SecureLiveApp : A Secure Sharing and Migration Approach for Live Virtual Desktop Applications in a Private Cloud

Dr. K. Karuppasamy[1]
Professor

K. S. Prabhu[2]
Assistant Professor

K. Arunkumar[3]
Assistant Professor

Department of IT, RVS College of Engineering and Technology
Coimbatore, Tamilnadu, India.

Abstract

Computing offers a flexible and relatively cheap solution to deploy IT infrastructure in an elastic way. An emerging cloud service allows customers to order virtual machines to be delivered virtually in the cloud; and in most cases, besides the virtual hardware and system software, it is necessary to deploy application software in a similar way to provide a fully-functional work environment. Most existing systems use virtual appliances to provide this function, which couples application software with virtual machine (VM) image(s) closely. In this paper, we propose a flexible collaboration approach to enable live virtual desktop application sharing, based on a cloud and virtualization infrastructure. This system supports secure application sharing and on-demand migration among multiple users or equipment. To support VM desktop sharing among multiple users, we develop a secure access mechanism to distinguish their view privileges, in which window operation events are tracked to compute hidden areas of windows in real-time. A proxy-based window filtering mechanism is also proposed to deliver desktops to different users.To achieve the goals of live application sharing and migration between VMs, a presentation redirection approach based on VNC protocol and cloud based VM migration is used. Results of evaluations have verified that these approaches are effective and useful.

Index Terms - Cloud Computing, User-level virtualization, Virtual machine, Deployment.

1. INTRODUCTION

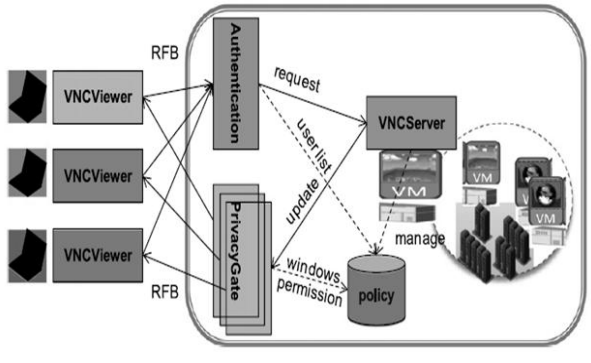
Nowadays, the Internet has become a data and computing center, with a large number of applications, and ubiquitous equipment. New Internet-based computing paradigms, e.g., cloud computing [1,2], have emerged, aiming to bring large-scale computing, storage, and data service resources together to build a virtual computing environment. These paradigms provide simple and transparent approaches that enable effective sharing and utilization applications over the Internet.

In the early personal computing era, to use software, users needed to install it under a granted license. This traditional method suffers several limitations as the software has increased both in amount and number of categories. First, software users need to deal with many complex tasks in terms of software installation, configuration, updating, and even troubleshooting. Besides, software which is dependent on their respective host operating systems may face compatibility issues. Normal users are thus loaded with an extra burden. In contrast, the concept of software as service (SaaS) [3] that emerged with cloud computing has been promoted by many companies, such as Amazon and Google. With SaaS, things become simpler. Software can be installed into VMs with easier encapsulation and secure Isolation. Users could access software on demand through the Internet without worrying about maintenance issues. There are two approaches to achieve these goals. One is to redevelop software (e.g., GoogleDoc) based on Web technologies. This not only requires much extra work, but also leads to compatibility problems on various browsers. The other approach is based on desktop virtualization, which separates the presentation and execution of applications, and provides a transparent way to deliver an application-based remote virtual desktop. Currently, a virtual desktop can be delivered based on remote display protocols, such as VNC (Virtual Network Computing) [4,5], and RDP(Remote Desktop Protocol) [6]. These protocols generally provide methods for

remote virtual desktop accessing, so that users can log into a VM and operate on the desktop.

Over the Internet, secure collaboration among users and equipment is an important requirement [7], and, in particular, user terminal equipment shows increasing mobility. In a cloud computing environment, applications are executed in VMs. Therefore to provide a novel flexible collaboration service among live virtual desktop applications (a live application in short) is the focus of our work. There are some new scenarios for live application sharing in a single VM or among multiple VMs. Ina single VM, the collaboration scenarios can be supported based on shared desktops. For example, in a remote teaching system, desktop sharing enables the instructor and students to operate on the same view. However, a traditional way for remote desktop sharing is to simply share the desktop login account and password, which may induce inconvenience and insecurity. In a cloud, a virtual machine monitor (e.g., KVM) only provides a coarse grained access control in VM-level granularity (e.g., Qemu VNCimplementation). This brings a disadvantage that authorization only has two possible results: success or failure, and it means that users may either all have complete operation rights with poor security and privacy protection, or cannot concurrently access the desktop at all. In particular, it is impractical to share the whole desktop including users' private windows. Thus, a fine-grained access control mechanism at window-level granularity is required to provide security and privacy protection

2. SYSTEM DESIGN



2.1. Secure sharing of a single VM desktop among multiple users

Fig. 2.1 shows the architecture of secure desktop sharing for multiple users. For every VM owned by a user, the display of running applications is delivered to the VNCViewer on the client sides through a VNCServer installed in the VM. The basic procedure of secure desktop sharing among multiple users is as follows.

