**AntOS: A novel solution of cloud-based OS based on X11 and Docker**

Zhen Du1, ZhuQing Xu1,Fang Dong1

1*Dept. of Computer Science and Engineering, Southeast University, Nanjing, P.R. China*

**Abstract**

Cloud-based operating system is the system that sets storage resource and computing resource in cloud. And people can use cloud-based operating system everywhere only by thin PC and remote-controlling software. With the development of technology, cloud-based operating system is increasing popular in enterprise market. Nowadays, product based on cloud-based computing are put forward continuously, like ChromeOS and SUNDE. But traditional cloud-based has some disadvantage: It is difficult for ChromeOS to transplant traditional desktop software to such platform, and the cloud operating system bases on web is weak in functionality; the high cost of performance as well as network also can’t be ignored.

To solve these problem, this article design AntOS, a full-featured operating system with less load in backend and network, which Linux software is deployed in cloud server and people can use them remotely. AntOS takes advantage of Docker, a container technology, making users able to deploy, install, and use software remotely. It also employs X11 commands, translating them from server to client and then client analysis these commands for GUI-drawing. So AntOS only need cheap embedded device for user to use computer with lower-cost hardware and screen.

**Keywords:** Cloud-based Operating System, Linux, X11, Docker

**1. Introduction**

With the development of technology, the devices with all kinds of operating system which allow people install software are more and more popular, like personal computer, mobile phone as well as tablet PC. They have a set of intact hardware, and software can be installed above operating system to enrich its functions[1]. However, this kind of operating systems have some shortage:

1. They exceedingly depend on local resource. Because all of software was deployed locally, computing resource and storage need by software was provided by device itself. When the device can’t satisfy the need of software in performance and storage, software doesn’t work. So it is harsher for the software to run in traditional operating system. For example, as far as Windows developed by Microsoft, the operating system itself need 1GHz CPU, 1-2 GB RAM, and 16-20 GB storage.
2. The hardware is much more expensive. Because all of the computing resource and storage need by software are provided by device itself. It must have a faster processor and more storage, which increases the prime cost. Mainstream chips like Core i5 worth 200 dollar. A ddr3 memory costs at least 40 dollars. This is big expenditure for devices which update every 2-4 years.
3. The lack of data safety. Traditional PC depends on local storage resource. So if the operating system is reset or the device is lost, the data will be leaked.

With the growth and popularization of cloud-computing technology, the concepts and ideas about cloud-based operating system are proposed continuously. Different from traditional operating system, cloud-based operating system deploys software in cloud, user can make use of software and operating system by browser and other client remotely. The software can be access everywhere in any kinds of devices by users[2]. Because software is deployed in cloud, so cloud-based operating system has some incomparable advantage:

1. Cloud-based operating system allows flexibility and variety in use, we only need to apply more performance resource and storage resource when we need. In this way, waste of computing resource can be avoided, which can save the cost.
2. Convenience in data-managing. Because all data resource is saved in cloud, it is reliable for cloud-based system to manage data. What’s more, all the fruit of employee’s work must be saved in cloud, so it can reduce the difficulty and cost of enterprise for data-managing.
3. Easy for software maintenance. Software deployed in cloud is much more convenient for user to manage uniformly. We only need to update software for one time, all of our client can use the software has been updated.
4. Lower cost in several dimensions. One of the advantages of cloud-based is to move much performance resource and storage resource to cloud, which can reduce the cost for user to keep the operating system staying in good state. These advantages make cloud-based operating system lighter and stronger, which makes it doesn’t need much from hardware, and only a thin PC can satisfy its need. Moreover, Hardware of cloud operating system doesn’t need to update frequently[3]. So cloud-based operating system has low prime cost in hardware, management, development as well as maintenance.

But traditional cloud-based operating system also has some problems. Although there is no accurate definition for cloud-based operating system, in term of the way we use cloud-based operating system, we can divide the all cloud-based operating system into two kinds: one is based on web and bowser, the other is based on virtual machine in cloud server and remote control software in client.

Operating system based on web is driven by browser. All the software of it is developed by web technology. This way largely reduce the cost of client, we can run this operating system only by simple and light client[4]. In contrast to “laptop”, a device named “webtop” working in this way[5]. This way has two shortages:

1. Limitations of the function. Because this kind of cloud-based operating system developed by web, limited by functions of web and browser, software run in this platform will be weak in functionality.
2. Difficulty in transplantation. Web is the environment totally different from traditional operation system, the developments of most of software need to be and start all over again.

Besides cloud-based operating system based on web, there has other operating system based on virtual machine and remote-controlling software. Broadly speaking, all the mainstream cloud platform belong to this category(Using virtual machine to divide resource of backend for user into different part[6] ). This method is not limited to web and browser, a desktop operating system can be provided to user directly. But this kind of cloud-based operating systems also have some disadvantages:

1. Delay can’t be avoided in virtual machine. Although with the development of virtual machine, the extra cost of it becomes less and less. But virtual machine has to run guest operating system above host operating system, the guest operating system will cause extra cost, will can’t be avoided. Undoubtedly, user have to bear more delay because of this extra cost.
2. The defect of interaction. From the view of user, command line is not humanitarian enough. So there a new solution in traditional cloud platform named VNC[7]. VNC, Virtual Network Computing, is a graphic remote-controlling software. It can display graph of operating system (like desktop of system) to user remotely in real time. VNC records the image of operating system and translate the video from server to the screen of client by video stream. User can interact with the picture in screen can his operation can send back to server so that user can interact with operating system remotely. Because VNC bases on video stream, so a relatively large amount of data need to be translated, VNC depend largely on the quality of network. And because of video compression and decompression, the distort of image can be ignored[8].

To solve these problems. This article propose a novel solution of cloud-based operating system based on Linux X11 and Docker, we name it as “AntOS”. It solves the problem of ChromeOS, including weakness of functionality and difficulties in software transplantation. And AntOS remains the way people interacting with traditional operating system, improving the efficiency of communication between client and cloud server, reducing the extra performance cost which can’t avoid for virtual machine. It provides these feathers:

1. AntOS can support complete Linux software, while it can also promise the less performance cost of backend. Instead of depend on solutions like virtual machine and openstack, AntOS take advantage of Docker, which promises both good functionality and low performance cost of servers[9].
2. A low dependency on network quality between client and server. In AntOS, video stream is replaced by GUI-drawing command, which includes less amount of data and depends less on network quality.
3. Faster response. Because AntOS’s backend use lighter technology, and uses responsive-first load balancing strategy, AntOS can response faster than traditional cloud-based operating system.

