Cloud System

Chihuang Liu, Jiashuo Wang

5100719070, 5100309436

Shanghai Jiao Tong University

Abstract

We introduce a new system to replace the current form in cell phone, tablets and even computers. With the new system, our devices have no need to be equipped with high-level of CPU, GPU, large memory or other advanced hardware anymore, which is quite power-saving so that the devices will surely be used for a longer time. Thus the terminals can be thinner and lighter and even just like a page of paper. Users have no need to update manually, because if the developer updates his application on @cloud system, all things presented to the user are updated, that is to say, developers have no need to worry about developing different version on different devices with different hardware. And in the fourth part of our report we will demonstrate how can we achieve this in detail technically, and in the last part we will show our system to realize this. And the results of our test is very well, and our system runs smoothly.

I. Introduction

Nowadays people's requirements for mobile terminals is keep increasing, as more multi-media services such as large scale 3D games that need much rendering job, watch high definition videos that need onerous decoding, or use some applications that need relative complex calculation. For these kind of reasons the manufacturers are providing consumers more and more powerful mobile terminals with higher level of hardware. The cores of CPU are more, which is up to eight now, and the frequency of CPU are higher, and RAMs are larger to guarantee a smooth experience. This will definitely bring us much trouble and the first one is the price of these terminals. Mobile phones are much more expensive than before thus it is harder for us to afford a satisfying one. And secondly the endurance is worse. Previously our mobile phones can be efficient for a week without a charging, but now my phone has to be charged everyday which annoyed me very much.

But where is the end of this kind of accumulation of hardware? Are we going on this way until the end of the world? Fortunately in 2006 the CEO of google, Eric Schmidt, put forward a concept, cloud computing, in SES San Jose 2006, which point out the direction of future development. Cloud computing is a style of computing in which dynamically scalable and offer virtualized resources are provided as a service over the Internet. And it is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

There are many cloud services now, such as iCloud of Apple, the virtual server offered by Amazon, and the virtual lab offered by Skytap. All these services more or less realized the vision of cloud, but none of them formulize a complete and practical system. The latter two are both offered to advanced developers, who account for a small proportion of users. The only service for common consumers is cloud storage, which does not make big changes.

What we have done is a cloud system for major consumers to use in their mobile terminals in everyday life, and can change the formation of such terminals. Briefly speaking, we are doing most of calculating and rendering job on our remote server, which communicate with mobile terminal through the Internet, and the only job for the terminals is to display compressed picture it received. In this way terminals do not need to have powerful hardware at all, because their job is very easy.

II. System Design

Our main idea is to shift the job of terminals to our servers, and to separate display unit from background device. Because what we really need for out mobile terminals is not the powerful performance, but the fluency experience when we use them. So the performance demand in the terminal is relative low with most job done in the servers. Because servers need not to be moved and not to consider about volume and battery, so they can be very powerful. In this way the pressure for mobile terminals with high demand for mobility and endurance is less. When the user want to use his terminal to finish a job, he just tell the cloud what he want by touching the screen or speaking to his terminal, and the cloud will know how to achieve his goal using related algorithm. And after the cloud has finished, it will send him back some pictures that are compressed in order to enhance transmission efficiency. Then his terminal will display these pictures to show you the results.

In our system, the experience is totally the same with that on common mobile terminals if enough network speed is guaranteed, but we still have many advantages, for example because your terminal need not to calculate and consider how to arrange the interface, so the terminal will not be ąřtiredąś after long time of running, so you will never feel your terminal too hot in the summer days.

III. Benefits

In the statements above, we can learn that using our cloud system, our mobile terminals can be designed very light and thin, which can only have a piece of screen, small CPU and GPU. Without heavy work of calculation, our terminals will be more power saving so the battery can be smaller. So much concepts can be realized such as products showed in figure 1. Not only

Figure 1: *Future Terminals*

hardware of terminals will be changed thoroughly, the concept of ąřdownloadąś an application will disappear because of our system because we never need any app to be stored in our terminal any more, all of your information is saved in the cloud and no data need to be stored in your device. When we want to use some specified function, we can just get it directly from the cloud. All applications are stored in the cloud and the cloud knows how to run them. In that time there is only a "menu" application in our terminals, and we enter it and choose some application to run. What we need is just kind of license, to enable us to access some data. Another benefit is to provide developer with easy control over their app, he can simply update his app in the cloud and all users could use the new version of app in the same time.

