Android as a Server Platform

Masashi Toyama, Shunsuke Kurumatani, Joon Heo, Kenji Terada, and Eric Y. Chen
NTT Information Sharing Platform Laboratories,
NTT Corporation
3-9-11, Midori-cho, Muasashino-shi, Tokyo, 180-8585, Japan
toyama.masashi@lab.ntt.co.jp

***Abstract-* The number of smartphone users and mobile applications are growing rapidly. Though smartphones are expected to have PC-like functionality, hardware resources such as CPUs, memory and batteries are still limited. To solve this resource problem, many researches have proposed architectures to use server resources in the cloud for mobile devices. We propose a conceptual architecture of Android as a Server Platform, which enables multiple user Android applications on cloud server via network. Though Android is mainly designed for physical smartphone, Android's two other features are useful to construct a server platform. --Android is open-source product and runs on an x86 CPU. We show three types of multi-tenant architecture for an Android server platform and discuss the direction to take to it reality.**

Keywords-Android; Virtualization; Multi-tenant; Cloud;

I. INTRODUCTION

The number of smartphone users and mobile application are growing rapidly. According to a recent report, 45 million people in the U.S. own smartphones and 234 million people subscribe to the mobile phone application stores[1]. There are several mobile Operating Systems (OSs), such as Symbian, iOS, Android, and Windows Mobile. Because thousands of application developers construct many kinds of applications for these platforms, users can easily enjoy their individual smartphone lifestyle.

Though smartphones are expected to PC-like functionality, hardware resources such as CPUs, memory, and batteries are still limited. Therefore, many application developers are forced to take into account these limitations. To solve this resource problem, some researches have proposed using server resources in the cloud for smartphones.

From this background, we propose Android as a Server Platform that enables many users to use resources on remote cloud servers. We discuss our motivation to adopt Android as a server OS as follow. Using a mobile OS enables the reuse of many mobile applications that is designed to be used on smartphone interfaces, such as software keyboards, touch panels and many sensors. Since a resolution of mobile OS is small, it is better to use a remote application via a network than a desktop OS. Android is an open-source mobile OS initiated by Google. The main reason to use Android as a server platform is that it is able to run not only for smartphones but also for the x86 platform including servers.

We propose a multi-tenant architecture of Android as a Server Platform. Section II describes the background and motivation of using mobile applications running on a server. Section III discusses related work. Section IV clarifies the architecture type for using Android multi-tenancy, and technical approaches to make it a reality are discussed in Section V. Section VI concludes this paper.

II. MOTIVATION

Mobile Application Platform on Cloud Server

As a numbers of service providers such as Dropbox[2] and Zumodrive[3] provide online storage services, the architecture for remotely using mobile application on server has many benefits for users. This approach, called Mobile Application Platform on Cloud Server, intends to handle not only user data but also user applications in a cloud server. This approach changes the application lifecycle as follows. "Write once, run everywhere. Install once, use everywhere." Figure 1 illustrates an overview of the concept. By executing a mobile application in the cloud server, users and developers free from device limitation such as CPU power, memory, and battery, and from device software environment such as OS or version. Moreover, once a user installs an application on the cloud server, she/he can use the application anywhere, an any device.

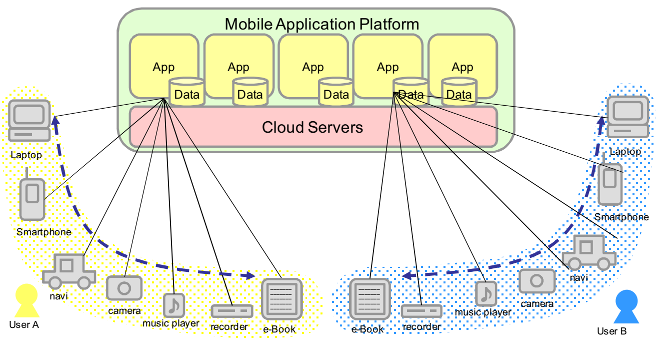

Figure 1 Mobile Application Platform

Multitenant for Android

Multi-tenancy, which means that software running on a server provides services to many users, is one of important features for cloud computing. From the viewpoint of both economy and ecology, it is beneficial to share hardware resources among users. Using a mobile OS would be more effective than using a desktop OS because the resource requirements of mobile OSs are smaller. However, to the best of our knowledge, there is still no service that uses Android as multi-tenant system. We discuss the multi-tenant architecture for Android and how to construct it.