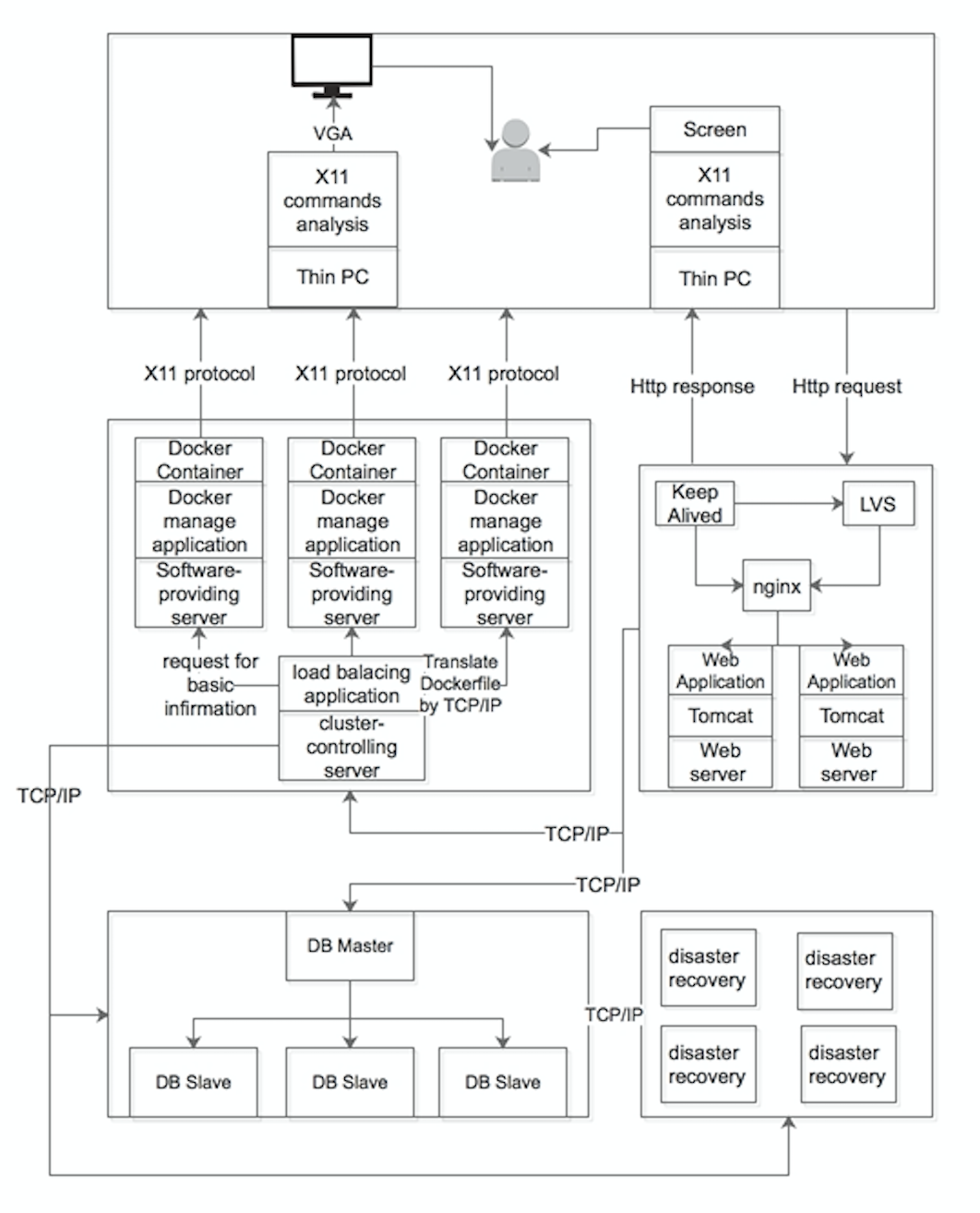
To achieve above points, and promise the whole system is easy to use, AntOS makes following contributions:

1. A GUI interface based on X11. X11 commands will be translated from server to client and analyzed locally to draw a GUI interface. Compared with traditional way like video stream, this way has faster responsibility and less dependency on bandwidth.
2. A responsive-first load balancing strategy. AntOS takes the network quality into first consideration to promise faster responsibility and less delay.
3. A dockerfile-writing tool based on text analysis. It is designed to solve the problem that Docker image is difficult to build. This tool will get the actual name of software and cope with error about dependency through user’s input, and generate dockerfile automatically.

This make these arrangements in following section: In section named “Design”, we will introduce the architecture of the whole system, and descript the implementation details of client, web server as well as software-providing server. In the section named “Implementation of key function”, we will introduce the implementation of three core function: translating and analysis of X11 commands, load balancing strategy, a tool to generate dockerfile. In “System Implementation and typical instances” section, the article will introduce the achieving result and typical instance.

**2. Design**

**2.1. System Architecture**

****

*fig.1.system architecture*

The design of system architecture can be divided into two part, and more details are contained by fig.1.

A Thin Client with a tiny Linux named AntOS. AntOS is based on Linux. It has a Linux core and several dependencies to promise a program made by Qt and Python can be used by user. This program is the only interface user can interact with our system.

The cluster of servers to provide software for user to use. The software is install in a in the cluster of servers. If someone wants to use a software, a thin client driven by AntOS is not enough, he has to connect with the server and the server will send the command for the UI-drawing to client, and client collects user’s operations and send back them to servers. User can interact with software and our system remotely by this way.

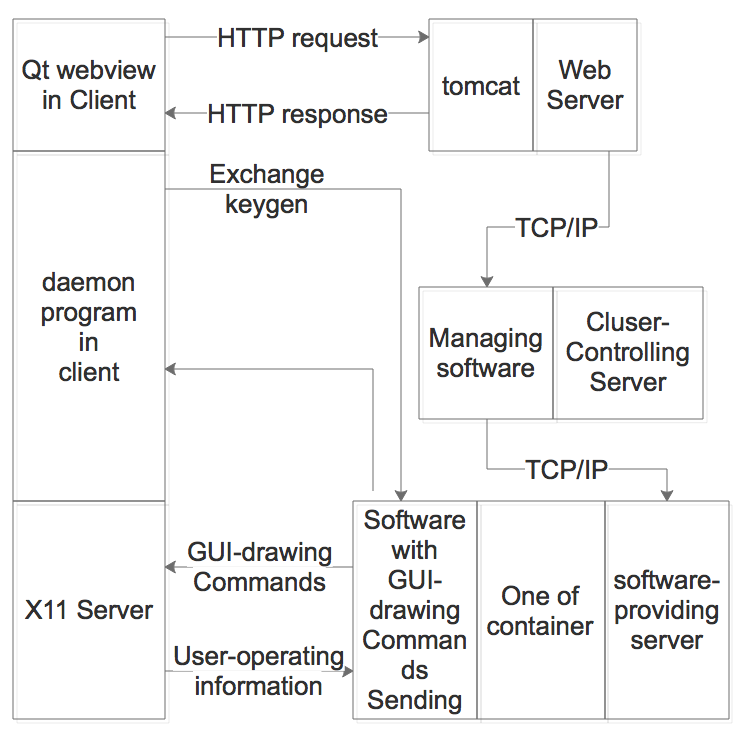
The servers in cluster can be separated into 4 part by its rule: cluster-controlling server, web application server, software-providing server, and sql server.

Cluster-controlling server is design to get the basic information (cpu occupancy rate, RAM usage, network load) of software-providing servers and choose the server with the best condition to run software for user.

Web application server provide a web application for user to manage his software. The client will connect to it directly.

Sql server store the basic information for the whole system to run, including user’s username and password for login, etc.

**2.2. Components Interactions**



*fig.2. interactions of different component.*

As we can see in the fig.2, the cluster-controller frame shield the different conditions of servers. When users want to install or open a software, they communicate with the cluster-controlling server by sending Http request. And the frame will communicate with servers by TCP which provide software as service. Taking our strategy of load balancing into consideration, the cluster-controlling server select a suitable server and let that server install the software or run a software.

When the cluster-controlling server select suitable server, it will translate some basic information to both server and client. The server provides corresponding software by starting the container, and client will send messages for requesting UI-drawing information from container in corresponding server. In the end, the client will communicate directly with servers providing the software the client need.

After the user use the software, he can stop it by shutdown the container the software in. That is a operating cycle of a user to use a software of AntOS.

**2.3. Design of each module**

In this section we will introduce AntOS from both client and server. The client of AntOS can be separated into two parts, one part is a web application that start automatically in full screen and can’t be stopped unless the hardware shutdown. This part is design to manage the software installed in server. Another part is a C/S structure software that analysis the GUI-drawing command and display the graphic interface in the screen. The servers provide the service that managers the nodes as well as the software controlled by users remotely.

**2.3.1. The thin client**

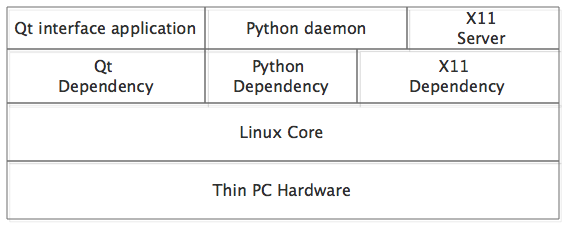
To reduce the cost from hardware as well as the whole operating system of the client. AntOS was designed to be more light with no loss of the full functions compared with the traditional operating system like Windows, Linux, Unix, etc.

