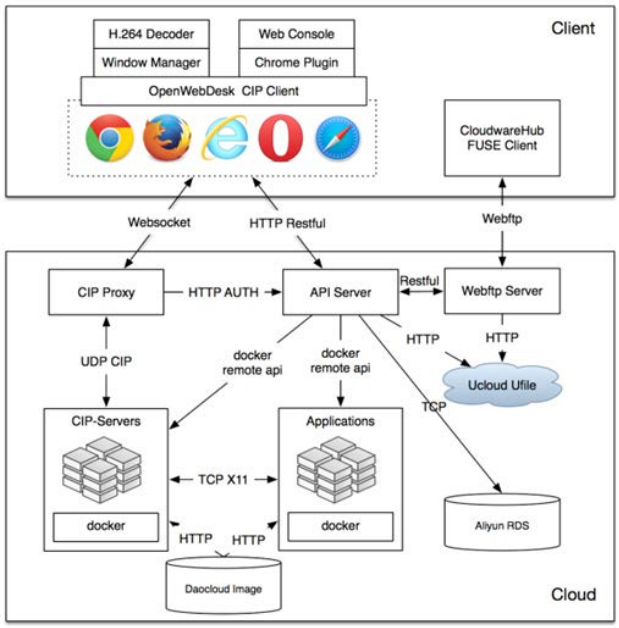

Fig. 8 Cloudware micro service communication protocol

In the whole protocol architecture, CIP protocol and Webftp protocol is special for the Cloudware, where CIP-compatible binary protocol is a set of cloud-piece display protocol X11 protocol, whose main task is to carry out UDP encapsulation, and Webftp protocol is set of file transfer protocol running on the Websocket protocol, achieving transparent access to the cloud and local file system [9].

6.4 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 the 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 fewer 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 to 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.

Scalable. 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.

7. DEMOSTRATION AND EVALUATION

This section first demonstrates the proposed Cloudware PaaS platform prototype, and then evaluates its performance by carrying out an interactive delay experiment that directly focuses on users' experience, which shows the effectiveness of our method.

7.1 Running environment and demonstration

Cloudware PaaS platform provides the micro-services environment. To achieve scalable deployment, faults tolerance and flexible configuration, we implement our system with the support of commercial IaaS. To test the validity, we make our Cloudware PaaS Platform prototype system, namely CloudwareHub[1], run on several cloud hosts with two cores, 4GB memory and 20GB disk. The Web Service runtime architecture is shown in Figure 9.

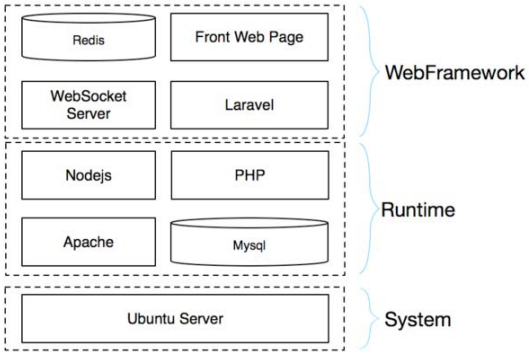

Fig. 9 Web Service runtime architecture

X Service and Container Service runs on another host to improve the efficiency of communication between Container and Xorg. The X Service and Container Service runtime is shown in Figure 10.

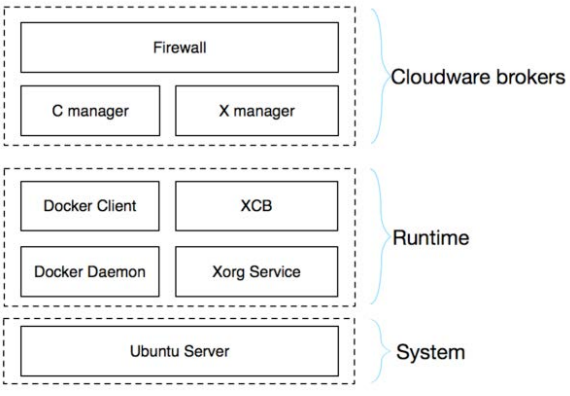

Fig. 10 X Service and Container Service runtime

Container Service is based on Linux with Ubuntu Server 14.04, 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[2], offering Cloudware service.