1. Extraction of application windows in the operating system. To enable administrators to assign application permissions to users at window granularity, SecureLiveApp needs to extract all related windows from the operating system. First, SecureLiveApp gets all of the window handlers on a desktop. Second, SecureLiveApp uses the window handlers to obtain related process handlers which the windows belong to, by which we can get the application process names. Finally, SecureLiveApp creates an available application list to users for making security policies.
2. Policy configuration on SecureLiveApp@ Server side. The owner of the virtual desktop creates a list of those who are allowed to access his/her desktop and assigns permissions. The policy is defector an access control list. The policies include two types —one is a permit policy which defines the windows that a user can access, and the other is a forbid policy which defines the windows that a user cannot access. When a VNCServer runs in the OS, you can right-click the MetaVNC icon to pop-up a menu. When one chooses Properties menu, some configuration options are shown on the tabs, and we add two extra tabs on the MetaVNC options dialog to support such configuration.
3. Authentication of requested users. When a user wants to access the remote virtual desktop, He/She will first launch the VNCViewer to send a connection request to the VNCServer though the RFB protocol. The authentication modules will authenticate the identity of the requesting user based on the local policy database. If the user is authenticated and acquires the permissions, the VNCServer will build a connection with theVNCViewer.
4. The PrivacyGate module functions. In current MetaVNC functions, a VNCClient object is related when the VNCServerconnection is created. Once some events occur on the server desktop, the VNCServer will deliver the updated desktop to every VNCClient. Therefore, each client will get the exact same desktop view. In order to allow each user to have a customized view, the VNCServer works on the proxy mode to hide some specified windows according to the policies. The significance of our design is a breakthrough in the current MetaVNC structure. A PrivacyGate is used to send the desktop to each VNCClient, which means that, when the server desktop is changed, it will update the desktop view though a PrivacyGate module based on the permit policy or deny policy. The PrivacyGate module will hide the window area for which the corresponding user has no browsing permission, and

then deliver the filtered desktop view to every VNCViewer.

5. Hiding specific application windows of PrivacyGate. In order to allow every user to have an independent view, the VNCServer needs to hide the specified windows according to the policy. First, SecureLiveApp gets the names of the invisible windows which the administrator has configured for the user. Through this name, SecureLiveApp calls a query function to obtain the handlers of the application window and then gets rectangle area of this window. Second, this rectangle area is added into the hidden area, which could be a simple rectangle or a complex polygon. When setting the hidden area, some overlapped areas with the hidden windows belonging to the permitted top-level windows should be shown.
6. Blocking input from keyboard and mouse at PrivacyGate. Since some users are permitted to only browse, not operate, a desktop, SecureLiveApp blocks input of keyboard and mouse from these users. If the input variable state is ‘‘DISABLE”, the server will block the client’s inputs, and the user can only view the desktop windows, but cannot operate the server’s windows. If the input variable is ‘‘ENABLE”, the user’s inputs from the remote client will be handled and responded normally. We control the input operations through the Windows hook function, and it intercepts and processes the requests to decide whether to forward the input events.

2.2. Application sharing and migration among multiple VMs

To realize remote application access in a cloud, we use the VNC protocol to transfer the virtual desktop of a remote VM. The VNC protocol works at the buffer frame layer and supports the remote access to graphical user interfaces, and the mouse or keyboard inputs can be transferred to the remote application, thus achieving a transparent access to the applications. In such a presentation streaming-based software delivery mode, when a client wants to migrate or share an application to another client, the presentation streaming of this application should be redirected and the corresponding VM will be cloned in the case of application sharing. Based on these considerations, we have designed the architecture for live application sharing and migration.

The key components are as follows.