It has a core of Linux. To make itself more lightweight, AntOS doesn’t have a desktop system, but we remain the X11 server for analyzing the GUI-drawing command.

The Linux core and the dependency that we choose as little as possible promise the program made by Qt as well Python can be working in the right way.

AntOS has two part:

1. A GUI Interface made by html and Qt (We call this program “Qt interface program”).
2. A daemon program to analysis GUI-drawing command. We will introduce these two part in details.



*fig.3. the architecture of AntOS in client*

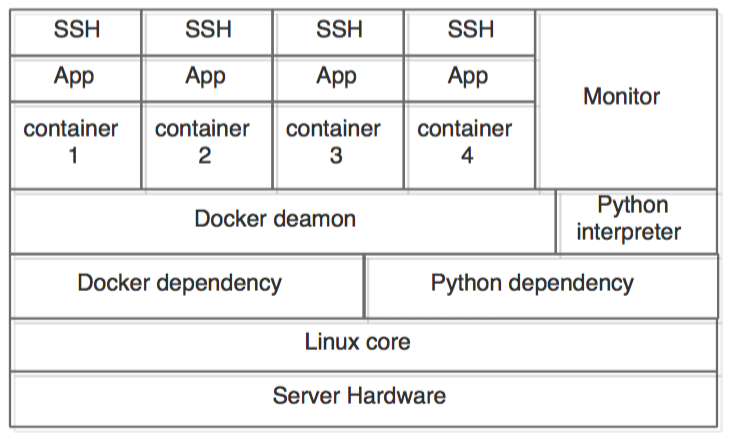
The Qt interface program is the whole image interface of AntOS. Substantially, it is a simple browser program access the tomcat deployed in cluster-controlling server. This program is designed for user to manage the software (start, stop, delete, etc.). It sends http request to the tomcat of cluster-controlling server. This server takes the place of client and finish what the client want to do. According to different results, tomcat return different information to Qt interface program and user can see it in the web view in the screen.

If users need to use a software they want, not only Qt interface program but also daemon program with X11 server.

X11 is a GUI library. Almost all GUI windows are drawn by X11. It provide the basic GUI environment[11]. It is a client-server structure, the software we used is the X11 client, and the users interact with the X11 server. X11 server is directly related to GUI-drawing, it draw the GUI in the screen so user can see the graphical information of the software they used[12]. The X11 server also collect user’s action and send them to X11 client, the software itself, and X11 client return back the new command to X11 server for drawing graphical window in screen.

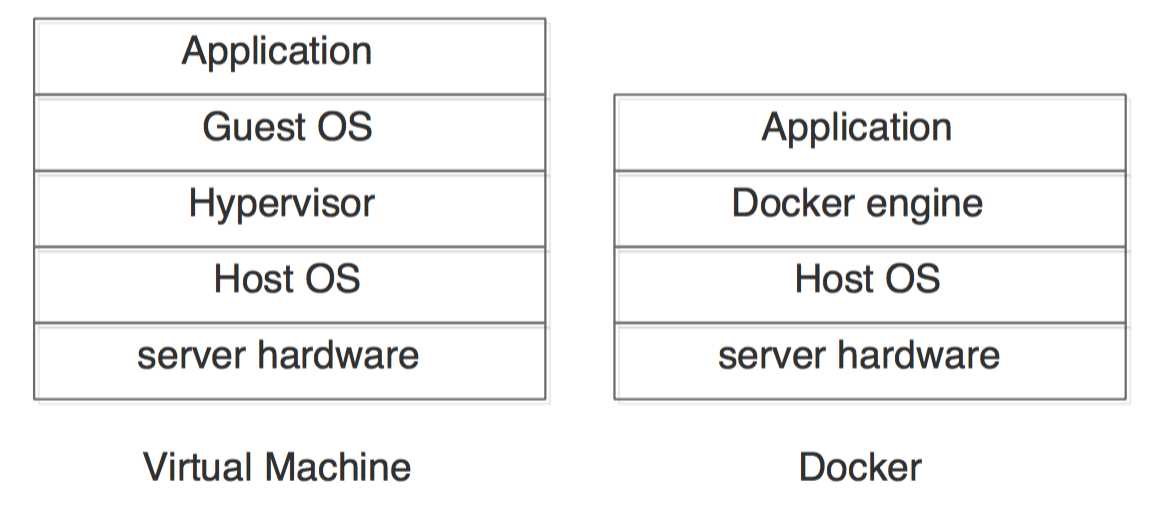
So when user need to open a software, the whole system will work in this way: Firstly, user click some buttons to send some http request to the tomcat in the cluster-controlling server, the cluster-controlling server select a program-providing server and send back the statement as well as the ip of this server to the client. Qt interface program transmit the information to the daemon program in this device. For security, we use asymmetric cryptography for the communication between client and program-providing server, so daemon program need to send the client’s public key to it. After the server receive the client’s publish key, the software will start running can our user this program by remote controlling. All the GUIs is drawn in the screen user faced with by X11 server, and all the operations send back to X11 client in servers.

**2.3.2. Lightweight and flexible service cluster in backend**

****

*fig.4. the architecture of AntOS in software-providing servers*

Because one of our goals is to make our cloud based computing service more lightweight, we abandon the traditional way that dividing the software-providing into several parts by using virtual machines like VMware and a frame like Openstack[13] to control them[14]. As is shown by fig.5, the virtual machines let several guest operating systems run on one host operating system[15], and because of this reason, there are always more than one OS cores run on the same physical machine. What cloud based computing servers provide are the softwares installed in guest operating system. So although want our user need is only a software, but we have to run the whole guest OS core for it. Absolutely, it is not efficient because of a big extra cost[16][17].



*fig.5.the simple structure of virtual machine and Docker*

The architecture of software-providing servers is shown by fig.4. We use Docker as a container technology[18] instead of using traditional solution like virtual machine and Openstack. Container technology can divide the whole physical machine and the operating system into several parts but it doesn’t need to run several guest operating system. It driven by “cgroup” library[19], dividing the system by setting different authorities for different users. Because all containers run on a single Linux, so all software our user using depend on only one Linux core.

The build of a container depends on dockerfile, which determines what should be install in the container and the basic statement of the container. Because we usually use one container to run only one software, which means a software is mapped to a dockerfile[20].

When the cluster-controlling server get the basic information of the suitable software-providing server that run the application our user need, it will send the dockerfile to software-providing server(All this work has been done by monitor in software-providing server). Dockerfiles won’t transform into containers the applications running in directly. Docker daemon program installed in every software-providing servers, they analysis the dockerfile and generate the imagine with the help of cache of other imagine[21]. After the imagines transform into container, the application installed in this container can be used by user immediately. And the GUI drawing commend will be send and encrypted by SSH protocol[22]。

**3. Implementation of key function**

In this section, we will introduce the details of the implement of AntOS. Because the implementation of the whole system is very complex and we use many technologies of could-based computing, but the most pivotal details of our implementation include three part:

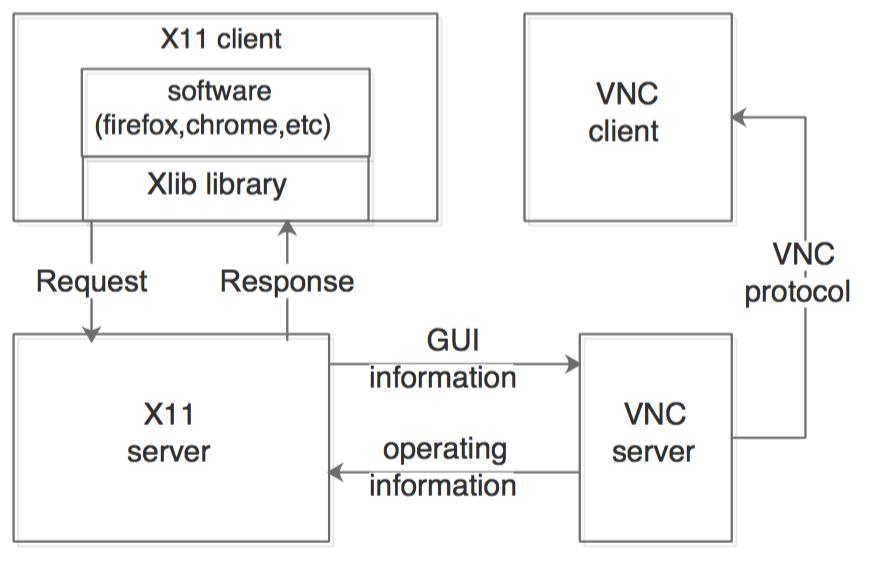
1. Using X11 commands for GUI-drawing remotely, we take both load of network and quality of video into consideration.
2. We implement an efficient load-balancing strategy between software-providing servers. We take importance in network quality, preventing one of the server in the cluster bear much more load of network than others, which can promise the responsiveness of AntOS.
3. We user Docker instead of virtual machine to reduce the extra need of performance, which will cut down the prime cost of our service.

