Secure Low-Cost Platform for startup Company Using Cloud Virtualization Desktop

[1]Sri Venkateshwaraa college of technology ,Sriperambadur ,vadakal ,Kanchipuram district
[2] Vinodhini R 412621104053 BE CSE department
[3]Nagaarjuna Prasanth SK 412621104026 BE CSE department
[4]Selvi M 412621104042 BE CSE department
Corresponding author: Yamini.J 412621104054 BE CSE department
This work was supported by the Sri Venkateswaraa college of technology ,Sriperambadur

ABSTRACT Smart VmobiDesk is an existing solution that provides mobile device management and virtual desktop infrastructure capabilities. Compared to VmobiDesk, our project focuses on leveraging Amazon Web Services (AWS) to provide a more scalable, secure, and cost-effective cloud virtual desktop solution for startup companies.

This project explores the implementation of a cloud-based virtual desktop infrastructure using Amazon Web Services (AWS) for startup companies. We utilize an EC2 instance (i-0d540511613f5be95) with a t3.micro configuration, hosting a virtual desktop environment. The instance is located in the eu-north-1b Availability Zone, with a private IPv4 address of 172.31.33.124 and a public IPv4 address of 13.61.173.20.The virtual desktop environment is designed to provide a secure, scalable, and accessible platform for startup companies to store and manage their data. With a total memory of 1024 MB and up to 5 Gigabit network performance, the infrastructure is optimized for efficient data processing and transfer.The project aims to integrate startup company details into the virtual desktop environment, enabling seamless data management and collaboration. By leveraging AWS services, we can ensure high availability, reliability, and security for startup companies' data.The cloud-based virtual desktop infrastructure offers numerous benefits, including reduced infrastructure costs, increased flexibility, and enhanced scalability. Startup companies can access their virtual desktops from anywhere, at any time, using various devices.The project's significance lies in its potential to transform the way startup companies manage their data and collaborate with team members. By adopting cloud-based virtual desktop infrastructure, startup companies can improve their productivity, efficiency, and competitiveness.

INDEX TERMS Mobile cloud computing, mobile device, BYOD, virtual desktop, android.

I. INTRODUCTION

In recent years, smart mobile devices (e.g., smart phones and tablets) have been widely used by people all over the world. They have already been an indispensable part of our daily lives. We use mobile devices to contact with families and friends, go shopping and surf the Internet. A large group of us use mobile devices to study online. Particularly, it is extremely common that we use mobile devices to handle documents and emails through mobile applications (apps) at work. In May 2017, Google announced that Android had 2 billion monthly active users [1], [2]. As of February 2017, the Google Play store has over 2.7 million Android apps published [3]. Statistics show that as of May 2016, those apps have been downloaded for more than 65 billion times [4].

Because of the great convenience, personal smart mobile devices have been increasingly used at work in many enterprises, so called Bring-Your-Own-Device (BYOD) [5], [6]. BYOD increases employees' morale by trusting them to use their own devices at work. The brought in convenience also makes the company appear more flexible and attractive[7].ManyemployersfeelthatBYODcanevenbeameans toattractnewhires,pointedbyasurveyindicatingthat44%of jobseekerslikeanorganizationmoreifitsupportsitsemployees to use their own devices [8]. According to Logicalis [9], high-growth markets (including Brazil, Russia, India, UAE, and Malaysia) demonstrate a much higher propensity of supporting BYOD. Almost 75% of the employees in these countries are allowed to do so, compared with 44% in the more mature developed markets. However, BYOD applications have several serious challenges.

The associate editor coordinating the review of this manuscript and

A. SECURITY AND PRIVACY CONCERNS

BYOD is an effective strategy for enterprises because it can effectively increase employees' productivity with lower IT be aware that BYOD also brings a number of security challenges. (*i*) It increases the exposure of a company's applications and data to malware and infections due to the lack of control and visibility on personal devices. (*ii*) Data leakage becomes a primary

concern as these personal devices now have access to sensitive corporate data. (*iii*) Providing IT support in BYOD environments is difficult due to the large diversity of personal devices regarding platforms, operating systems, and so on.

Many solutions have been proposed to mitigate the risk of supporting BYOD. Among those, device virtualization [10]–[12] and Virtual Desktop Infrastructure (VDI) [13], [14] are the most popular. Device virtualization leverages a hypervisor to allow mobile users to simultaneously run different mobile operating systems on a single device. This strategy clearly separates the access of corporate data from personal data. However, the concern still remains because corporate data is still accessible or even stored on personal devices. In addition, virtualization itself consumes a lot of computing resources.

VDI is a desktop virtualization approach which delivers a desktop OS image to end devices over the network. The desktop OS image is generated from a desktop operating system (typically, Microsoft Windows) that runs and is managed in a datacenter.VDIallowsitsuserstooperatetheOSandinteract with applications run on the OS as if they run locally. The endpoint may be a traditional Personal Computer (PC), thin client or even a mobile device. VDI seems like the best option considering mobile data protection because data will not be stored on employees' devices. Instead, the data is stored in the company's servers. This ensures that in case a device gets lost or damaged, enterprise data remains intact.

However, VDI lacks of apps since it only supports PC OS such as Windows 7 or Windows 10. Compared with apps on Android or Apple mobile operating system, the number and type of mobile apps developed for Windows is very limited. Moreover, applications running on PC OS are usually designedfor13-inchorlargerscreen,itisdifficulttoremotely use these applications on mobile devices with typical 6-inch or smaller screen.

In a word, device virtualization, VDI and VMI are all designed to address the security and privacy concerns from BYOD. However, device virtualization only partially mitigates the data leakage concern of BYOD. VDI addresses the problemwellbutitprovidesapooruserexperienceformobile users.VMI addresses the security and privacy concerns of BYOD as well as provides good experience for mobile users.

B. LIMITATIONS OF SMART MOBILE DEVICES

When using smart mobile devices, most people suffer from the following problems.

(i) Limited hardware capability. Most smart mobile devices have limited power supply, constrained storage capacity, and computation capability. Under common circumstances, mobile users need to charge their devices daily to keep power supply and clear or back up data termly to free storage space.

Most mobile users have to replace their mobile devices every year due to outdated hardware.

(ii) Frequent upgrades for apps. Most Android apps are developed by small teams (more often, a single developer). Frequent updates of apps are very common in order to strive the competition in the market. Mobile users frequently experience crash during apps usage and have to upgrade apps over and over again.