III. Related Work

Integration of Mobile device and Cloud

Researches have proposed integration between mobile devices and cloud computing. Satyanarayanan et al.[4] outlined their vision of allowing mobile users to seamlessly use nearby computers to obtain cloud-computing resources by instantiating a "cloudlet" that rapidly synthesizes virtual machines on a nearby infrastructure that can be accessed through a Wireless LAN. Canepa et al.[5] presented a framework named "Ad Hoc cloud providers". At this framework, mobile devices can execute their jobs using other device resources around them as if it is executed on one cloud server.

Our approach is closely related to that of Chun and Maniatis[6]. They proposed the creation of clone VMs to run mobile applications as if they were running on mobile devices. They recognized five categories of augmented execution to speed up mobile applications, namely Primary, Background, Mainline, Hardware, and Multiplicity, and presented a research agenda to bring the vision into reality. Their project homepage can be found in [7]. Our multi-tenant architecture for Android can be seen a specific study of Multiplicity.

Multi-tenancy

Royon et al. proposed multi-user, multi-service execution environment named "virtual service gateway"[8]. They classified existing multi-application environment approaches by modifying Java runtime, and proposed an overlay approach to run virtually original application. As modifying approach has advantages of performance and isolation, overlay approach has advantages of usability on a standard Java Virtual Machine. Bezemer discussed the direction of multi-tenancy[9]. They recognized five features of a multitenant platform, namely Performance, Scalability, Security, Zero-Downtime and Maintenance, to prevent maintenance nightmare. We discuss and evaluate a proposed architecture based on some of these features.

Virtual Smartphone over IP

Beyond constructing a mobile application platform, we have previously proposed a proof of concept prototype implementation named "Virtual Smartphone over IP" [10]. An overview of the implementation is shown in Figure 2. In this prototype, Android-X86[11] is adopted on a mobile server OS running on a hypervisor. The client program installed on a physical smartphone can remotely interact and control Android-x86 images. The client program transmits various events from the physical device not only the keyboard but also the touch screen and various sensors such as GPSs, accelerometer, and thermometers, to the mobile server OS and receives graphical screen updates from it via Virtual Network Computing (VNC). These programs enable to use server side virtual mobile OS applications as if it is running on a physical smartphone.

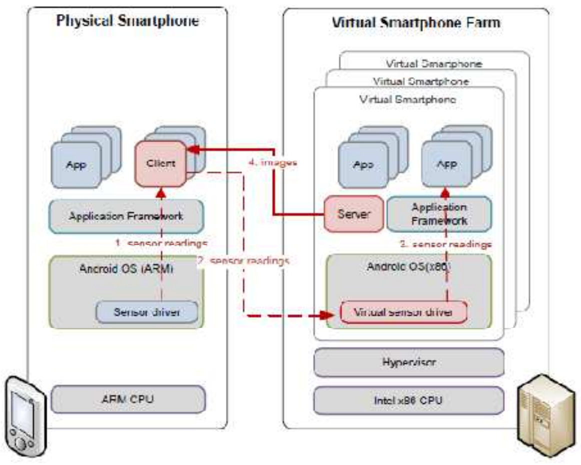

Figure 2: Overview of Virtual Smartphone over IP system

The performance evaluation using a common smartphone and a server shows that our virtual smartphone on a server is at least 10 times faster than on a physical smartphone.

IV. Multi Tenant Architecture for Android

This section discusses the process to construct multi-tenant architecture for Android based on related work. Figure 3 shows an overview of the architecture Android on a Server. Android system info is drawn upon Google's "Application Developers" document[12]. We discuss the three types of approach, hypervisor-layer, kernel-layer, and framework-layer, for multi-tenant architecture.

The hypervisor-layer approach shown in Figure 3-(i) uses the Virtual Smartphone over IP system as already stated in Section III. Each user owns her/his Android OS image on a server and freely runs her/his application in a separate VM Multi-tenancy is achieved by running multiple users VMs in a server via a hypervisor.