Figure 2: *Apps in the Cloud*

IV. Principle in Detail

As is discussed above, it is quite easy for users to use and for developers to update the application. There will be no concept of update for users; that is to say, users will always use the newest version of an application instead of caring about updating to the new version manually. To achieving this goal, we have to create a new way to design all the applications, which is suitable to apply on our @cloud system.

How the design will go on is depended on the habit of users when they use the cloud applications or games. Now letąŕs imagine together. When the user enters the application on his mobile device, the device will send an enter request to @cloud automatically. And an associated thread will be created in the cloud application process on @cloud system, which is used to deal with all the operations of this user. Obviously, the cloud application process will contain lots of threads to support each user to operate on their own devices. Then the initial layout will be rendered and compressed and sent to the user device. The only job the user device to do is to decompress and let the user see the interface of the application (As is shown in Figure 3). You see, most of the calculations do not take place on the user device. That is why only the basic hardware is needed on the user device instead of high-level CPU, GPU, large memory and other advanced hardware that we are pursuing today.

When the user do something with this application (For example, if the user is playing a cloud car-racing game on his device, he will control the car to let it turn left or right or even use nitrobooster), the device will send the relative command to @cloud system automatically. The thread on @cloud system will execute and generate the new layout, which will be sent to the

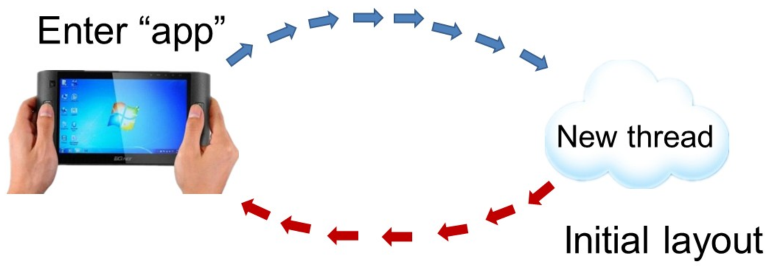

Figure 3: *Initial Layout*

user by rendering and compressing (Shown in Figure 4). As is known to all, the game UI usually

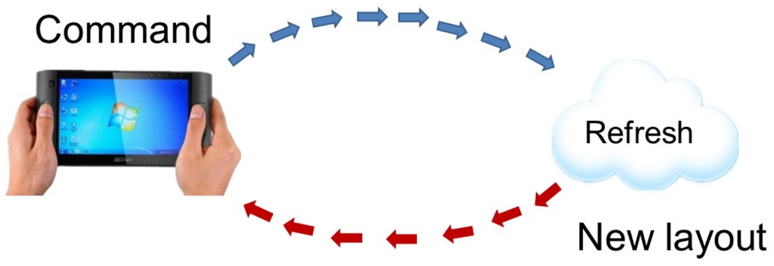

Figure 4: *Refresh the Layout*

has lots of dynamic images. In this circumstance, the layout will be refreshed automatically with some interval, which is dynamically defined by the predicted speed of the dynamic image and also the current network speed. So the user device will receive the layout every once in a while so that the display on the device screen watched by the user looks continuous (Figure 5). From