(iii) Security threat from malicious apps. With the increasing popularity of Android OS, it becomes an attractive target for malware because of its openness [22], [23]. A report from F-Secure states that the number of malicious software on the Android platform accounts for 97% of the overall number of mobile malware [24].

Mobile Cloud Computing (MCC) [26]–[28] has therefore been proposed by many researchers to overcome the above mentioned limitations of smart mobile devices by integrating cloud computing [29], [30] into the mobile environment. The strategy enables mobile users and application providers to effectively and elastically utilize resources in an on-demand fashion. However, existing research works and products on MCC are only able to employ cloud computing to optimize certain mobile applications, such as mobile cloud albums, mobile cloud storage, and mobile cloud games [31], [32]. It is stilldifficulttoprovidecloudservicesforgeneralmobileapps due to their rapid growth.

To address these issues, Virtual Mobile Infrastructure (VMI), a general framework that provides more reliable and secure solution for BYOD, has been proposed. To be specific, in VMI, mobile apps run on a mobile Operating System (OS)/virtual machine that is located on a remote server in a cloud data center. By this way, most workloads are offloaded from mobile devices to the remote server. Further, a mobile device requires negligible power to display apps on the screen, which makes mobile devices more durable. VMI takes the benefits of VDI, but it is specifically designed for mobile users. Thus, VMI provides better user experience under the BYOD environment than VDI. However, since the entire virtual desktop needs to be transferred from a remote server and displayed on a mobile device, VMI suffers from performance issues.

C. CONTRIBUTIONS

In this article, we address these issues by designing and implementing a VMI prototype with optimized network transfer mechanisms and display virtualization. In particular, our main contributions are as follows.

1) We design and implement vDesk which is a prototype system for VMI. vDesk provides an implementation of VMI desktop virtualization on windows where mobile devices can remotely access windows virtual desktops which are running in remote virtual machines. vDesk focuses on virtualizing the display of windows desktops, redirecting users' input events, and providing audio support and remote camera control and access.

2) Extensive experiments are conducted on different mobile devices under different settings to evaluate the performance of vDesk in terms of the response time, network bandwidth, application performance and overall system overhead. The experimental results show that vDesk provides system users with almost the same experience as in local when accessing and operating the remote Android virtual desktops.

The rest of this article is organized as follows. Section II describes the design and implementation of the vDesk system. In section III, we evaluate the performance of vDesk and analyze the experimental results. Section IV discusses the potential limitations of vDesk and the future work. In Section V, we summarize related works.

Finally, we conclude this article in section VI.

II. DESIGN AND IMPLEMENTATION

In this section, we first discuss the VMI framework. Subsequently, our VMI system named vDesk is introduced.

A. OVERVIEW OF VMI

VMI is a framework that provides secure and reliable solution for mobile users to operate on virtual desktops when there is a need. The key idea is to host a mobile operating system located on a local or cloud server. Virtual desktops are generated as images of the operating system and delivered to mobile devices via mobile optimized protocols through the network. The mobile users can then operate on virtual desktops on their mobile devices locally, as well as enjoy all provided services and apps. The objective of VMI is to provide smooth local operating user experiences while offload major resource consumption to servers.

As shown in Fig. 1, VMI consists of three parts: mobile clients, a VMI server and, an authentication server. The VMI server houses a customized mobile OS (mostly Android) that is hosted on virtual machines (VMs). These VMs run on a hypervisor on a centralized host machine. Each VM runs a server process to handle a client's connection, receive the client's request, and deliver the virtual mobile desktop to the client's screen. Each mobile client runs a thin client app to connect to the server running on the VM through the network. For security concern, each mobile user is allocated an account by the VMI server for identification verification. An account will be validated by the authentication server when a mobile client connects to a VM running on the server.

In VMI, mobile apps running in a VM are transferred in the form of a single thin client app to the user's mobile device. The thin client app runs independently from the remote platform OS, and can run on any mobile device and OS (iOS, and usually Android). Mobile devices receive pixel information from the remote apps, and in return send key, gesture, location, and device information. No apps or data are saved or stored on the mobile device itself. One secured communication protocol is used to transfer user input back to the remote server in order to provide maximized security.

B. OVERALL ARCHITECTURE OF vdesk

tis our prototype system of VMI and is constructed as a Client-Server system in order to provide mobile users with remote access to virtual mobile desktops and implemented on one of the most popular mobile operating systems, Android system. As shown in Fig. 2, the vDesk client is an application running on Android devices (smartphones or tablets). It is composed of a display module, a multi-touch module, an audio module, and a camera module.

The display module displays the virtual android desktop on mobile devices. The multi-touch module handles the mobile users' input events. The camera and audio modules provide mobile users with rich multimedia experience. Meanwhile, the server of Vdesk is an application running on an Android-x86 virtual machine providing corresponding services to the mobile client. The client is connected to the server through remote access. For security concern, access permission to an Android virtual desktop is given to a client only if it passes the authentication verification in the first place (1). The server responds to success verification by sending the desktop content to the client (2). Then, the client can operate on the virtual desktop and use apps such as an office suite, a file manager, a music player, and camera applications during which the generated data will be transferred between the client and the server (3).

C. DISPLAY VIRTUALIZATION

To make vDesk a viable replacement for the traditional desktop, it needs to be able to deliver similar appearance and user experience that end-users expect to. Furthermore, vDesk should work within the framework of existing display systems, intercept display content from unmodified applications, and redirect these content to remote clients. In order to provide good performance, the virtualization should intercept display content at an appropriate abstraction layer, so that sufficient information can be observed to optimize the processing of displaying content in a time efficient manner. To support transparent user mobility and eliminate client administration complexity, vDesk should support its usage with thin and stateless clients. To ensure this, all persistent display states should be stored on the server. There are many remote display protocols for mobile VDI system, such as VNC, RDP, SPICE, and etc. However, none of them can be directly used in vDesk due to that they are not compatible with Android OS.

Given that traditional display systems are structured in multiple abstraction layers, there are three layers at which vDesk can intercept the desktop display content on Android OS. The top layer is application layer which presents high-level overall characteristics of the display system. The second layer is the system service layer. It is responsible for creating a hardware-independent abstraction of the display hardware to meet the requirements of the display system and its applications. The video hardware layer is a low-level, hardware-dependent

FIGURE 1. The overview of VMI.