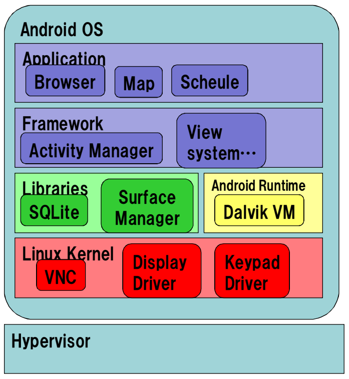

Figure 3: Overview of Android on a Server

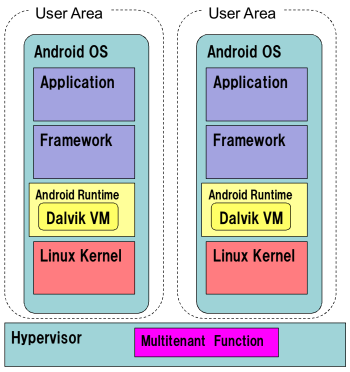

Fig.3-(i) Hypervisor-layer approach.

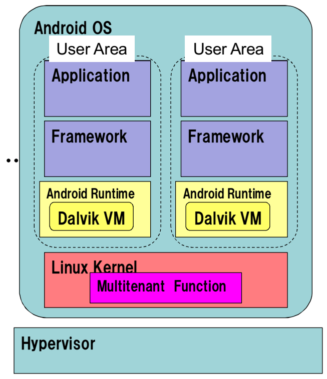

Fig.3-(ii): Kernel-layer approach

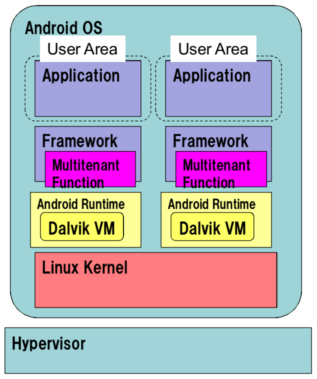

Fig.3-(iii): Framework-layer approach

This approach has the advantage of application usability and maintenance. From the viewpoint of application usability, every mobile application that can run on Android-x86 is usable because each Android OS runs only one application. The maintenance is focused about OS upgrade. Android has different versions and version up is currently on going.

The second approach implements multi-tenant function in kernel-layer, as illustrated in Figure 3-(ii). This approach changes Android OS to run multiple user applications in separate processes. This approach is similar to an ordinary thin client server running multiple user applications in a server. The main challenge is that original Android supports only one display and keypad device since Android is mainly designed to work on a smartphone.

Another approach is to create a multi-tenant function at framework-layer, similar to existing a Java-based multi-tenant framework. As shown figure 3-(iii), this approach remodels Android the framework and APIs to support multiple user applications. The main challenge is how to run existing Android applications in modified framework.

The quantitative evaluation of these three types of architectures is illustrated in Table 1. As shown in[10], the hypervisor-layer approach is feasible and good for maintenance. However, it seems to have a scalability limitation caused by a hypervisor. Because each VM try to separately maintain their resources, it is difficult to control unused resources. The other two approaches have an advantage in scalability but have a disadvantage in maintenance because they change the Android OS. From the viewpoint of running existing application, the kernel-layer approach is better because it does not changes Android runtime environment. Moreover, we assume that the kernel-layer approach is easy to develop because Android is implemented based on the Linux kernel so that can support multiple displays, keypads, and applications.

The next section, we focus the kernel-layer approach and show research agendas about how to Android OS support multitenant.

Table 1. Quantitative evaluation of proposed architecture

Multi-tenant type	**Scalability**	**Application Usage**	**Mentenance**
Hypervisor	×	○	○
Kernel	○	○	×
Framework	○	×	×

V. Research Agendas

In this section, we discuss the major research challenges to construct multi-tenant functions for Android, and the approaches we are currently taking to build this architecture. Most of the difficulty is related to the fact that Android is not designed to share resources among users.

A function overview of the architecture is illustrated in Figure 4. We define two new functions for enabling multi-tenant for Android. The first function is the multiple application controller installed in an Android OS, and the second is the user area manager located in a host OS. The multiple application controller enables running of multiple applications as if each application is running on independent physical Smartphone. It is important requirement to decrease implementation cost for Android OS because of maintenance about OS version up problem. The user area manager controls server resources and acts as an interface between a terminal and the multiple application controller.