- Client Controller: With a modified VNCViewer to display application windows from multiple VM machines, the Client Controller manages multiple VNC connections between the VNCViewer and VNCServers. Using an inter-process communication technology, the Client Controller can close or create a specified VNC connection.
- Server Controller: This maintains information of all live users and applications in the VM pool. This component provides a unified management and processing of all clients’ requests, which includes application presentation streaming, and application sharing and migration service.
- VM Manager: This provides functionalities of monitoring, stopping, cloning, or restarting a specified VM with running applications. The VM Manager receives notifications from the Server Controller to clone a VM or manage the VM pool.

2.3. Virtual desktop application sharing and migration protocols

When a Client Controller requests to migrate or share a live application, two corresponding protocols are designed to achieve these goals. Several steps are taken to complete the migration or sharing, as shown in Fig 2.2 and Fig 2.3.

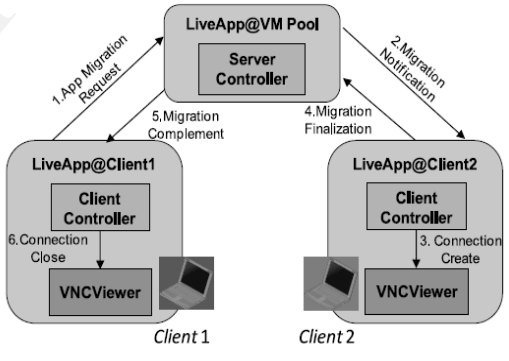


Fig 2.2 Application Migration Protocol

The steps of application migration are as follows.

Step 1: Client1 → Sever Controller: {AppName, MigrationSource,MigrationDes}. Client1 sends a migration request to the ServerController with the AppName, MigrationSource, and MigrationDes. In SecureLiveApp, all connected clients first register in the ServerController, so Client1 can choose the destination from the client list on the LiveApp@Client side.

Step 2: Sever Controller → Client2: {AppName, VMInfo, MigrationSource, and MigrationDes}. Client2 gets a migration notification from the LiveApp@VM Pool side, and the VM IP address and VNCServerport are enclosed in VMInfo, so the VNCViewer can connect to a remote VNCServer to view the

application. Sometimes, Client2 does not have a public IP address (e.g., it is located in a local network), so the Server Controller cannot initialize a connection to Client2. In SecureLiveApp, we can change the mode ofStep 2 through actively querying the Server Controller at a certain interval.

Step3:ClientController@Client2→VNCViewer@Client 2: {VMInfo}.Client2's Client Controller creates a new Client2's VNCViewer connection to the remote VNCServer running on the VM, and the VNCServer will also close the connection with Client1. Therefore, the presentation streaming of Client1 can be redirected to Client2, and the application view and data are exactly the same due to there being no change in its running environment.

Step4:Client2→ServerController: {MigrationFinal}. Client2sends a MigrationFinal message to the Server Controller.

Step5:ServerController→Client1 :{MigrationComplement}.The Server Controller updates the meta information of VMPool, and sends a Migration Complement message to Client1.

Step6:ClientController@Client1→VNCViewer@Client 1 :{VMInfo}. Client1's Client Controller clears the connection record of Client1's VNCViewer, and shows a migration success dialog on Client

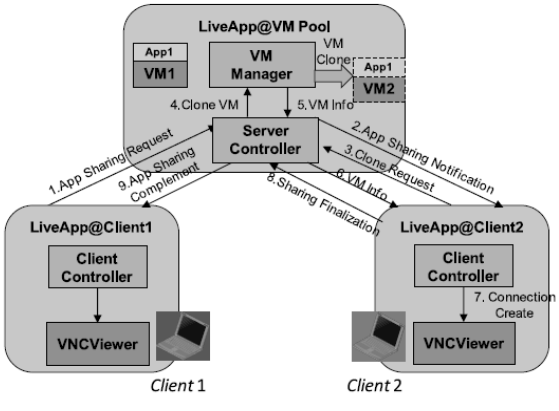


Fig 2.3 Application sharing protocol among multiple VMs.

1.The steps of application sharing are as follows.

Step 1: Client1 → Server Controller: {AppName, SharingSource,SharingDes}. Client1 sends an application sharing request to ServerController with the AppName, SharingSource, and SharingDes. In SecureLiveApp, all connected clients first register in the ServerController, so Client1 can choose the destination from the client list on the LiveApp@Client side.

Step 2: Server Controller → Client2: {AppName, SharingSource, SharingDes}. Client2 gets a sharing notification from the LiveApp@VM Pool side. Sometimes, Client2 does not have a public IP address (e.g., it is located in a local network), so the Server Controller cannot initialize a connection to Client2. In SecureLiveApp, we can change the mode of Step 2 through actively querying the Server Controller at an interval.