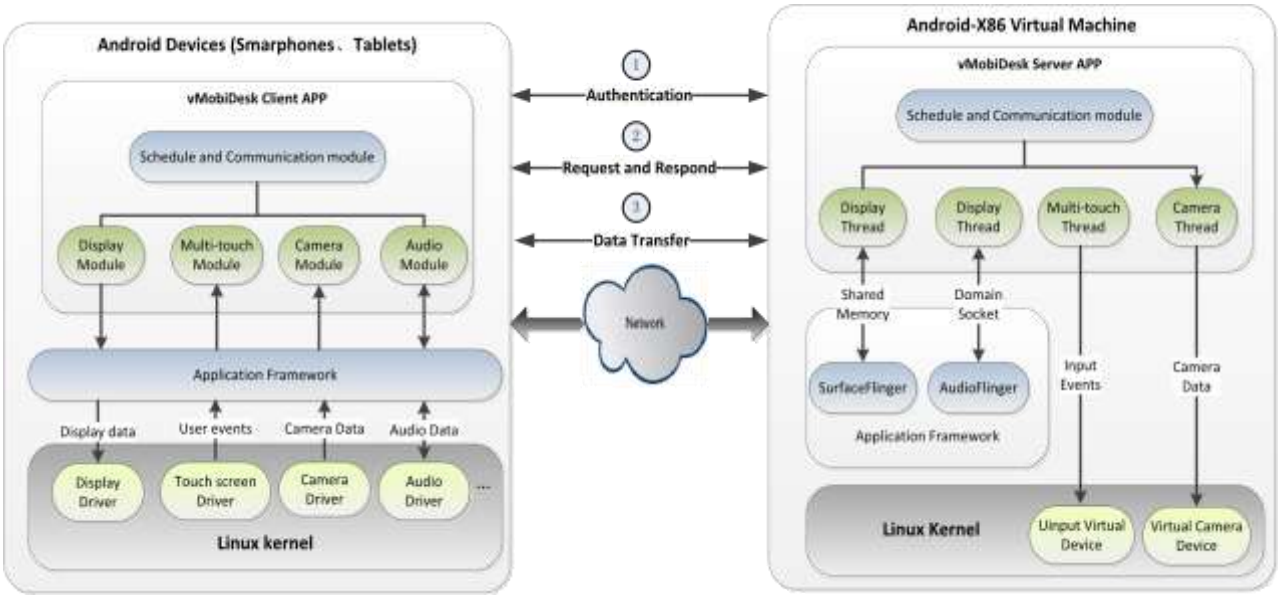


FIGURE 2. The overall architecture of vMobiDesk.

layer that exposes the video hardware to the display system.
does not intercept desktop displays at the application layer. The reason is that doing so requires a significant amount of application logic and computational resources on the client end in order to translate high level commands. Compared with VNC, vDesk does not intercept the display content at framebuffer layer because it consumes more server's CPU and causes responding delay. vDesk captures the display content of Android virtual desktop at the system service layer through SurfaceFlinger (a service for Android display system) which composes all the surfaces from applications. Subsequently, vDesk generates a complete desktop surface and writes to the hardware framebuffer for display.
ThearchitectureofdisplayvirtualizationisshowninFig.3. When a client connects to a server, the server agent requests the SurfaceFlinger to update the client's screen for every 10 milliseconds (1). The server intercepts the desktop content from the GraphicBuffer of SurfaceFlinger service, and then the intercepted content is temporarily stored in a shared memory region (2). Because it is unnecessary to display the

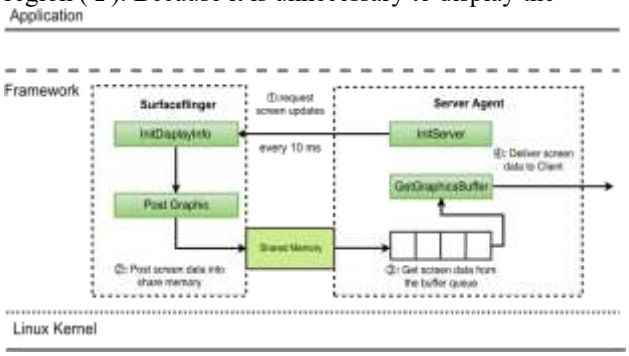


FIGURE 3. The architecture of display virtualization.

desktop on the server's screen, the desktop content will not be written to its hardware framebuffer. The server then reads it from the shared memory (3), and transmits the screen data to the client through the network (4). The display solution used in vDesk can significantly saves server's CPU consumption and improves its responding speed. It provides mobile users with better visual experience.
As mentioned above, the display content is captured in SurfaceFlingerandthentransferredtotheserveragentinstead of being directly delivered to the client through the network for display. This decision is made because direct data transfer between the server and a client through network increases security risk. SurfaceFlinger provides a more reliable and secure service display service for Android apps and has been widely used. In addition, Android OS does not well support the traditional memory sharing and Inter-Process Communication (IPC) mechanisms like Linux OS. It only provides Anonymous Shared Memory (Ashmem) for memory sharing and Binder for IPC. Therefore, SurfaceFlinger is chosen to post the display content to an Ashmem and then notifies the server agent to read the content through a Binder. After that, the server agent delivers the display content to the client.

D. INPUT REDIRECTION

With the arising of smartphones and tablets, user and device interactions no longer restricted to keyboard and mouse as majority of those devices are equipped with multi-touch screens. The responsibility of the input module is to sense the touch input events of mobile users and then seamlessly redirect those events to the server. Unfortunately, there is no remote computing protocol specifically designed for touch screen input devices.
The workflow of input redirection is shown in Fig. 4. Touch input events handled in vDesk can be loosely classifiedintosingle-toucheventsandmulti-touchevents.For instance, touching an icon with one finger to open the app on an smartphone is a single-touch event, and zooming in and out pictures with two or more fingers is a multi-touch event. On the client side, if a single-touch event is detected, user's input on virtual mobile desktop is intercepted then forwarded to the server as a single point coordinate. If a multi-touch event is identified, multiple coordinates are combined, and

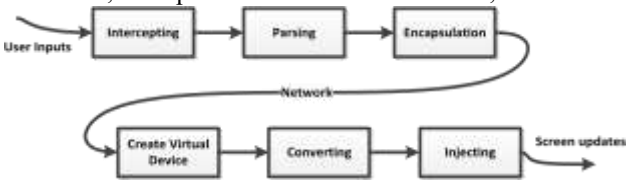


FIGURE 4. The workflow of input redirection.

then forwarded to the server. The client's user touch input events are intercepted at the device driver layer to ensure that the input events are original and application-independent, so that can be adapted to different Android systems. On the server side, all forwarded touch events are converted to local events in order to adapt to the local system and then injected into the server's input system for updates.