On CloudwareHub platform, you can first find a specific Cloudware in the Cloudware base, as shown in Figure 11, and then you can add your Cloudware in your workspace. Finally, you can run your Cloudware from your workspace by double clicking the Cloudware icons, as shown in Figure 12 (Running a Matlab Cloudware service). We also developed a Web desk style environment for the Cloudware service, named OpenWebDesk. Users can use the OpenWebDesk environment like a traditional desktop, as shown in Figure 13.

Fig. 11 Find Cloudware in the Cloudware base

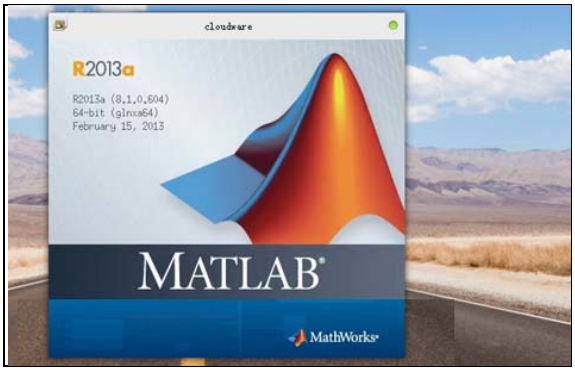

Fig. 12 Running a Matlab Cloudware service

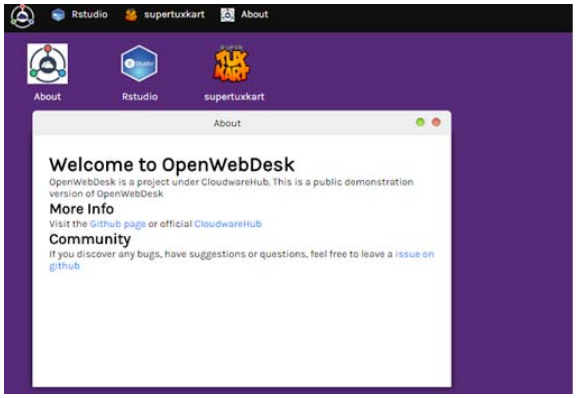

Fig. 13 OpenWebDesk environment

[1] http://www.cloudwarehub.com/

[2] https://www.ucloud.cn/

7.2 Experiments evaluation

Finally, to evaluate the user 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 to the experience. Bad internet environment will lead in poor user experience. Based on these results, we can further analyze the most influential interactive delay of Cloudware system.

There are two ways for video streaming transmission of CloudwareHub: FFRM (Fixed Frame Rate Mode) and EDM (Event Driven Mode). And we test them respectively and list the results below.

7.2.1 Influence from network bandwidth

Three specific Cloudware including *gedit*, *rstudio*, *supertuxkart*, are separately tested again to get delay in different terminal network, as shown in Figure 14. We can see that the delay is fixed when the data is of a small amount, the main cause of delay is not data or bandwidth, but the TCP process and memory copy. Rendering complexity has influence, but not the linear relationship. But in general, complexity increases the delay. In addition, the terminal network influent a lot to the experience. In a serious case of poor network and a large amount of data, TCP timeout will happen.

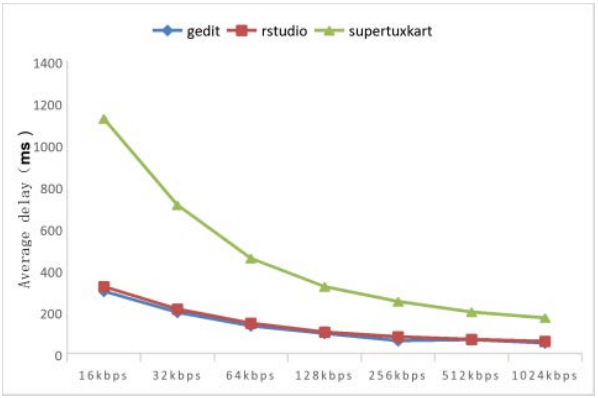

Fig. 14 The interactive delay(ms) of different Cloudware in different terminal network

7.2.2 Influence form GPU

To test the influence from GPU, we add cloud machine with GPU function on Ucloud, with four logical cores, 8GB memory, 20GB storage, 2Mbps public network bandwidth and Tesla K80 GPU which has two cores. We only use one of it to launch the experiment. Based on the hardware environment above, we run the Supertuxkart respectively in GPU Enabled and GPU Disabled environments. By simulated keyboard input, we collect delay every 2 seconds, as shown in Figure 15.

