CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2010; **22**:419–440
Published online 18 September 2009 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.1500

Towards virtualized desktop environment

Xiaofei Liao, Hai Jin[*, †], Liting Hu and Haikun Liu

Services Computing Technology and System Lab, Cluster and Grid Computing Lab, Huazhong University of Science and Technology, Wuhan 430074, China

SUMMARY

Virtualization is being widely used now as an emerging trend. Rapid improvements in network bandwidth, ubiquitous security hazards and high total cost of ownership of personal computers have created a growing market for desktop virtualization. Much like server virtualization, virtualizing desktops involves separating the physical location of a client device from its logical interface. But, the performance and usability of some traditional desktop frameworks do not satisfy end-users. Other solutions, including WebOS, which needs to rebuild all daily-used applications into Client/Server mode, cannot be easily accepted by people in a short time. We present LVD, a system that combines the virtualization technology and inexpensive personal computers (PCs) to realize a lightweight virtual desktop system. Comparing to the previous desktop systems, LVD builds an integrated novel desktop environment, which can support the backup, mobility, suspending and resuming of per-user's working environment, and support synchronous using of incompatible applications on different platforms and achieves great saving in power consumption. We have implemented LVD in a cluster with Xen and compared its performance against widely used commercial approaches, including Microsoft RDP, Citrix MetaFrameXP and Sun Ray. Experimental results demonstrate that LVD is effective in performing the functions while imposing little overhead. Copyright © 2009 John Wiley & Sons, Ltd.

Received 3 January 2009; Revised 29 May 2009; Accepted 17 July 2009

KEY WORDS: virtualization; desktop; VNC; VMM

[*]Correspondence to: Hai Jin, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, Huazhong University of Science and Technology, Wuhan 430074, China.
[†]E-mail: hjin@hust.edu.cn

Contract/grant sponsor: National 973 Basic Research Program of China; contract/grant number: 2007CB310900
Contract/grant sponsor: Program for New Century Excellent Talents in University; contract/grant number: NCET-08-0218
Contract/grant sponsor: Hubei Nature Science Fundation; contract/grant number: 2007ABD009
Contract/grant sponsor: China National Natural Science Fundation (NSFC); contract/grant number: 60703050
Contract/grant sponsor: Wuhan Chenguang Research Plan Foundation; contract/grant number: 200850731350
Contract/grant sponsor: Ministry of Education-Intel Information Technology Special Research Fundation; contract/grant number: MOE-INTEL-08-06

1. INTRODUCTION

Virtualization is being widely used now as an emerging trend. Among the applications, using virtualization technology on the desktops (e.g. the PC workstations and laptops) is an important branch. Although server virtualization is now a mainstream technology for data centers (DCs) and cluster computing communities [1], desktop virtualization has attracted considerable attention in the recent years, both academically [2,3] and commercially [4]. Desktop virtualization has the potential to offer a new, cost efficient paradigm shift to ease the demand for resources while maximizing the return on investment. Combining the potential cost savings with other advantages such as disaster recovery, robustness, scalability and security makes this an attractive computing model to deploy.

However, the challenges of managing multiple desktops and providing QoS guaranteed operation interfaces remain. The following lists several open issues in building a novel desktop environment:

- Lack of customizing and managing end-user desktop environments. End-users want to customize their desktop environments, deploy their desktops rapidly and automatically, move their desktop environments onto different physical machines seamlessly. But, it is difficult to realize these demands based on the traditional schemes, where the desktop environment is tied to operating system and physical machines.
- Lack of suspending and resuming their working states based on multiple checkpoints. Internet suspend/resume (ISR) [5] can suspend or resume the last state of desktop working environment, but it cannot recover one from multiple checkpoints of one desktop environment. Suspending and resuming from multiple working states is harder than only from the last one state. We should propose the incremental policy to save the states data, including memory, CPU and stack states and disk data if necessary.
- Lack of synchronizing incompatible resources from different platforms. In fact, end users want to operate multiple applications and data sets from different platforms, including different operating systems and different hardware platforms, at the same time. The reason is because the server side of all virtual desktop systems uses the remote display protocol to transfer full-screens to the client side. Hence, the user has to face multiple screens and switch among them. As a coarse granularity transferring unit, full-screens increase the amount of data transmitted, resulting in more bandwidth occupation and performance degradation. As a matter of fact, only the application's graphical user interface (GUI) that the user interacts with needs to be transmitted.

On the whole, the architecture of traditional virtual desktop systems imposes fundamental limitations on the customization, convenience, acceleration of pixel data transferring, incompatible applications merging, desktop environment managing. One desktop environment described in the above sentences cannot be implemented based on the traditional ideas, even including the popular desktop solution, WebOS.