A virtual device has to be created on the server to receive and process touch events because there is no real touch screen device on the server side. Since Android is a Linux-based system, vDesk employs the Linux *uinput* module which is a virtual interface to create a virtual device. Therefore, all the transferred input events from the client will be transformed to uinput events so that they can be injected into the virtual device and re-executed on the server.

E. AUDIO SUPPORT

In order to support audio applications such as teleconference, vDesk is implemented with audio record and playback functionalities. On the client side, there are two modules for audio redirection. The playback module is designed to replay the audio data transferred from the server. The recording module is responsible to capture the audio data such as voice input of the client. On the server side, there are two separate channels used to process playback data and recorded data, respectively. The AudioFlinger service of Android-x86 OS is modified to process input and output audio data for audio applications.

Fig. 5 shows the workflow of audio redirection. In audio playback redirection, when a user plays a music in a virtual mobile desktop, the audio data will be intercepted by the audio thread of AudioFlinger and delivered to the server. Then the server transfers the audio data back to the client's playback module for replay. The redirection of audio recording works in a similar way. When a user opens an audiorecordingapplicationonhisvirtualmobiledesktop,this action will be immediately detected by the server's recording thread. Then a message will be sent from the server to activate the microphone on the client's mobile device. After that, audio data captured by the client's microphone is transferred to the record channel of the server through the network, and then processed by its recording application.

Similar to the display virtualization, AudioFlinger does not directly communicate with the client but the server, instead. However, data transfers between AudioFlinger and the server is not through the shared memory. This is because that audio redirection is more time-sensitive and contains less data than that in display virtualization. Transferring audio data through

FIGURE 5. The workflow of audio redirection.

shared memory may result in high latency and data loss. Therefore, we employ the UNIX-domain socket to transfer audio data between AudioFlinger and the server agent. The audio data is buffered on both the server and client sides to support real-time audio applications. To reduce the server's CPU usage, received data will not be forward to the server's audio device. Experimental results show that vDesk provides low latency audio support to users.

F. REMOTE CAMERA

Camera is one of the most important modules in smartphones or tablets. It can be used to take photos and record videos. In order to support camera applications such as video conference and scanner apps, it is necessary to provide camera redirection in vMobiDesk so that mobile users can use cameras in the virtual mobile desktop.

In vMobiDesk, to support camera redirection, there are two modules on the client. One is designed to get the information of the client's physical camera devices such as the type and number. The other module captures camera data and transfers data to the server. On the server side, since there is no physical camera device on the machine, virtual camera devices are created. The number of virtual camera devices is determined by the number of cameras on connected client's mobile device. The virtual camera devices can receive and process data received from clients' physical cameras, and return the results to the camera applications on the server.

The design of remote camera is shown in detail in Fig. 6. When a client connects to a server, its camera information will be collected by the server (1). The collected information is then sent to the MediaServer (another system service of Android OS) to create virtual camera devices (2). When a user opens a camera application in a virtual mobile desktop, this action will be immediately detected by the server's cameraservice.Thenamessagewillbesentfromtheserverto the client to activate the camera on the client's mobile device (3). After that, data captured by the client's physical camera device is transferred to the server through the network (4),

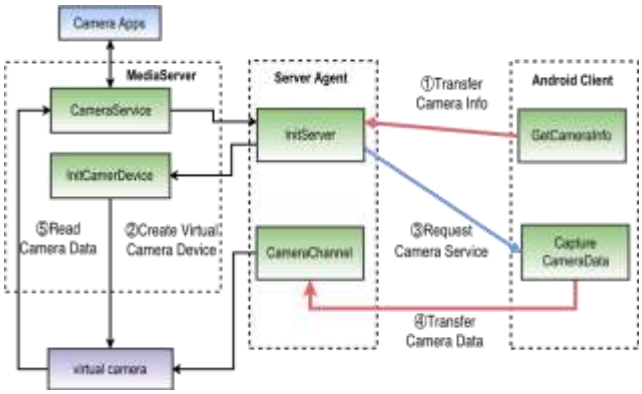


FIGURE 6. The design of remote camera.

and then processed by the camera module on the server (5). Here the key challenge is how to handle recorded videos which usually carry a large amount of video data. To reduce network bandwidth consumption, the recorded video data is encoded to H264 [33] which is one of the best format for video streaming. Originally, recorded video data is captured by client's physical camera and transmitted to the server's virtual camera devices. After being processed on the server, the video needs to be transferred back and displayed on the client's screen for previewing. Data transmits back and forth may lead to long delay and poor Quality of Service (QoS) due to the limited mobile network bandwidth in a VMI system. To address this problem, we propose not to use data transferred back from the server but to utilize the client's local buffered information for displaying or previewing. By this means, the vMobiDesk system is able to reduce delay and improve user experience for camera applications.

G. RESOLUTION ADAPTION

Due to the heterogeneity of users' mobile devices, the display resolution of those mobile devices may be significantly different. In order to provide mobile users with good viewing experience when they work on the virtual mobile desktop, resolution adaption is supported in vMobiDesk. Similarly, as soon as a client connects to a server, its resolution information will be collected by the server. According to this information, the server transforms the virtual desktop to the correspondingresolution,andthenshowittotheclient.Inthis way, mobile users have the flexibility to choose its favorable display resolution.

TABLE 1. Hardware and software specs for the experimental platform.

Server Host	3.40GHz Intel Core i7-6700 processor
	8 GB of RAM
Hypervisor	Virtualbox-5.1
	Qemu-KVM-2.4
	Xen-4.2
Guest OS	Android-x86-5.1.1 (running vMobidesk server module)
Client Device	Huawei M3 tablet
	Huawei Mate 8
	Google Nexus 9
	Vivo X7
Client OS	Android-6.0 (running vMobidesk client module)
Wireless Network	2.4G WiFi
	5G WiFi

III. PERFORMANCE EVALUATION

In this section, we present the performance evaluation of vMobiDesk through a series of experiments. The experiments are conducted on different clients and servers under different network communication settings.