Now we will introduce these feather in details.

**3.1. Container providing service with X11**

As we all know, most of graphic remote control is based on video stream, but video stream means the server need to translate color information of each pixel. So a mature method to optimize this process was invented: The combination of imagination compression and incremental transmission of picture. This way reduce the load of network by compress each frame of video and only translate the part that is different from last frame. But this way largely reduce the quality of clarity and color. By using X11 commands for GUI-drawing remotely, AntOS not only provide lossless imagination for the most of time and reduce the load of network averagely.

Now this section will analysis why AntOS choose translating X11 commands instead of video stream. VNC is a remote control software. It work like this[23][24][25]:

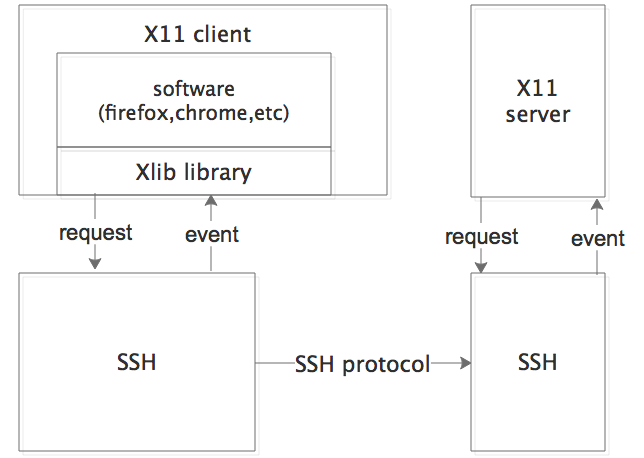


*fig.6.the process VNC works*

Software、X11client、X11server and VNC server are installed in server and VNC client is installed in client the user use. Software we use are client of X11 library by using the API of X11 client. X11 server is a other process work closely with X11 client. The software run to satisfy the need of user by receiving and executing the commands translated from the operations made by users in VNC client.

After X11 client executes the command, some changes of GUIs will take place, so X11 client will send GUIs-drawing command to X11 server by Unix socket. And X11 server generate the graphic information of software. VNC server gets these information in the way similar to screenshot applications and sends all information, especially color information(like RGB values), to VNC client in the way the VNC client can analysis by TCP/IP.

AntOS send the GUI-drawing commands directly. We take advantage of an important feather of X11------X11 commands[26]. It works in the way shown by fig7.



*fig.7. the process the AntOS work*

In AntOS, software and X11 client are installed in server while X11 server in the client the users use. AntOS abandons Unix socket and uses TCP/IP to send GUI-drawing commands remotely. Differing from VNC , X11 server answer for the graph part of a software, listening user’s operations and send “event[27]” to X11 client, while X11 client answer for the computing part of a software. What X11 client sends to X11 server according to its computing result is named “request[28]”.

The process to translate information between software and user of AntOS is a subset of VNC. Because a step of VNC related to video compression is a large performance expenses and AntOS doesn’t have this step. This is a convincing reason for us to believe the method AntOS used is more high-speed as well as Lightweight.

AntOS also has less bandwidth dependency. H.264 is a mainstream video compression and translation protocol[29]. According to H.264 document, code rate of a 1080p video is suggested to 8500Kbps. According to X11 protocol, an “event” message usually has a size of 32 bits, and a “request” usually has a size of hundreds bits. So in practical use, the need of bandwidth is no more than 1200 Kbps. And after the software is opened completely, the main part of GUI is no need to change, the need of bandwidth would be even much less.

So in term of GUIs display, AntOS does better than traditional cloud-based operating system both in bandwidth-dependency and performance expenses.

Because all software using Xlib[30](X11 library) is the X11 client, so what we need to do is to install a application with GUI and using SSH encryption protocol for security. The basic Docker should include:

RUN apt-get install –y openssh-server

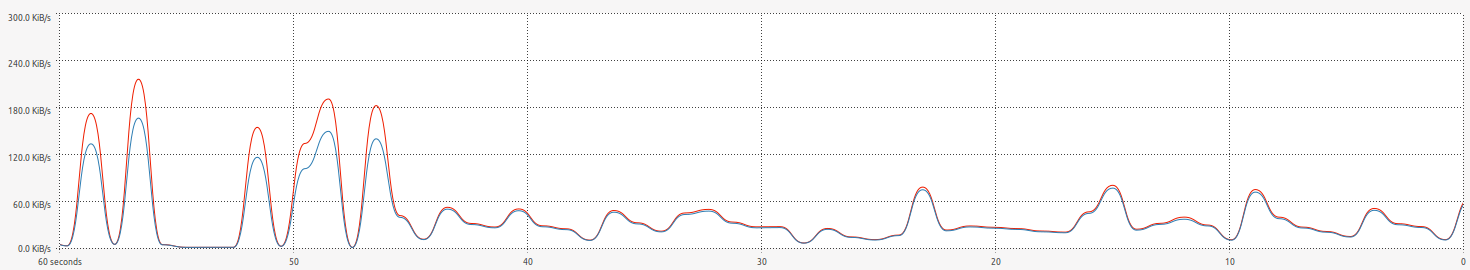
CMD["/usr/sbin/sshd", "-D"]

When it comes to ssh encryption protocol, what we need is exchange public key. The daemon program in client is designed to do this, it send the public key to container’s file named “/root/.ssh/authorized\_keys”. And client and server can communicate with X11 commands encrypted by SSH. When we use the X11 commands directly, it’s much easy for AntOS to encrypt because of lower data size.

When the software-providing server has been selected, the cluster-controlling server will send back the ip and port of container. The software-providing server share the same ip address with Docker, and it will allocation free port to container. So when container is built, the client can get the ip and port of container through cluster-controlling server. By run the command like this:

*ssh –i [the directories of privacy key] –o UserKnownHostsFile=/dev/null –o StrictHostKeyChecking=no -Y -X root@[the IP address of container] -p [the port of container]*

And then client can be connected with the software-providing server.



*fig.8. the bandwidth of typical scenario*

Fig.8 is the bandwidth of typical scenario. The horizontal axis is time axis and the vertical axis records the number of bandwidth. The red line presents input bandwidth while the blue line presents the output bandwidth. As we can see, during the whole use procedure, we bandwidth is no more than 200KB/s. The first peak happen when we start LibreOffice, then when we are typing, the bandwidth is no more than 50KB/s. And when we open Firefox the second peak happen again. What is deserve to be mentioned, VNC is based on incremental text transmission, so the condition of bandwidth is similar with AntOS, but it is much higher than AntOS at the same graph quality.