Step 3: Client2 → Server Controller: {CloneRequest}. Client2sends a clone request to the Server Controller after it accepts the sharing message from Client1.

Step 4 & Step 5: Server Controller → VMManager: {VMClone,VMInfo}. The Server Controller sends a VM clone request to the VMManager, and the VMManager clones a VM with the Libvirt command.

Step 6: Server Controller → Client2: {VMInfo}. The cloned VMIP address and VNCServer port are enclosed in VMInfo, so the VNCViewer can connect to a remote VNCServer to view the application.

Step 7: Client Controller@Client2 → VNCViewer@Client2: {VMInfo}.Client2's Client Controller creates a new Client2's VNCViewerconnection to the remote VNCServer running on the cloned VM. Therefore, Client2 can view the duplicated application like Client1,and the initial application view and data are exactly the same.

Step 8: Client2 → Server Controller: {SharingFinal}. Client2sends a SharingFinal message to the Server Controller.

Step 9: Server Controller → Client1: {SharingComplement}. TheServer Controller updates some Meta information of VMPool, and sends a Sharing Complement message to Client1, and lient1 will unlock its interaction.

In terms of application sharing, we intend to provide a completely duplicated application for the two clients. In SecureLiveApp, this requirement is met by cloning the VMin which the application runs. We will discuss the design and implementation of VMcloning in a later section. When a Client Controller sends a VM cloning request to the Server Controller, the latter will send the VM id to the VM Manager, who will return the new VM IP address after the cloning completes. In this new cloned VM, all the disk and memory data are the same as the original one, thus providing a duplicated application view.

3. RELATED WORK

The representative projects on desktop virtualization include THINC [9,8,10], Citrix XenDesktop [6,11], Microsoft Terminal Service [6] and some VNC systems [5,12]. THINC is a remote display system architecture for high-performance thin-client computing in both LAN and WAN environments. THINC enables higherlevelgraphics primitives used by applications to be transparently mapped to a few simple low-level primitives that can be implemented easily and efficiently. Citrix provides full VDC(Virtual Desktop Computing) using their ICA protocol in parallel with the Ardencies image and provisioning manager and desktop server hypervisor. Recently, XenClient has extended the benefits of desktop virtualization to mobile users, offering improved control for IT with increased flexibility for users. RDP enhancements in Windows Server 2008 and in recent MS client Operating Systems will also address some of the problems identified in relation to video and other graphics-intensive applications over RDP. VNC[4,13–15] is based on the PRB protocol, which is a simple and powerful remote display protocol. Unlike other remote display protocols such as the X Window System and Citrix's ICA, the VNCprotocol is totally independent of operating system, windowing system, and applications. RealVNC [12] proposes different remote display solutions for client access: the software is executed at remote servers; the user's client just gets the presentation desktop. This solution only focuses on the separation of execution and presentation, and does not involve software deployment and execution-related fields. MetaVNC [5] pursues a remote desktop environment on which users can control applications on different hosts seamlessly. MetaVNC is a window-aware VNC, and it merges windows from multiple remote desktops nto a single desktop screen. In addition, some products and research work have emerged to address the software service requirements for mobile equipment in recent years. Microsoft Application Virtualization (App-V, previously named SoftGrid) is a core component of the Microsoft desktop optimization pack for software assurance: it transforms applications into centrally managed virtual services that are ever installed and do not conflict with other applications. The Progressive Deployment System (PDS) [16], Yang's work [17], and FVM [18] employ OS-level virtualization technology to reduce the deploying, updating, and management labor cost of IT as well as the execution environment isolation. All the virtual software packages are managed at central server sites. When a user wants to use some software, the software package will be delivered to the local machine in a streaming way. MobiDesk [19] is a mobile virtual desktop computing hosting infrastructure, and it transparently virtualizes a user's computing session by abstracting underlying system resources in three key areas: display, operating system, and network. It provides a thin virtualization layer that decouples a user's computing session from any particular end-user service, and moves all application logic to hosting providers. In summary, there are some desktop virtualization approaches to providing remote access to a cloud computing environment. However, these approaches only focus on displaying the remote desktop, and do not consider flexible collaboration for live application sharing and migration.

4. CONCLUSION

In cloud computing environment, users can get SaaS subscriptions instead of traditional perpetual-use licenses from software vendors. We have developed a dynamic prototype system to support application sharing and migration on demand among multiple clients. The System provides two key services: a secure multi-user sharing service for the virtual desktop of a VM and multi-VM application sharing and migration. We designed a mechanism for tracking window operation events, and the hidden window areas can be computed quickly. To filter various windows, a proxy-based filtering mechanism is used to deliver a desktop to different users. To achieve the goals of live application sharing and migration between VMs, a presentation redirection approach based on VNC protocol and a VM cloning service based on the Libvirt interface are used.