A common sequence of application launch is illustrated in Figure 5. When user wants to use an application, the user terminal contacts the user area manager and order to launch application. The user area manager checks the server resources and select which guest OS to run application. The multiple application controller launches the application based on a order from the user area manager. The user area manager returns VNC connection information such as IP address and port.

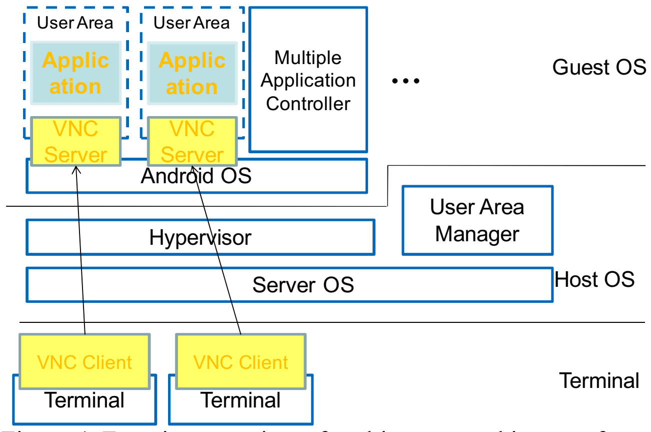

Figure 4: Function overview of multi-tenant architecture for Android.

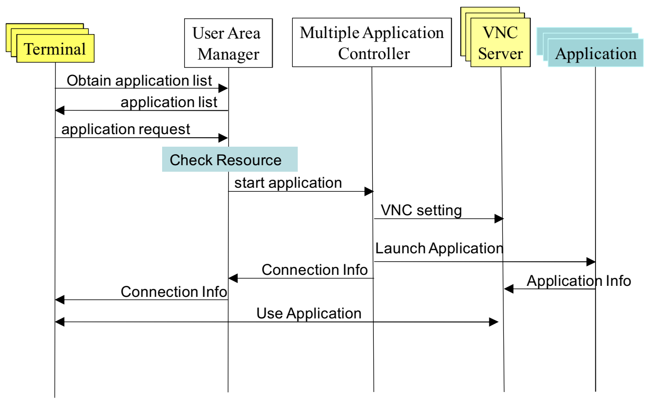

Figure 5: Sequence overview of Application Launch

The reasons we still run Android OS on hypervisor is following.

- **Maintenance:**
  An OS image is easy to backup, restore, and check the server environment similar to other hypervisor-based virtualization.
- **Application Environment:**
  The Android OS has backward compatibility regarding application runtime. For example, an application that is made for Android 1.6 runtime can work in Android 2.1 runtime. However, device information such as sensor, keyboard, and display size, varies from one physical device to another. Running Android OS in various setting is simple method to keep application environment.
- **CPU management:**
  The Android is not designed to run an a multiple core environment. Therefore hypervisor's CPU allocation is important to strengthen the efficiency of CPU power.

Discussion on Multiple Application Controller

Multiple application controller is an enabler to run multi user applications. The discussion is focused about device support for multi user and how to run application.

Multiple device support for Android Application

In the virtual smartphone on IP system, the VNC protocol is used to upload terminal events and to get application view image. Although a Linux server has multiple screen and keyboard devices, Android OS has only one screen device. Original Android has only one frame buffer located at "/dev/graphics/fb". From the viewpoint of the Android application lifecycle, only one application can use the display, and the other applications automatically run in the background process[13].

To solve this issue, multiple display device support and application mapping is required. We can search activities to make Android support multiple displays. There is an e-book reader using Android that has two displays. Moreover, the 0xdroid project[14] already has shown sample implementation of a two display device using frame buffers. We are planning to construct virtual frame buffer to support multiple foreground applications on Android.

Data Security Integration

The Android has security mechanism to protect the application data sector from other application using user account based access control. Of course Android is not designed to use same application for different users. So, if users share same application data sector, mixing of unexpected user data may occur.