A. EXPERIMENTAL SETUP

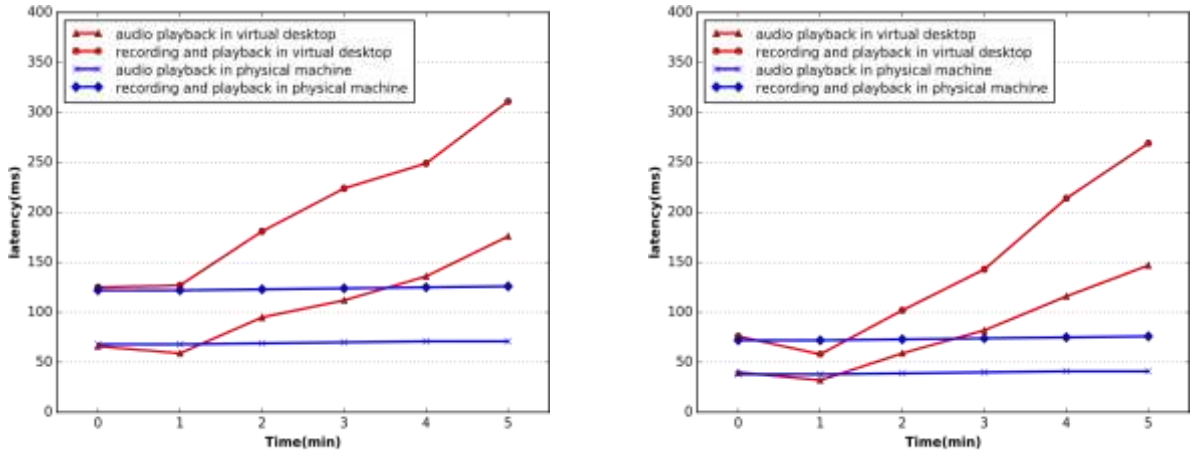
As shown in table 1, in the experiments, the server machine has a 3.40GHz Intel Core i7-6700 processor and 8 GB RAM. The server operating system is Android-x86-5.1.1 and runs in a virtual machine created by Virtualbox [34], KVM [35], and Xen [36] hypervisors, respectively. The tested client devices include Huawei M3 tablet, Huawei Mate 8, Google Nexus 9 and Vivo X7 running on Android-6.0 operating system. A 100 Mbps, 1 ms latency LAN network is utilized to obtain local 2.4G and 5G Wifi communication between the mobile devices and the server. vMobidesk consists of a client module and a server module, the client module is running on the tested mobile devices as an application, the server module of vMobiDesk is running on the Android-x86 virtual machine as a system service. The performance of vMobiDesk is mainly evaluated in terms of response time, network bandwidth consumption, application performance, and the overall system overhead.

We have tested the performance of vMobidesk for different hypervisors and mobile devices. Because the experimental results show that there is no significant difference among different hypervisors and mobile devices, we therefore give the complete results for the response time. For the bandwidth consumption and audio performance, only the results of Xen hypervisor and Huawei smartphone are provided. For the remote camera, we give the results for Xen hypervisor and all the mobile devices.

B. RESPONSE TIME

In vMobiDesk, the server provides mobile users with remote access to the virtual mobile desktop through local Wifi. When a user works on the virtual desktop, the system needs a certain response time to redirect each operation to the server and return back the results to the client. Response time is one of the most important metrics to evaluate users' experience. The response time is expected to be small so that the users can havesimilarexperiencewhenoperatingonthevirtualdesktop as that on their local mobile devices. The response time is tested when the server is configured in different resolutions in vMobiDesk. The average response time is collected from operations such as opening a folder with a file manager, typing several words in a Word document and opening a Web page, and etc..

As shown in Fig. 7 and Fig. 8, the response time is always less than one second which is acceptable for human perceiving. But it increases with resolutions, this is because that a virtual desktop with high resolution contains much more data than that with low



(a) Latency of audio applications under 2.4G WiFi (b) Latency of audio applications under 5G WiFi

FIGURE 10. The latency of audio applications in vMobiDesk under different network environments

resolution, it takes more time to transfer data and to display the virtual desktop on the client end under high resolution.The response time stays almost the same for different virtual machines and different mobile devices, because vMobiDesk is implemented in an Android guest OS instead of hypervisor layer. Different virtualization platforms show little influence on response time of Android virtual desktops. As to different mobile devices, the client of vMobiDesk is running on a mobile device just as a common Android app, different mobile devices also have little impact on the response time of vMobiDesk. Because a virtual desktop with high resolution contains much more data than that with low resolution, it takes more time to transfer data and to display the virtual desktop on the client end under high resolution. In addition, the response time performs differently under 2.4G WiFi and 5G WiFi, especially under high resolution. Because 5G WiFi is more stable and faster than 2.4G. vMobiDesk performs better in 5G WiFi network environments.

C. BANDWIDTH CONSUMPTION

This experiment is designed to measure the detailed bandwidth consumption when browsing web pages in virtual desktops with different resolutions. In practice, we test the bandwidth consumption of web pages for different virtual machines and mobile devices. Similar to the response time, there is no significant difference. Therefore, here only shows the experimental results for Xen virtual machine and Huawei smartphone. As shown in Fig. 9, higher resolution results in more network bandwidth . Web pages contains image consuming more bandwidth than web pages with text. Since 5G WiFi is more stable and faster, it consumes more bandwidththan 2.4G WiFi. When the resolution is below 1024*768, in 2.4G WiFi, due to the severe signal-jamming, packets are discarded during transmission. When the resolution is over 1024*768, even 5G WiFi may suffer from data loss (This is not shown in the Figure).

D. AUDIO SUPPORT

To evaluate the performance of audio applications in vMoBiDesk, we conduct experiments to measure the latency of audio playback and audio recording under different network environments. For audio playback, we use a music player to play a 5 minutes long music. The latency is the time cost from starting playing music on the server to hearing the music on the client. For audio recording, we develop a recorder app which can record and replay the recording simultaneously. The latency is the time from the user speaks on the client to hears himself from the client's audio device.