In this paper, we propose a lightweight virtual desktop (LVD) management architecture to address the above challenges. LVD combines the advantages of centralized management based on virtualization technology and inexpensive PCs. In our solution, the designed and implemented prototype consists of five parts: the template-based VM repository (TVR), the distributed DC, the applications cluster (APPC), the VM central manager (VCM) and the client terminal. The key technologies behind these parts include 'window-based remote frame buffer (wRFB) protocol', ‘zero-copy’, ‘Magnet algorithm [6]’ and so on. With these parts and technologies, LVD provides a comprehensive suite with important functions: (1) backups, suspends, resumes per-user working environment; (2) supports for rolling back to different working environments by continuous backup at different working periods; (3) supports for using incompatible applications of heterogeneous platforms in a single virtualized platform.

We implement LVD in the cluster with VMs and measure its performance for real applications. Then we compare the performances of our LVD prototype system with those of the most current and widely used desktop systems, including Microsoft Remote Desktop, Citrix MetaFrameXP and Sun Ray. Experimental results on web and audio/video applications demonstrate that LVD performs well in its functions improving the response time while decreasing the power consumptions.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Some typical scenarios of virtualized desktop environments are described in Section 3. The LVD architecture and key technologies are presented in Sections 4 and 5. Section 6 presents the function evaluations. Section 7 presents the performance evaluations. We conclude our work in Section 8.

2. RELATED WORKS

Virtual machine can simulate all the hardware components, such as memory, disk, CPU and so on. A computer that installs a virtual machine monitor can simulate several virtual machines at the same time. Each of these simulated virtual machines can independently install operating systems and run applications simultaneously. One virtual machine managed by the virtual machine monitor (VMM) can be suspended and resumed. The VMM could take a snapshot of a virtual machine’s memory state, and restore the machine states when the snapshot is created.

We divided the previous virtual desktop techniques into two groups: the physical machine-based (PM-based) techniques and virtual machine-based (VM-based) techniques. The PM-based techniques use the remote display protocol or WebOS approach to realize the desktop virtualization, whereas VM-based techniques improve the traditional work by deploying the virtual machines in the server side or client side and thus benefit from better fault isolation and higher resource utilization.

2.1. PM-based techniques

There are two kinds of PM-based techniques. The first is called the remote display protocol-based technique. As one of the earliest desktop virtualization techniques, the thin-client computing model uses a remote display protocol to communicate between a server and a client over the network. The protocol allows graphical displays to be virtualized and served across a network to a client device, while the application logic is executed on the server. The user has remote access to the resource on the server and the server returns a temporary working environment for the client. Typical thin-client protocols include VNC [7], RDP [8], THINC [9], pTHINC [10] and so on. Since the technique mainly depends on continuous display synchronization between user interface on the client and application logic on the server, how to increase the display efficiency becomes the main challenge. Being the first thin client capable of transparently playing full-screen video and audio at full frame rate in both LAN and WAN environments, THINC is the groundwork of other solutions.

The second is the WebOS-based technique. WebOS can quickly build on-demand virtual desktop environments using web technology. Typical WebOS systems include YouOS [11], EyeOS [12], Glide [13] and Orca desktop [14]. The advantage lies in less bandwidth occupation and better cost-performance. However, the drawback is obvious: since a web application executes the front end on the client, and the back end on the server, it fails to be compatible with the original applications and thus requires their reprogramming.

In sum, the PM-based technique has drawbacks which include that (1) the user's working environment cannot be suspended and resumed; (2) the full-screen image is transmitted from the server side to the client side resulting in a great amount of transferring data and degradation of the system performance.

2.2. VM-based techniques

There are also two kinds of VM-based techniques. The first is the server virtualization-based technique [15]. To simplify administration and to reduce management chores and operating costs while maintaining reliability and safeguarding against disasters, the significant benefits of server virtualization technology is now being applied for companies' desktop users. Virtual desktop infrastructure (VDI) [16] is such an integrated desktop virtualization solution combining server virtualization with remote presentation technology. The desktop operating systems and applications run inside the virtual machines and the server side distributes such VMs to each user on the client side. Users use a remote display protocol to access the VM on the server and the server returns all the features of the VMs.

Client virtualization-based technique is the second one. The typical systems are ISR [5,17,18] designed by CMU and Collective designed by Stanford. ISR is a mobile computing technology that can preserve one's uniquely customized computing environment as one package and then move it to different locations. ISR is implemented by layering virtual machine technology on distributed file system technology. It enables a hands-free approach to mobile computing in which commodity hardware may be widely deployed for transient use. Through rapid and easy personalization and depersonalization of anonymous hardware, a user is able to suspend work at one machine and to resume it at another.

Generally, the server virtualization-based technique (1) trades off the user's ability to customize their own environment; and (2) forces the user to switch among multiple platforms while the user wants to run incompatible applications of different operating systems. Client virtualization-based technique requires a powerful client side running the VM.