There are two approaches to solve this problem. One is virtual SD card approach. If we control the all user accesses to SD card, it can prevent access to another user's application data. The benefit of this approach is that implementation impact is limited to SD card driver. Second approach is using a file system function. GNU chroot enables changing the application root directory dynamically. This approach would also prevent unexpected file access.

Running same named application on Android

An Android application is managed based on application name. For example, if you choose and touch an application to start, the Android OS calls the application based on the name like 'jp.co.ntt.vsp'. If the application is already running, the Android OS just changes the foreground application. This suggests that Android OS does not seem to run the same named application as another process.

If we solve this problem passively, we have to modify Android's application management system. This approach does not meet the requirements because we have to modify Android core system. The simplest way to solve this problem is obtain a source code and change the application name for users such as from "jp.co.ntt.vsp" to "jp.co.ntt.vsp._user1". Another solution is to keep do not to run same application by User Area Manager.

Keeping application linkage

This problem is related to run same application. Android has its own inter-application interaction function named Intent [15].An overview of Intent is described below. "*Intent messaging is a facility for late run-time binding between components in the same or different applications.*" Therefore, the Multiple Application Controller has to take into account not only one but also an interacting application.

Discussion on User Area Manager

The user area manager is an interface between a mobile terminal and the multiple application manager. When a user terminal wants to run an application, it contacts this function and obtains the information to view and control application in Android OS.

Resource Management on Guest OS

To enable multi-tenancy, kernel-layer approach requires a function to manage where and how to run mobile application based on physical device information. This function controls all application lifecycles such as install, configure, start, and stop. Implementation of this function depends on the user area manager, but we are now planning to implement it using Android Debug Bridge [16].

Physical Device Management

Since the hypervisor-layer approach adopts only one application in each VM, the kernel-layer approach runs multiple applications in each VM. Therefore, user management should be launched out of guest OS. .In this architecture, the host OS has uniform management function to authenticate and authorize a user's application request.

VI. Conclusion And Future Work

In this paper, we proposed Android as a Server Platform system that enables the use of sharing server-side Android OS among multiple users. We also showed the technical difficulty and approaches related to multi-tenant architecture for Android OS, which is originally designed to use single user. We plan to develop a prototype system about proposed multi-tenant Android architecture. We believe that proposed architecture shows high performance on virtual image-based virtualization for mobile applications.

References

1. Android Phones Steal Market Share. http://www.informationweek.com/news/mobility/smart_phones/showArticle.jhtml?articleID=224201881.
2. Dropbox - Home - Online backup, file sync and sharing made easy. , http://www.dropbox.com/.
3. ZumoDrive - Enjoy your media and documents from every device. , http://www.zumodrive.com/.
4. **M. Satyanarayanan, V.Bahl, R. Caceres, and N. Davies,** The Case for VM-based Cloudlets in Mobile Computing. : IEEE Pervasive Computing, 2009.
5. **G.H-Canepa and D.Lee** A Virtual Cloud Computiong Privoder for Mobile Devices. San Francisco : MCS'10, 2010.
6. **B.G. Chun and P. Maniatis.** Augmented Smartphone Applications Through Clone Cloud Execution.
7. CloneCloud Project at Intel Research,. http://berkeley.intel-research.net/bgchun/clonecloud/.
8. **Y.Royon, S.Frenot, and F.L.Mouel** Virtualization of Service Gateways in Multi-provider Environments. Heidelberg : CBSE 2006, 2006.
9. **C.P.Bezemer and A.Zaidman** Multi-Tenant SaaS Applications: Maintenance Dream or Nightmare? Antwerp, Belgium : IWPSE-EVOL' 10, 2010.
10. **E.Y.chen and M.Ito.** Virtual Smartphone over IP. Montreal,QC, Canada: IEEE WOWMOM, 2010.
11. Android-x86 Project - Run Android on Your PC(Android-x86 - Porting Android to x86). http://www.android-x86.org/.
12. Android Developers. http://developer.android.com/index.html.
13. Application Fundamentals | Android Developers. http://developer.android.com/guide/topics/fundamentals.html.
14. 0xdroid. http://code.google.com/p/0xdroid/.
15. Intents and Intent Filters. http://developer.android.com/guide/topics/intents/intents-filters.html.