**3.2. Responsive-first load balancing strategy**

The this section we will introduce the load balancing strategy of AntOS. Deserving to be mentioned, the load balancing strategy this article discusses is not like network load balancing that among several same virtual machines or containers, like OpenStack Neutron. But its train of thought can be taken into consideration. The load balancing strategy is based on the statements related to 3 parts of each server: cpu occupancy rate, RAM usage, network load. These message can be collected into a triple (c , r , n) in order(We name it server statement triple for convenience). The triple would be calculated like table.1.

*Table.1. The calculation of triple*

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

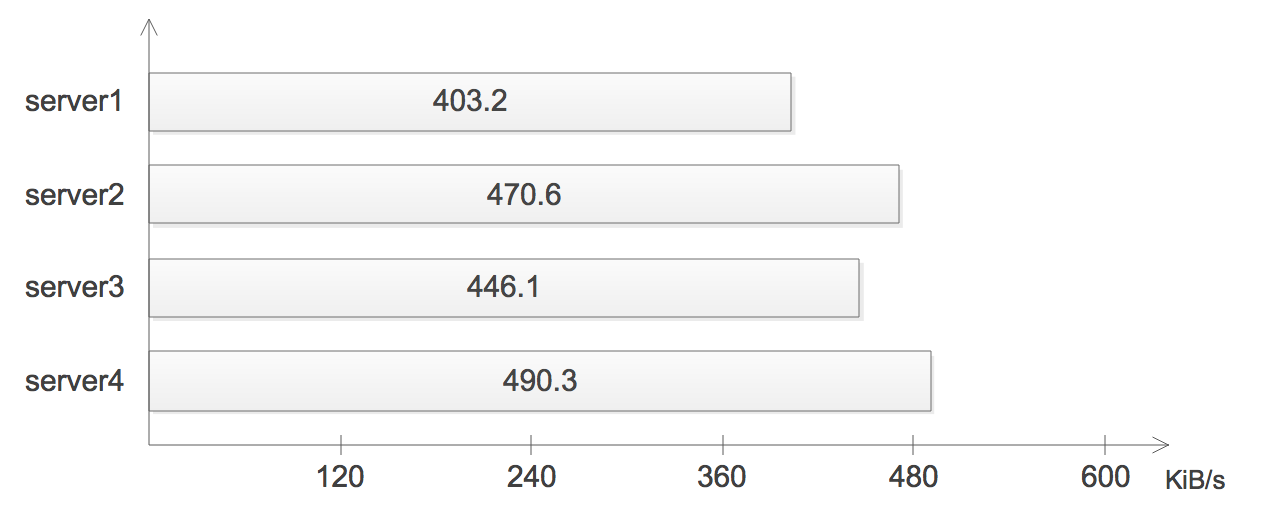
A monitor program run in each server for these information. It collects each information every minute and get a weighted average between new number as well as old number by 9:1. Because each container occupies different CPU performance, RAM as well as network, so if we only focus on the amount of software run in each server and use round robin strategy cannot take actual situation into consideration.

When user need to run a software, he will send http request to cluster-controlling server. This server will send request for the statements of each software-providing server, and triple include statements of CPU, RAM, network will get together in cluster-controlling server.

Because AntOS has a responsive-first load balancing strategy. So each element of server statement triple don’t have equal weight, so we need to calculate server statement score like this:

Higher score means higher load. So cluster-controlling server will select the server with lowest score to run the software or container that our user need. We set the network quality to the highest level, to promise a responsive-first strategy.

According to our test, we run several containers in our system, the bandwidth of each software server is shown in fig9.



*fig.9. the bandwidth in AntOS’s load balancing strategy*

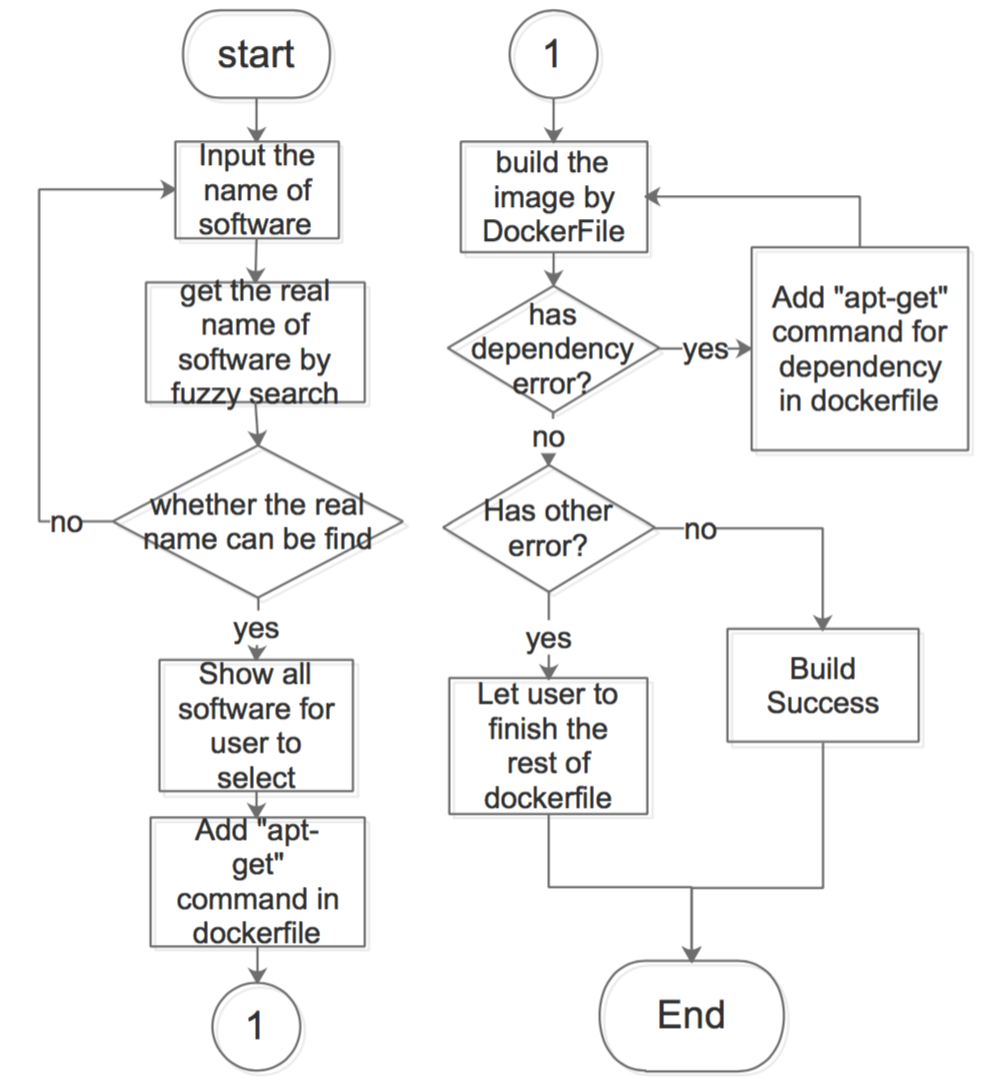
So we can get the conclusion that if we have enough server, the load of band width will be allocated to different server and the network quality of each software, or container, will be promised.

**3.3. Dockerfile intelligent generation tool**

Because we choose Docker to divide server into several part for different users to install the software they need only by the name of software.

Dockerfile-writing tool is a part of Qt interface program. It is made by web technology.

To make the generation of Docker file more convenient and high efficient. We shouldn’t let our user write dockerfile by theirself, even our user don’t need to know the existence of dockerfile.



*fig.10. the procedure the Dockerfile-writing tool works*

The whole procedure the Dockerfile-writing tool is shown by fig10. Firstly, because of the reason that users usually don’t provide complete and precise name of software they need, we should help them to get the real name of software by fuzzy search. For example, if someone need software named “vim”, but there are many different software has similar software related to “vim”, like “vim-common”, “vim-dbg”, “vim-runtime”, etc. AntOS will show all these softwares for users to select. AntOS use a Linux command like “apt-cache search [name of software]”to fuzzy search.

After AntOS can get a real name of software, A command “apt-get install –y [name of software]”will be add to dockerfile.