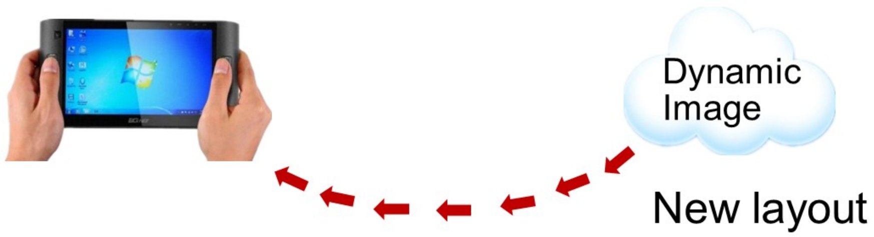

Figure 5: *Processing of Dynamic Image*

here, we can imagine the quality of the display depends on the internet speed. The higher the Internet speed is, the better the quality of our service is. So now letąŕs make a simple estimation to find out which kind of network is suitable for the cloud service. Assume we want the layout with resolution of 960×640. And each of the RGB is stored with 1 byte. According to the human visual characteristics, we think a movie to be continuous by displaying at least 24 frames of the movie per second. So we can get the total data size the @cloud system generates is,

$$D = 960 \times 640 \times 3 \times 8 \times 24bps = 350Mbps \tag{1}$$

In the 3G environment, the downlink rate v is about 10Mbps. That is to say, we should guarantee the compression ratio with

$$R = \frac{D}{v} = 35 : 1 \tag{2}$$

which is possible with the current technology of compression. For JPEG, the compression ratio can reach that. And we will choose a better compression technology with higher compression ratio. The 3G network, however, does not quite satisfy us so that we may have to sacrifice the resolution to make a trade-off. But with the more and more rapidly development of 4G LTE and Wi-Fi network, and if the network is the higher-speed 4G or Wi-Fi network, such transportation is totally allowed. When there is better network, the frames per second and the picture resolution can be higher, which will give users an effect of better quality.

When the user wants to exit the application or let the application run in background, the relative command will be also sent to @cloud system. If the user tries to exit, the associated thread on @cloud system will be killed (Figure 6). In detail, the resources this thread has occupied are

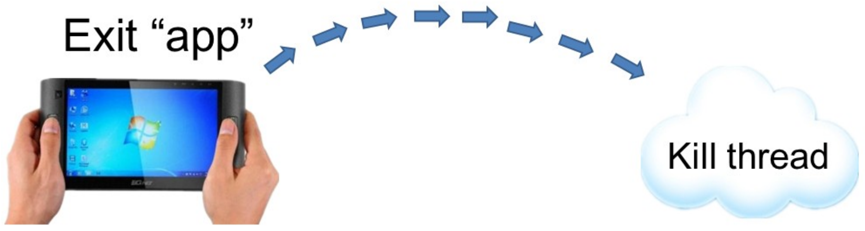

Figure 6: *Exit and Kill*

released and its user data will be written into the cloud disk sections of this user to support his next request of this application. But if the application is not terminated but runs in background, the associated thread will only be suspended and even swapped out of the memory of the @cloud system. Next time the user get back to this application, the data and status will be written in the cloud memory again and begin to execute, which finally refreshes a new layout to transmit to the user device to display.

According to the life cycle of the cloud application descripted above, we will have a clear understanding why the programs are only running on the @cloud system and we do only need to write the programs for @cloud system. With the unified interface on user device, every command sent by the user device will show us how the user is operating. Then the thread of this program on @cloud system will calculate the next status and layout according to such operations. As with the communications between user device and @cloud system, the socket is the easiest way to achieve our goal. Berkeley sockets (or BSD sockets) is a computing library with an application programming interface (API) for internet sockets and Unix domain sockets, used for inter-process communication (IPC). Besides the socket API, we can also use the STREAMS-based Transport Layer Interface (TLI) API to implement the communications between user device and @cloud system.

V. Simulation and Result

As the game is the one that needs the most calculations, we choose to make a simple game to prove our thought. In our simulation, we regard an Android phone as the user device (although we don't need its ability of strong calculation. The only thing we want the Android phone to do is to decompress the layout and display it) and the PC as the @cloud system server. The communications between user device and @cloud system server is directly via Wi-Fi (it's kind of like Figure 7). We use Java to make these programs. And socket is used