We can see that GPU can largely reduce the interactive delay mainly because the powerful 3D rendering ability of GPU. To further research the influence from GPU, we test the CPU utilization rate respectively in GPU Enabled and GPU Disabled, as shown in Figure 16.

We can see that GPU makes CPU free from computing intense tasks. Without GPU, CPU is responsible for all the rendering tasks, and its average utilization rate is higher than 350%, leading to a serious delay. While GPU reduces it to 70%, makes it possible to rapidly run the Cloudware and video streaming compression tasks.

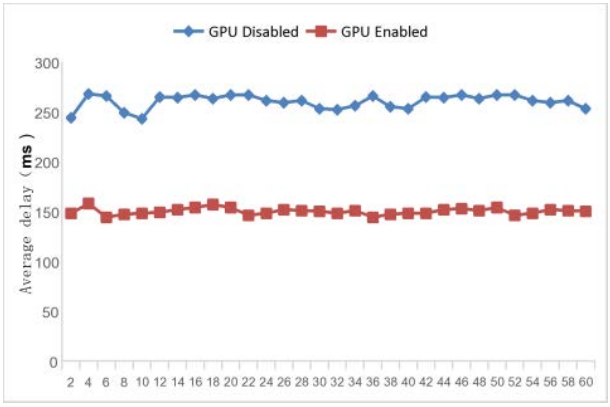

Fig. 15 Influence from GPU

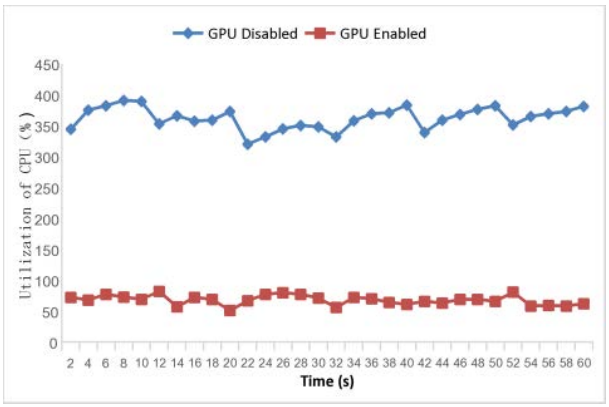

Fig. 16 Influence to cloud CPU from GPU

7.2.3 Influence from different transmission pattern

To test the delay of FFRM(Fixed Frame Rate Mode) and EDM(Event Driven Mode), we run Eclipse Cloudware in 50FPS and 20FPS. We collect delay every 2 seconds of the whole process from the terminal continuous input event to the terminal rendering , as shown in Figure 17.

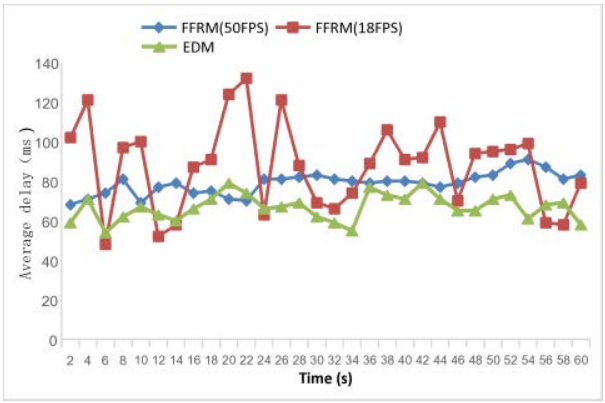

Fig. 17 Influence from different transmission pattern

We can see that in low FPS FFRM, the delay is not stable mainly because the rendering event is random while the compression program runs in a fixed cycle. While EDM has a relatively low and stable delay. Theoretically, FPS in high frame rate should has a lower delay than that of EMD, but the results show that this is not true. That's mainly due to the complex video compression programs, especially in the case of multiple windows. Its computing is complex, leading to the increasing system workload and therefore, the increasing delay. Comprehensive considerations of cloud and network should be taken to choose the most suitable transmission pattern in the real Cloudware system.