All these methods have been implemented in this prototype based on extended MetaVNC and the virtual machine monitor KVM. We experimentally verified that these approaches are effective and useful. Several extensions will be made for future work. We are currently developing virtualization-based software as a service platform, and we are exploring how to integrate the system into some mobile based computing environments for flexible collaboration among smartphone users.

5. REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D.Patterson, A. Rabkin, I. Stoica, M. Zaharia, Above the Clouds: A Berkeley Viewof Cloud Computing, 2009.

[2] RajkumarBuyya, Chee Shin Yeo, SrikumarVenugopal, James Broberg, Ivona brandic: cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility, Future Generation ComputerSystems 25 (6) (2009) 599–616.

[3] M. Turner, D. Budgen, P. BreretonTristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, Andy Hopper, Turning software into a service,IEEE Computer 36 (10) (2003) 38–44. http://www.cl.cam.ac.uk/Research/DTG/attarchive/pub/docs/att/tr.98.1.pdf.

[4] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, Andy Hopper, Virtual network computing, IEEE Internet Computing 2(1)(1998)33-38.http://www.cl.cam.ac.uk/Research/ DTG/attarchive /pub/docs/att/tr.98.1.pdf.

[5] MetaVNC, a part of the Collective at Stanford University http://metaVNC.sourceforge.net/.

[6] T.W. Mathers, S.P. Genoway, Windows NT Thin Client Solutions: Implementing Terminal Server and Citrix MetaFrame, Macmillan Technical Publishing,Indianapolis, IN, 1998.

[7] Jianxin Li, JinpengHuai, Chunming Hu, Yanming Zhu, A secure collaboration service for dynamic virtual organizations, Information Sciences 180 (17)(2010) 3086–3107.

[8]Albert Lai, Jason Nieh, BhagyashreeBohra, VijayarkaNandikonda, Abhishek P.Surana, SuchitaVarshneya, Improving web browsing on wireless PDAs usingthin-client computing, in: Proceedings of the 13th International World WideWeb Conference,WWW2004, New York, NY, May 17–22, 2004, pp. 143–154.

[9] Info World Test Center staff. InfoWorld's 2010 Technology of the YearAwards[Z]. Info world,2010.

[10] Albert Lai, Jason Nieh, On the performance of wide-area thin-client computing,ACM Transactions on Computer Systems (TOCS) 24 (2) (2006) 175–209.

[11] Citrix Systems—Virtualization, Networking and Cloud. http://www.citrix.com/.

[12] RealVNC – VNC R remote control software, http:// www.realVNC.com/.

[13] Tom Wall, Virtualisation and thin client: a survey of virtual desktop environments, Technical Report, Dublin Institute of Technology, 2009, http://arrow.dit.ie/ahfrcart/5/.

[14] C. Taylor, J. Pasquale, Improving video performance in VNC under high latency conditions, 2010, in: International Symposium on Collaborative Technologies and Systems, CTS, 17–21 May, 2010, pp. 26–35.

[15] Oren Laadan, Ricardo Baratto, Dan Phung, Shaya Potter, Jason Nieh, DejaView:a personal virtual computer recorder, in: Proceedings of the 21st ACM Symposium on Operating Systems Principles, SOSP 2007, Stevenson, WA, October 14–17,2007, pp. 279–292.

[16] Alpern, Bowen, Joshua Auerbach, et al. PDS: a virtual execution environment for software deployment, in: Proceedings of the First ACM/USENIX International Conference on Virtual Execution Environment, March, 2005.

[17] Yu, Yang, Fanglu Guo, Susanta Nanda, Lapchung Lam, Tzi-cker Chiueh, A Feather weight virtual machine for windows applications, in: Proceedings of the Second ACM/USENIX Conference on Virtual Execution Environments, VEE'06, June, 2006.

[18] C. Sapuntzakis, D. Brumley, R. Chandra, N. Zeldovich, J. Chow, J. Norris,M.S. Lam, M. Rosenblum, Virtual appliances for deploying and maintaining software, in: Proceedings of Seventeenth USENIX Large Installation SystemAdministration Conference, October, 2003.

[19] Ricardo A. Baratto, Shaya Potter, Gong Su, Jason Nieh, MobiDesk: mobile virtual desktop computing, in: Proceedings of the 10th annual international conference on Mobile computing and networking, MobiCom '04, ACM, NewYork, NY, USA, 2004, pp. 1–15.