Then we need to solve the problem with dependency of software. When an image of Docker is build, the errors of dependency will show. We use the regular expression to catch what dependencies we need and add command like “apt-get install –y [name of dependency]”to dockerfile.

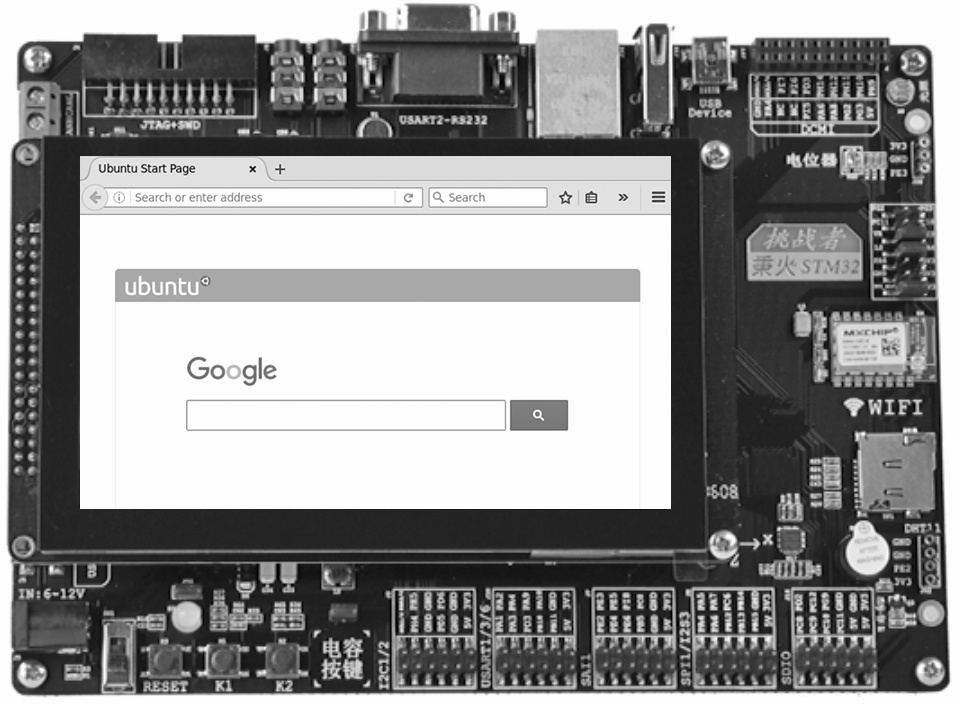
If the error is not related to dependencies, the error will be shown to user, and AntOS also provide graph interface to let user write dockerfile theirselves.

**4. System Implementation and typical test**

In this section, we will introduce the implementation of AntOS both in software and hardware.

**4.1. Hardware**

AntOS can be used just by a displayer and a low-cost hardware like fig11.



*fig.11. a typical instance of AntOS in thin PC with screen*

Because all computing processes happen in application-providing server, We can used desktop software with the hardware doesn’t have much computing ability.

The hardware of AntOS can be separated into two kinds, one is a thin PC with a screen(fig.10) like smart mobile phone. Because the client hardware only need to receive data from internet and show the GUI of remote software locally. So this device can run desktop-class applications, which is much better than traditional smart phone. The other kind of AntOS hardware is a thin PC with VGA port, the graph of software can be shown in any screen connecting with the VGA port(fig12).

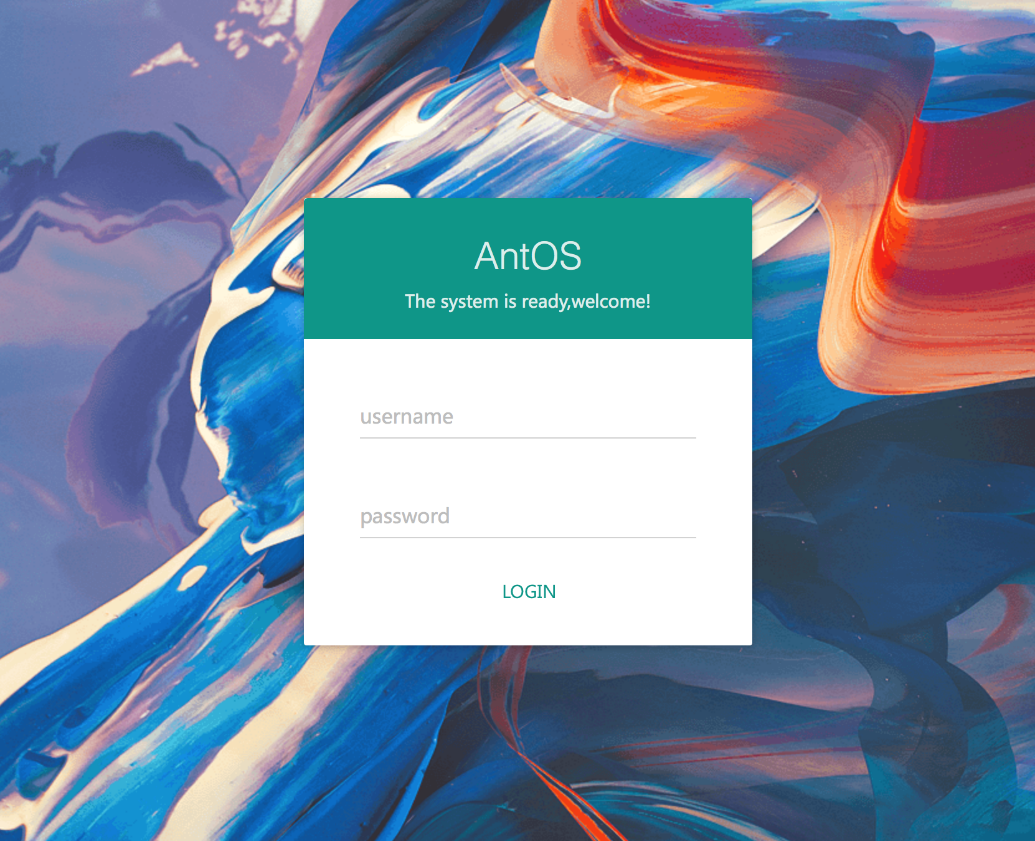


*fig.12. a typical instance of AntOS by using VGA*

It is very convenient for enterprises and educational services to deploy the whole system.

**4.2. Software**

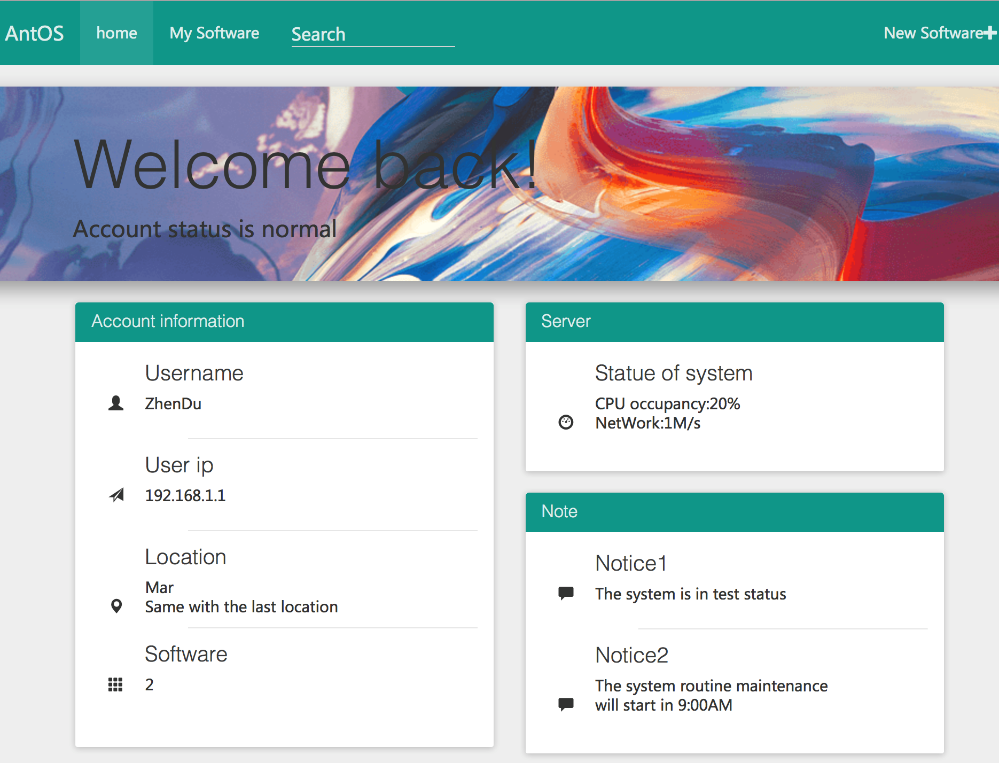
The whole system begins with interface like fig.13.