To sum up, the user experience of Cloudware is determined by many factors. Improving the performance of the cloud, and adding the GPU rendering tasks can significantly improve the Cloudware performance. In addition, improving network quality and reducing the amount of rendering data transmission is the key to improving the user experience of cloud. While how to solve the inherent problem of network latency and how to guarantee user experience in the jitter network situation are critical to future research.

8. Conclusion

The development of cloud computing not only leads to the reformation of data center, but also affects the traditional methods of 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. Regarding various kinds of cloud resource as services will promote the micro-service design model and make it easy to be adjusted for cloud. From the users' perspective, Cloudware paradigm is the trend, which can directly offer services to them. We believe that Cloudware paradigm will be the main application and develop model in the future.

The idea of Cloudware paradigm model largely relies on the structure of the cloud and the Internet. The user experience is also dependent on multiple factors. The key to improving the quality is to improve the internet quality and to reduce the data transported. Similarly, how to solve the fixed delay of the internet is the critical issue in further research. Under present circumstance, even though the containers and the video compression can meet the need of the majority of users, technological challenges also exist including further using rate increase, video stream cross-compression, self-adaptive network compression, real-time network, interactive terminal unified platform, GPU virtualization and transparent storage technology, etc. Because of the limited space, only parts of the challenges and problems are listed. Now the technology and theory of Cloudware are still in their early development, and the authors will continue to focus on researches and improvements in related fields.

9. Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant NO. 61672384), 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).

10. REFERENCES

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

[2] Mei H., Liu X., Internetware: An Emerging Software Paradigm for Internet Computing, J. Computer Science and Technology, 2011, 26(4): 588-599.

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

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

[5] Boettiger C. An introduction to Docker for Reproducible Research, ACM SIGOPS Operating Systems Review, 2015, 49:71-79.

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

[7] Grechanik M. et al., Creating Web Services from GUI-Based Applications, In Proceedings of IEEE Int'l Conf. Service-Oriented Computing and Applications, 2007, pp. 72–79.

[8] De Lucia A. et al., Developing Legacy System Migration Methods and Tools for Technology Transfer, Software: Practice and Experience, 2008, 38(13): 1333-1364.

[9] Meng X. et al., Legacy Application Migration to Cloud, In Proceedings of 2011 IEEE Int'l Conf. Cloud Computing (CLOUD), 2011, pp. 750–751.

[10] Lord J., The W12 Network Window System, Master's thesis, School of Computer Science, McGill Univ., 2012.

[11] Karampaglis Z. et al., Secure Migration of Legacy Applications to the Web, Information Technology and Open Source Applications for Education, Innovation, and Sustainability, Springer, 2014, pp. 229-243.

[12] Zhang B. et al., A Black-Box Strategy to Migrate GUI-Based Legacy Systems to Web Services, In Proceedings of 2008 IEEE Int'l Symp. Service-Oriented System Eng., 2008, pp. 25-31.

[13] Wang S.T. et al., Development of Web-Based Remote Desktop to Provide Adaptive User Interfaces in Cloud Platform, Int'l J. Computer, Information, Systems and Control Eng., 2014, 8(8): 1195-1199.

[14] Chen B., Hsu H., Huang Y., Bringing Desktop Applications to the Web, IT Professional, 2016, 18(1): 34-40.

[15] Guo D., Wang W., Zhang J.X., et al., Towards Cloudware Paradigm for Cloud Computing, In Proceedings of The 9th IEEE International Conference on Cloud Computing (CLOUD), 2016, San Francisco, USA, June 27 - July 2, 2016.

[16] Guo D., Wang W., Zeng G.S., et al., Microservices Architecture based Cloudware Deployment Platform for Service Computing, In Proceedings of 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE), Oxford, UK, 29 March - 2 April, 2016, pp358-364.

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

[18] Lu Y., Zhang Q., Wei B., Real-time CPU based H.265/HEVC Encoding Solution with x86 Platform Technology, In Proceedings of Computing, Networking and Communications (ICNC), 2015 International Conference on. IEEE, 2015: 418 - 421.

[19] Bankoski J., Intro to WebM, In Proceedings of International Workshop on Network & Operating Systems Support for Digital Audio & Video. ACM, 2011: 1-2.