The latency is collected in every minute. The test repeats tentimesandtheaveragesarecalculated.AsshowninFig.10, audio applications show lower latency in 5G WiFi than that in 2.4G WiFi which indicates that the performance of audio redirection in vMobiDesk is influenced by network bandwidth. The experimental results also demonstrate that the latency in virtual desktop accumulates as time goes by. But the latency is always below 400 ms. It is acceptable if most of the time users only use audio service in a short period. The latency increase may be caused by the deviation of system clock in a virtual execution environment. To verify our inference, we deploy the server of vMobiDesk in a physical machine that runs Android-x86-5.1.1. The experimental results in Fig. 10 demonstrate that the latency is not accumulated as time goes by while running vMobiDesk in a physical machine.

E. REMOTE CAMERA

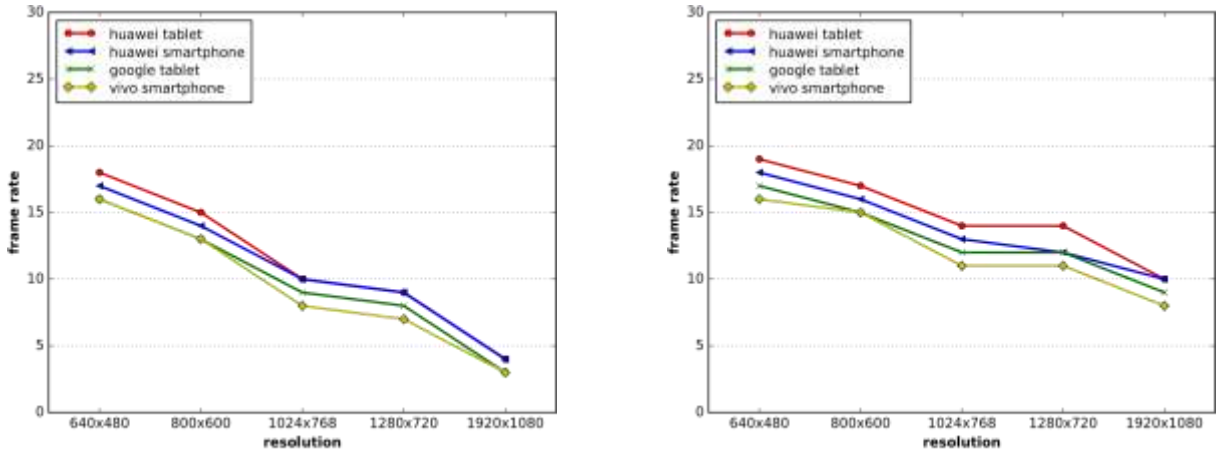
Remote camera is one of the most important modules in vMobiDesk. The performance is measured based on the effective frame per second (FPS) and bandwidth consumption with different resolutions under 2.4G and 5G WiFi network environments on different mobile devices.

As shown in Fig. 11, the results is highly related to the quality of hardware camera employed on different mobile devices. Huawei tablet performs the best. Its effective frame rate can reach up to 20 FPS. For all mobile devices, FPS decreases as the resolution increases. That is because the increase of resolution results in the increase of image size. The network can not transmit all data produced by the camera in time. Specially, when the resolution is 1920 * 1080, we have optimized vMobiDesk to transfer the video data only once.Theeffectiveframeratecanreach12FPS,anduserscan still watch video smoothly.

As shown in Fig. 12, the camera occupies lots of bandwidth since it transmits the generated data from the client to the server through the network. Fortunately, though the recording

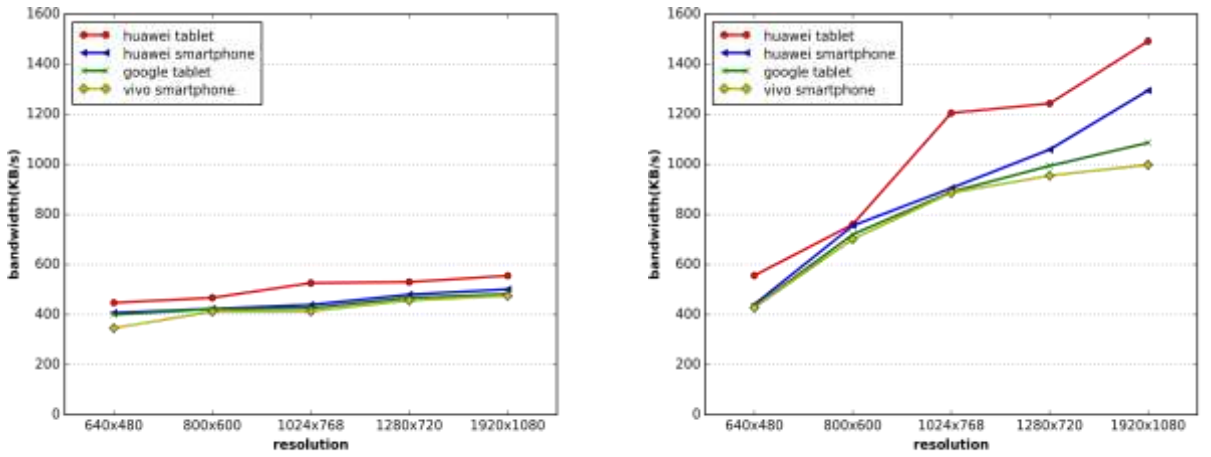
TABLE 2. The overhead of the vMobiDesk client.

may



(a) FPS of remote camera under 2.4G WiFi (b) FPS of remote camera under 5G WiFi

FIGURE 11. FPS of remote camera under different network environments.



(a) Bandwidth consumption of remote camera under 2.4G WiFi (b) Bandwidth consumption of remote camera under 5G WiFi

FIGURE 12. Bandwidth consumption of remote camera under different network environments.

Application \ Overhead	CPU usage (%)	Memory usage (MB)
Idle	5%	20
File Manager	8%	22
Office	12%	45
Browser	18%	44
Music Player	11%	60
Recorder	13%	49
Camera	15%	177

TABLE 3. The overhead of the vMobiDesk server.

Application \ Overhead	CPU usage (%)	Memory usage (MB)
Idle	4%	13
File Manager	10%	24
Office	14%	58
Browser	15%	66
Music Player	13%	60
Recorder	11%	52
Camera	19%	194

produce large amounts of video data, the optimization which reduces two times transfer to one greatly reduces the bandwidth consumption.

F. SYSTEM OVERHEAD

In this section, we present the CPU and memory usage profiling of vMobiDesk. The statistic information is collected during the execution of the client application (shown in table 2) and the server (shown in table 3).