*fig.13.the login interface*

This is the login interface of AntOS, when AntOS starts running, this interface will be showed in the webview of the Qt interface program. User need to input his username and password, which will be checked online.

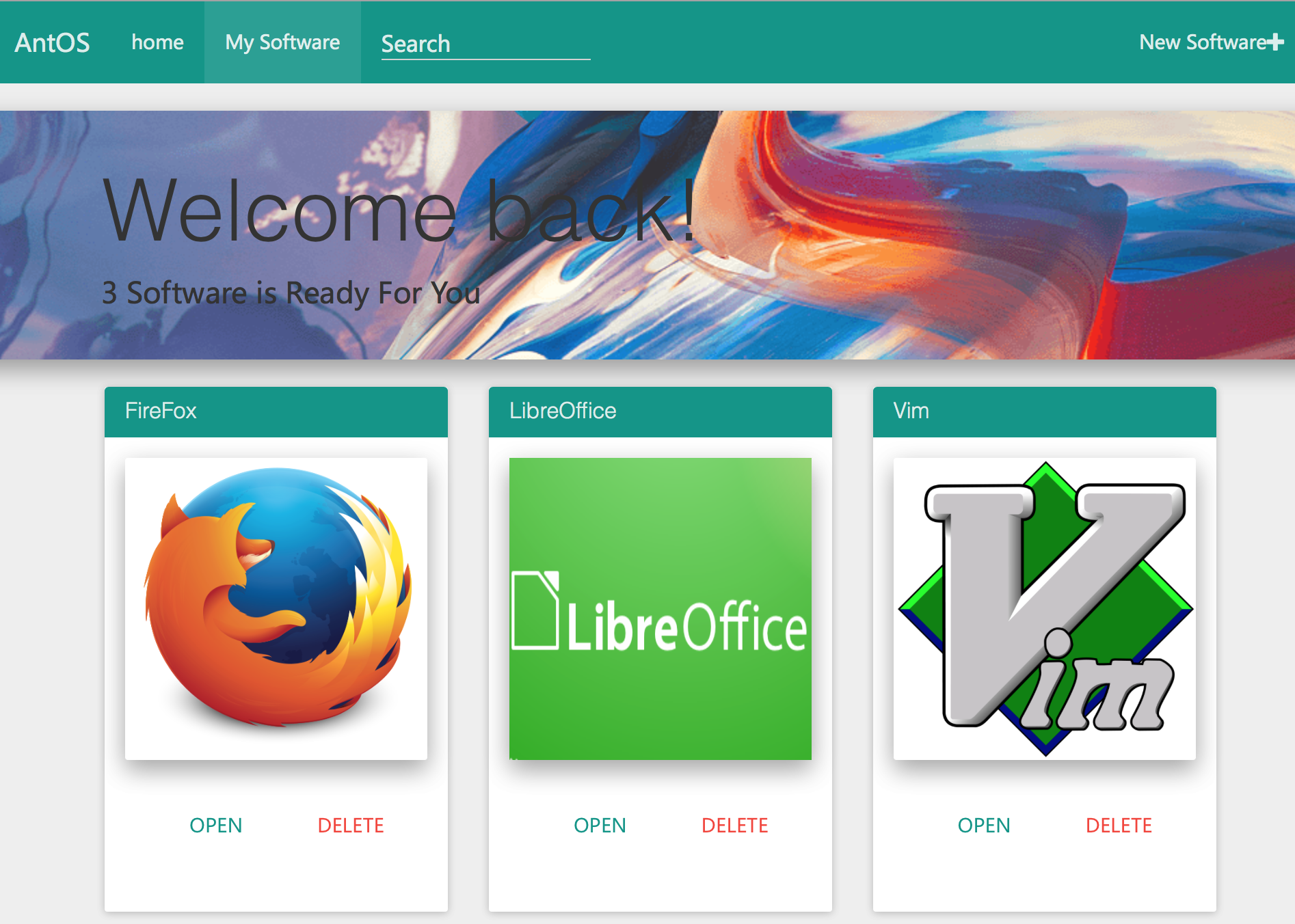
If user pass the checking, the user can go to the home page like fig.14.



*fig.14.home interface*

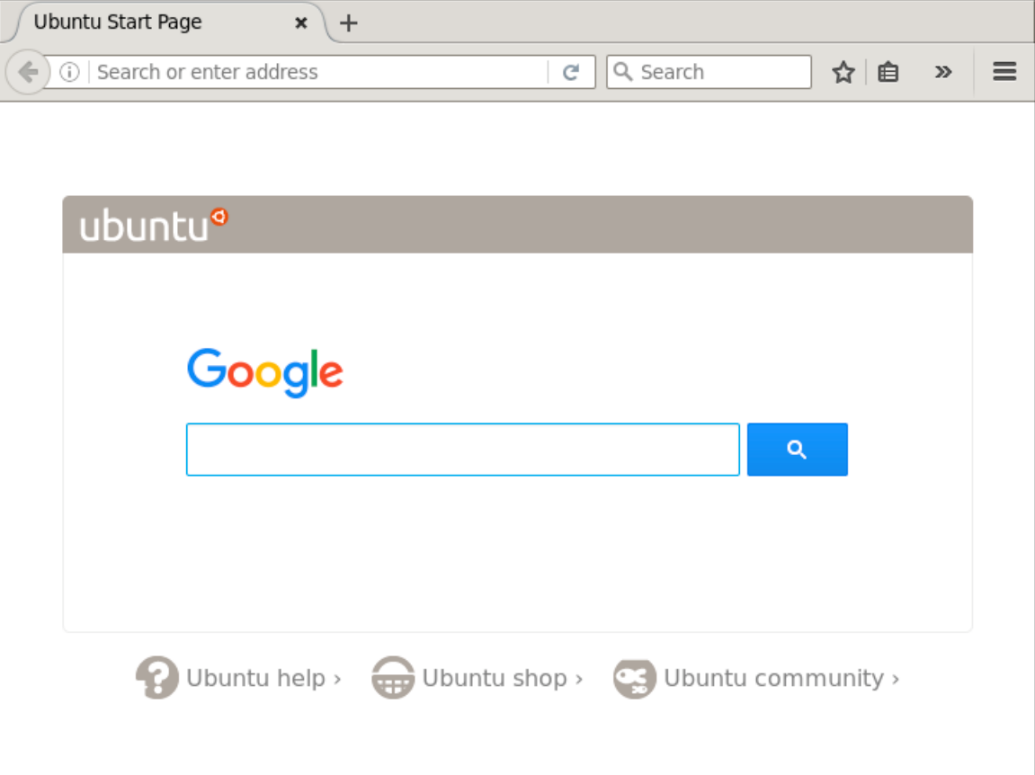
In home page, user can see the information of account and system. At the top of interface, AntOS provide button for user to other page.

When user click “My Software”, he can see all the software he has been install in cloud.



*fig.15.Software-managing interface*

User can open or delete software in this page. And when user click “New Software” to write the dockerfile with the help of “dockerfile writing tool”. When user click “open” of Firefox, X11 commands of firefox running in server will translate to X11 server in client and GUI of Firefox will be displayed in screen(fig.15.).