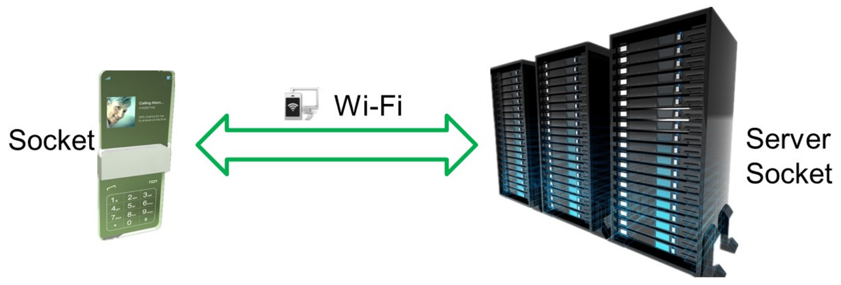

Figure 7: *Program with Socket*

when communicating between @cloud and devices.

For the user device program, which is the Android program here, we just add the functions of detecting which operation the user is doing and also decompressing. As soon as the program finds out what the user wants to do, it will send a command with Java socket. When the program on PC, which plays the role of the thread created in the application process on @cloud system, receive the command as the Java socket server, it will calculate what the next layout will be and then generate and compress the new layout, which will be sent back to the Android phone.

We made a game similar to sokoban. On the PC, we run the Java program CloudGameServer, which is used to deal with the operation sent by the user device and judge when the user wins the game. Part of the thread running on PC cloud server (@cloud system) is Algorithm 1. And

Algorithm 1 Thread on @Cloud System

```
1: Create an object of serversocket to listen to the port.
2: while (True) do
3:     while (Accept the request sent by the user device) do
4:         if (user presses the Up button) then
5:             Judge whether the hero cross the border.
6:             If not, do the up operation.
7:         else
8:             Go on the similar operations with Down, Left, Right button like Up button.
9:         end if
10:        Write the compressed layout to the output stream.
11:        Send the compressed layout data.
12:    end while
13: end while
```

on the Android phone, user device here, we run the Android program CloudGameDemo, which is used to send the operation command to the PC server and deal with the layout the device itself receives. Part of the thread running on Android phone is like Algorithm 2.

Algorithm 2 Thread on User Device

```
1:  while (True) do
2:      Keep listening to the IP of the PC server.
3:      while (There is some data in) do
4:          Load the data in the input stream to the memory.
5:          Decompress and display the layout.
6:      end while
7:      while (Detect some button is pressed or some area of the screen is hit) do
8:          Generate relative command according to the unified interface.
9:          Send the command data with socket.
10:     end while
11: end while
```

When running the programs on both PC and Android phone, you can play this simple game on the phone (in Figure 4-2). As we use Wi-Fi to transmit, the speed of the Wi-Fi is the factor that counts. In our dormitory, the strength of Wi-Fi is very strong. The latency cannot be observed. But in the classroom the speed of Wi-Fi is not so fast as in dormitory. So there is some latency, which, however, is far less than 1 second. Although it is a simple simulation, we can still get the conclusion that it really works. With the development of network, the speed will be much higher. So @Cloud system will play a more and more important role.

VI. Conclusion

In this paper we introduce a new form of user device which need only basic hardware. Both users and developers can benefit from the @cloud system. Life becomes more convenient when we use the cloud applications and program on @cloud system. We then introduce how to implement it with our demo simulation. The result shows that it is feasible and easy to implement. In the end we will conclude our report with our carefully designed logo.

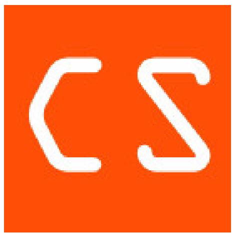

Figure 8: *Cloud System*

References

[1] Reto Meier (2010). *Professional Android Application Development*.

[2] Bruce Eckel (2007). *Thinking in Java(4th Edition)*.

[3] Cay S.Horstmann,Gary Cornell (2006). *Core Java 2, Volume II–Advanced Features (7th Edition)*.

[4] Weimin Zhang, Jianfeng Tang (2009). *Cloud Computing: A profound change in the future*.

[5] David S.Linthicum (2010). *Cloud Computing and SOA Convergence in Your Enterprise*.