On the client side (table 2), the idle state consumes about 5% CPU and 20 MB memory. While a mobile user launches different applications in the virtual mobile desktop, the CPU usage rises to about 8%, 12%, 11%, 13% and 15%, accordingly. The memory usage also rises to 22 MB, 45 MB, 44 MB, 60 MB, 49 MB, and 177 MB.

Table 3 summarizes the CPU and memory consumption of vMobiDesk on the server side. It shows only 4% CPU and very little memory are consumed when the server is idle. Even when running different applications, server's CPU usage is always under 20% and the memory usage is less than 200 MB.

To summary, vMobiDesk brings slight overhead to the entire system. This is because that it is implemented by modifying the Android-x86's system services such as SurfaceFlinger service, MediaServer, and even the Linux kernel to support desktop virtualization. The core modules of vMobiDesk are integrated

into the Android operating system. Thus, the main overhead is caused by data transfer from Android OS to the server agent.

IV. DISCUSSION

In this article, we have implemented vMobiDesk, a prototype system for VMI. The proposed prototype, while not perfect, shows its advantage when being deployed to mobile cloud data centers. In a series of experiments, the results show that vMobiDesk performs well for common applications in popularvirtualizationenvironmentssuchasVirtualbox,KVM,and Xen. vMobiDesk is adaptive for a variety of mobile devices such as Huawei, Vivo, and Nexus. It can be easily integrated into existing cloud products so that the cloud providers are able to provide mobile users with Android-based VMI services. The benefits of VMI and future investigation direction are summarized in this section.

A. BENEFITS OF VMI

1) VMI provides a general framework for running mobile apps in remote servers. Most workloads are offloaded from mobile devices to cloud data centers. That makes mobile devices more durable, and mobile users can use more abundant apps than before. In addition, since the mobile OS is running on virtual machines located on cloud data centers, mobile users can create, suspend, hung up, delete, and stop the virtual mobile desktop through the hypervisor anytime and anywhere.

2) VMI can reduce workload of application developers. Before, there are a variety of smart mobile devices such as iPhone, Samsung, Huawei, and etc., and different mobile OS such as iOS, Android and Winphone, and etc.. Ifwe buildamobile application,we haveto buildit for each platform. Using VMI means that we can write one single version app and then deploy it to any device.

3) VMI can mitigate the risk and challenges of supporting BYOD policies in enterprises. Much like VDI, corporate data never gets stored on employee devices. Each employee gets assigned a profile that is centrally managed and stored on secure company servers. Employees only need to install a secure app on their devices, log in, and access all their company files and data without worrying about mixing personal and company data or IT controls their gadgets. VMI also saves time, effort, and resources. Through a central management system, IT administrators can modify profiles, check for security, and pushes updates to employees' VMIs through a single console.

B. LIMITATIONS AND FUTURE WORK

First, the response time can be further reduced. Experimental results show that the response time is within the range from 180 ms to 500 ms under 5G WiFi. It seems to be tolerable for common applications such as office and file managers, but it isunacceptableforvideosandgames.Therefore,theresponse time can be further reduced so that vMobiDesk can be widely adopted. One possible way is to improve network environments in our experiments since WiFi network tends to be interfered by other WiFi networks, which may increase the response time. On the other hand, better compression algorithms can be leveraged to process the screen updates so that the response time can be reduced.

Second, the vMobiDesk can be further extended to support requirement for watching High Definition (HD) videos and playing mobile games. At present, vMobiDesk is not strong in supporting HD videos and games. Because they both contains a lot of data communication and frequently updates. Therefore, it is very important to further explore solutions to better process videos and games in VMI system. InspiredbyexistingVDIsystemforWindows,webelievethat redirecting drawing commands for playing videos and games from the server to the client should be an effective method. ThismethodallowsGPU-intensiveworkloadsoffloadedfrom the server to the client. Though this method may require more processing workload on the client side, it can provides mobile users with good user experience when playing HD videos and games.

Third, as shown in section III, vMobiDesk is designed and implemented by modifying the source code of Android operating system services such as SurfaceFlinger and MediaServer. Though this method makes vMobiDesk more adaptive to the current Android system, frequent upgrades of Android OS may incur compatibility and maintenance problems. In order to improve the compatibility and maintainability, we plan to implement vMobiDesk system in a non-invasive way. According to quantitative analysis on the architecture of Android operating system, we propose to implement vMobiDesk at Android's Hardware Abstraction Layer (HAL) which is a general framework among different Android versions and composed of dynamic-link libraries that can be easily loaded and off-loaded.

Last, existing popular virtualization technologies (Virtualbox, KVM, and Xen) still show some weakness such as heaviness and poor performance. Most of them are unable to provide good support for GPU virtualization [37]–[40], especially for Android system. But GPU is very important for playing HD videos and games. we plan to utilize Linux container [41], a lightweight virtualization technology to virtualize the operating system on the server. Running Android system in containers will provide mobile users using virtual mobiledesktopwithabetteruserexperiencethanrunningthat in Virtualbox, KVM, or Xen virtual machines.

V. RELATED WORK

VMI provides a general framework for running mobile apps in a cloud data center. Meanwhile, VMI is a better solution for BYOD policy than Virtual Desktop Infrastructure (VDI). In this section, we summarize existing works related to our research.

A. BRING YOUR OWN DEVICE

The concept of BYOD (Bring Your Own Device) refers to the policy of permitting employees to bring personally owned mobile devices to their workplace, and access privileged company information and applications with these devices. In recent years, BYOD has been explored by more and more researchers. Keith W considers the security and privacy in BYOD [42]. In their opinion, mobile devices contain a wealth of data that a user might deem private. If personal data is co-mingled with the employer data on the same device, it is difficult to build barriers between personal and employer data. Bill Morrow also indicates that BYOD may result in security implications for data leakage, data theft and regulatory compliance [43]. To protect valuable information, organizations should stop making a distinction between devices in the corporate network anddevices outside of it. [10], [11]proposed mobile virtualization, which refers to the technology that

enables a single device to offer two or more people with differentsystemsettingsanduserprofilesandtotallydifferent operating environments.