*fig.15.Firefox run in AntOS*

**5. Related work**

Nowadays, the designs of cloud operating system are under these two way:

1. Using the web technology, and the whole system is based on a browser.
2. Based on virtual machine and remote controlling software.

ChromeOS is the most famous cloud-based operating system, is designed to achieve a OS with low consumption as well as low cost. All of applications of ChromeOS is based on browser. ChromeOS is designed for e-commerce and education. Because of the reason that the system is only need to support a browser to run, so the client of ChromeOS is very cheap, but it has some disadvantage: the lack of software resource and functional deprivation. And because the environment of browser is totally different from traditional operating system, the experience of software is different for users, and the transplant of the software is difficult for developer.

The way based on virtual machine and remote controlling software occupy a certain share. TransOS and its design concept------“transparent computing”, makes the system run above BIOS and under the real operating system[31][32]. ZeroPC show the desktop system in browser and run the real operating system in the server, using http request to control remotely. Not only ZeroPC, but also VCloud\_Air and SUNDE also use such method. This way makes the software run in traditional operating system can be deployed in cloud directly to be used by users. It remain the habit people using traditional operating system, and we don’t need to worry about the quantity and quality of the software. But the disadvantages are also obvious, it increase the extra performance cost and increase the load of network because of video stream.

This article analysis the problem of traditional cloud-based operating system in functionality, load of network as well as extra cost. We explain how AntOS solve these problem and how we dsesign and implement this operating system based on Docker and X11.

**6. Conclusion and future work**

In this paper, we systematically investigated the existing popular cloud-based operating system, the reveal the shortage of them:

1. Cloud operation system based on browser depend on less computing ability and less load of bandwidth. But limited by APIs of Web technology, the software of this operating system is much less functions than desktop software.
2. Cloud operation system based on virtual machine and remote controlling software has much more cost in terms of performance as well as network.

So AntOS is designed to reconciling contradictions between functionality and performance by translating the X11 commands and using Docker technology to replace the using of virtual machine. In actual environment, we find that AntOS has the same experience as the traditional operating system because AntOS can provide lossless graph of software running remotely.

AntOS also need some future work. According to our test, when it comes to the pictures showed in the software UI, AntOS always causes much more load of bandwidth, which is worse than VNC, so image need to be special treatment in AntOS. And we think the further compress of X11 commands is possible. So we need a more radical compression algorithm.

**References**

1. *Silberschatz A, Galvin P B, Gagne G, et al. Operating system concepts[M]. Reading: Addison-Wesley, 1998.*
2. *Mell P M, Grance T. SP 800-145. The NIST Definition of Cloud Computing[M]. National Institute of Standards & Technology, 2011.*
3. *Janssen M, Joha A. Challenges for adopting cloud-based software as a service (SAAS) in the public sector[C]// European Conference on Information Systems, Ecis 2011, Helsinki, Finland, June. 2011.*
4. *Seltzer M A. SCO tarantella offers new twist on an old thin-client dance[J]. Network Computing, 2000:24-26.*
5. *Rhie K. From Desktop To WebTop: Achieving True Computing Freedom, Anytime, Anywhere.[C]// Webnet 2000 - World Conference on the WWW and Internet, San Antonio, Texas, Usa, October 30 - November. 2000.*
6. *Patel L, Singh G, Gupta R. LINUX BASED CLOUD OPERATING SYSTEM[J]. International Journal of Engineering Innovations & Research, 2012, 1(1):18-22.*
7. *Wikipedia. Virtual Network Computing[EB/OL]. https://en.wikipedia.org/wiki/Virtual\_Network\_Computing.*
8. *StackExchange. Difference between vnc and Windows Remote Desktop[EB/OL]. [2010-5-5]. http://superuser.com/questions/137797/difference-between-vnc-and-windows-remote-desktop.*
9. *Yue M A, Huang G. Research on Application Virtualization Based on Docker[J]. Computer Engineering & Software, 2015.*
10. *Docker. DockerDocs[EB/OL]. [2016-11-8]. https://docs.docker.com.*
11. *Wikipedia. X Window System[EB/OL]. [2016-10-30]. https://en.wikipedia.org/wiki/X\_Window\_System.*
12. *The Linux Information Project. The X Window System:A Brief Introduction[EB/OL]. [2016-3-29]. http://www.linfo.org/x.html.*
13. *Sefraoui O, Aissaoui M, Eleuldj M. OpenStack: Toward an Open-source Solution for Cloud Computing[J]. International Journal of Computer Applications, 2012, 55(3):38-42.*
14. *Corradi A, Fanelli M, Foschini L. VM consolidation: A real case based on OpenStack Cloud[J]. Future Generation Computer Systems, 2014, 32(1):118-127.*
15. *Goldberg R P. Architecture of virtual machines[J]. Proceedings of the Workshop on Virtual Computer Systems, 1973:74-112.*
16. *Smith J E, Nair R. The Architecture of Virtual Machines[J]. Computer, 2005, 38(5):32-38.*
17. *Choudhary A, Rana S, Matahai K J. A Critical Analysis of Energy Efficient Virtual Machine Placement Techniques and its Optimization in a Cloud Computing Environment[J]. Procedia Computer Science, 2016, 78:132-138.*
18. *Fink J. Docker: a Software as a Service, Operating System-Level Virtualization Framework[J]. Code4lib Journal, 2014(25).*
19. *Rami, Rosen. Linux Containers and the Future Cloud[EB/OL]. [2016-11-8]. http://www.haifux.org/lectures/320/netLec8\_final.pdf .*
20. *Crashcourse, Wiki. Docker Dockerfile[EB/OL]. [2015-9-4]. http://www.crashcourse.ca/wiki/index.php/Docker\_Dockerfile.*
21. *TheNewStack. Understanding the Docker Cache for Faster Builds[EB/OL]. [2014-7-16]. http://thenewstack.io/understanding-the-docker-cache-for-faster-builds/.*
22. *Kandhil N, Kumar A. “A STUDY ON SECURE SHELL (SSH) PROTOCOL”[J]. International Journal of Computer Science & Management Studies, 2011, 11(02).*
23. *Nguyen M V, Choe S, Huh E N. An efficient mobile thin client technology supporting multi-sessions remote control over VNC[C]// IEEE International Conference on Computer Science and Automation Engineering. 2012:155-159.*
24. *VNC. Virtual Network Computing[EB/OL]. [2016-11-8]. http://virtuallab.tu-freiberg.de/p2p/p2p/vnc/ug/howitworks.html.*
25. *Richardson T. Virtual network computing[J]. IEEE Internet Computing, 1998, 2(1):33-38.*
26. *Scheifler R W, Gettys J. The X Window System.[J]. Software Practice & Experience, 1986, 5(2):79-109.*
27. *Christophe Tronche, ch@tronche, com. Event Structures[EB/OL]. [2016-11-8]. https://tronche.com/gui/x/xlib/events/structures.html.*
28. *Robert, W, Scheifler. X Window System Protocol[EB/OL]. [2016-11-8]. https://www.x.org/releases/X11R7.6/doc/xproto/x11protocol.html.*
29. *Wiegand T, Sullivan G J, Bjontegaard G, et al. Overview of the H.264/AVC video coding standard[J]. IEEE Transactions on Circuits & Systems for Video Technology, 2003, 13(7):560-576.*
30. *Christophe, Tronche, ch@tronche, com. The Xlib Manual[EB/OL]. [2005-10-31]. https://tronche.com/gui/x/xlib/.*
31. *Zhang Y X, Zhou Y Z. A new cloud operating system: design and implementation based on transparent computing[J]. Acta Electronica Sinica, 2011, 39(5):985-990.*
32. *Zhang Y, Zhou Y. TransOS: a transparent computing-based operating system for the cloud[J]. International Journal of Cloud Computing, 2012, 1(4):287-301.*