B. VIRTUAL DESKTOP INFRASTRUCTURE

Many productive desktop virtualization systems have been developed and applied to various commercial applications due to its strength in reducing maintenance and operating costs, improving resource utilization efficiency and so on. VNC [44]and THINC [12] are famous thin-client systems proposed in academic research field. While in industry, there are Microsoft Remote Desktop [16], Citrix XenDesktop [17], VMwareView[14],SunRayandHPRemoteGraphicsandso on. VNC (Virtual Network Computing) is a popular remote display system based RFB protocol. It uses a virtual driver to maintain local copy of the framebuffer state. The state information is used to refresh its display, and forwards user input directly to the server. VNC shows good performance for office applications but does not support multi-touch, audio or camera. Therefore it is not suitable for VMI system.

THINC and its portable version pTHINC intercept low level video driver commands and adopts a push mode to interact with client. It is efficient for UI compression but suffers from compression performance degradation over multimedia content encoding. As a result, it achieves good performance in multimedia playback with sufficient bandwidth but not for network environments with low bandwidth. RDP (Remote Desktop Protocol) is widely used in desktop virtualization products such as Microsoft RDS and VMvare view. For office applications, such as a text editor or a spread-sheet, RDP is highly optimized. The display changes are quite small and have low frequency that can be efficiently processed. However, it is proprietary and only supports Windows operating system. MobiDesk [45] proposes a thin client solution for mobile devices by optimizing the WAN traffic involved in performing remote computing. The solution is primarily meant for mobile laptops and is similar to other remote computing approaches in principle. However, all these solutions assume the server is a PC (Windows 7, Ubuntu, Mac OS, etc.) desktop. They can not be easily moved to a mobile device such as smartphone or tablet.

VI. CONCLUSION

With the rapid adoption of smartphones and tablets, MCC has recently attracted significant attention from both academia and industry. In addition, BYOD has been adopted by more and more enterprises because of the efficiency and convenience. However, existing solutions for MCC and BYOD still expose several drawbacks such as lack of generality and security. To address the existing problems in MCC and BYOD, we propose vMobiDesk, a prototype system for virtual mobile infrastructure (VMI). We implemented vMobiDesk on Android operating system which is one of the most popular mobile operating systems. We conducted extensive experiments to evaluate vMobiDesk in terms of response time, network bandwidth consumption, application performance and the overall system overhead on several popular mobile devices under different virtualization and network environments. The experimental results show that vMobiDesk provides mobile users with good user experience on remotely accessing Android virtual desktops for several applications (e.g., office suite, file manager, web browser, music player, and camera applications).

REFERENCES

1. Protalinski, "Android passes 2 billion monthly active devices," VentureBeat, San Francisco, CA, USA, Tech. Rep., 2017.
   A. Ng, "Google's Android now powers more than 2 billion devices," CNET. CBS Interact., Tech. Rep., May 2017.
2. *Number of Android applications*, AppBrain, Mar. 2017.
3. N. Statt, "Android users have installed more than 65 billion apps from Google Play in the last year," Verge. Vox Media, Washington, DC, USA, Tech. Rep., 2017.
4. R. Ballagas, M. Rohs, J. G. Sheridan, and J. Borchers, "Byod: Bring your own device," in *Proc. Workshop Ubiquitous Display Environ., Ubicomp* 2004.
5. Y. Song, "'Bring your own device (BYOD)' for seamless science inquiry in a primary school," *Comput. Edu.*, vol. 74, pp. 50–60, May 2014.
6. [Online].Available:http://www.retailwire.com/discussion/16188/happiness-is-bringing-your-own-computer-devices-to-work
7. K. Casey, "Risks your BYOD policy must address," InformationWeek, San Francisco, CA, USA, Tech. Rep., Jun. 2013
8. [Online]. Available: http://cxounplugged.com
9. J. Andrus, C. Dall, A. V. Hof, O. Laadan, and J. Nieh, "Cells: A virtual mobile smartphone architecture," in *Proc. 23rd ACM Symp. Operating Syst. Princ. (SOSP)*, 2011, pp. 173–187.
10. W. Chen, L. Xu, G. Li, and Y. Xiang, "A lightweight virtualization solution for Android devices," *IEEE Trans. Comput.*, vol. 64, no. 10, pp. 2741–2751, Oct. 2015.
11. J. Shuja, A. Gani, K. Bilal, A. U. R. Khan, S. A. Madani, S. U. Khan, and A. Y. Zomaya, "A survey of mobile device virtualization: Taxonomy and state of the art," *ACM Comput. Surv.*, vol. 49, no. 1, pp. 1–36, Jul. 2016.
12. R. A. Baratto, L. N. Kim, and J. Nieh, "THINC: A virtual display architecture for thin-client computing," *ACM SIGOPS Operating Syst. Rev.*, vol. 39, no. 5, pp. 277–290, Oct. 2005.
13. [Online]. Available: http://www.vmware.com/products/horizon-view
14. [Online]. Available: http://en.wikipedia.org/wiki/TightVNC
15. [Online]. Available: http://msdn.microsoft.com/en-us/library
16. [Online]. Available: http://hdx.citrix.com/hdx
17. [Online]. Available: http://technet.microsoft.com/
18. M. Becher, F. C. Freiling, J. Hoffmann, T. Holz, S. Uellenbeck, and C. Wolf, "Mobile security catching up? Revealing the nuts and bolts of the security of mobile devices," in *Proc. IEEE Symp. Secur. Privacy*, May 2011, pp. 96–111.
19. M. L. Polla, F. Martinelli, and D. Sgandurra, "A survey on security for mobile devices," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 446–471, 1st Quart., 2013.
20. Q. Li and G. Clark, "Mobile security: A look ahead," *IEEE Secur. Privacy*, vol. 11, no. 1, pp. 78–81, Jan. 2013.
21. J. Huang, X. Zhang, L. Tan, P. Wang, and B. Liang, "AsDroid: Detecting stealthy behaviors in Android applications by user interface and program behavior contradiction," in *Proc. 36th Int. Conf. Softw. Eng. (ICSE)*, 2014, pp. 1036–1046.
22. Ferreira, V. Kostakos, A. R. Beresford, J. Lindqvist, and A. K. Dey, "Securacy: An empirical investigation of Android applications' network usage, privacy and security," in *Proc. 8th ACM Conf. Secur. Privacy Wireless Mobile Netw. (WiSec)*, 2015, pp. 1–11.
23. [Online]. Available: http://www.f-secure.com/
24. Huang, "Mobile cloud computing," *IEEE COMSOC Multimedia Commun. Tech. Committee (MMTC) E-Lett.*, vol. 6, no. 10, pp. 27–31, 2011.
25. X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.
26. • • •