3. SCENARIOS IN VIRTUALIZED DESKTOP ENVIRONMENT

The following scenarios happen frequently. Figure 1 shows the scenarios in virtualized desktop environments. From the figure, the end-users often ask a question: can I have different working environments at home, at the office, and during a trip? At home, I may want a PC with more entertainment softwares, while at the office, a PC with corporate applications is needed. Furthermore, can I logically suspend a machine at one Internet site, then travel to another site and resume it there on another machine? We call this as the hypothetical capability 'suspend/resume'. Though, some

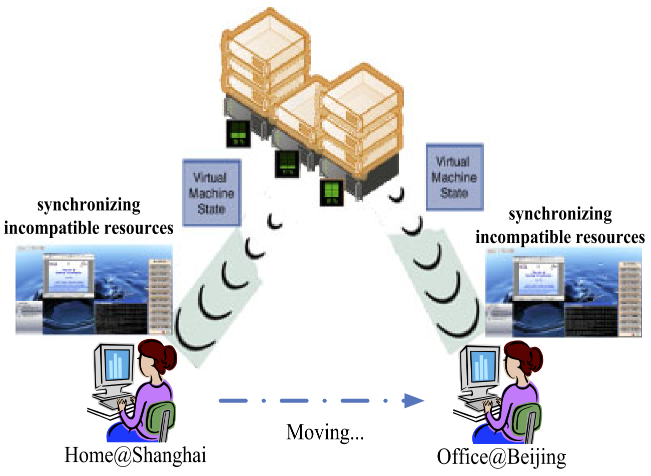


Figure 1. Scenarios in virtualized desktop environment.

previous works ignore the above needs. Collective [19,20] deploys the system disk for each user, the contents of which at every boot are made identical including the image's operating system and applications.

WebOS is a good way to get rid of the limitation of space. However, unfortunately, service providers have to rebuild all applications provided in WebOS. It is a huge task for every company and cannot keep the remaining softwares active. The VDI [16,21] system combines server virtualization with remote presentation technology. The desktop operating systems and applications run inside the virtual machines (VMs) and the server side distributes such VMs to each user on the client side. However, the user still has to face two operating platform screens while interacting with applications on Windows and Linux. Another problem is that a number of previous works have been done to reduce the power consumption, from the perspective of both local techniques [22,23] and cluster-wide techniques [24]. Nonetheless, previous schemes seldom take advantage of the virtualization technology and the characteristic of particular platform to save energy. Most virtual desktop systems fail to consider the reduction of power consumption of the DC. Undoubtedly, 'green computing' has been a hot idea in cluster computing for many years. Anecdotal evidence from DC operators (e.g. [25]) indicates that a significant fraction of the operation cost of these centers is due to power consumption and cooling. To lower the cost of management, we should give prominence to the problem of power consumption.

4. DESIGN OF LVD

The motivation of designing LVD is to solve the challenges discussed in Section 1: (1) how could the applications be customized and configurable freely by the end user; (2) how could per-user states be backed up in a storage-saving way and an optional data-sharing way; (3) how could the incompatible applications running on different operating systems be used by the end user at the

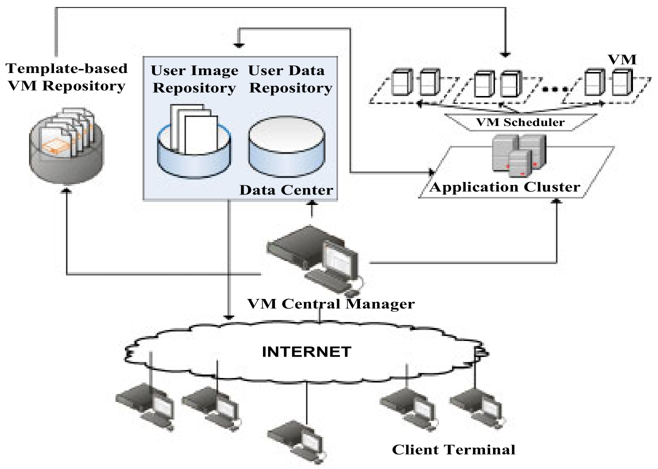


Figure 2. Architecture of LVD system.

same time without switching the platforms; (4) how could the application display commands be transmitted bandwidth effectively. We introduce the design of LVD in this section.

4.1. Overall architecture

As illustrated in Figure 2, The LVD system consists of: (a) the TVR; (b) the DC; (c) the application cluster server (APPC); (d) the VCM and (e) the client terminal. We start by showing how the LVD works from a user's perspective.

If it is the first time for a user to login, first, he is presented with a list of appliances by the VCM on which he can click to begin constructing his personal working environment. Second, the VCM informs the TVR to create the customized VM image including the operating system and applications. Meanwhile, the VCM chooses an appropriate physical machine to boot the image among all the application servers in the cluster on the basis of load-balancing and energy-saving. Third, the user accesses the running VM using our remote display protocol (wRFB). The user transmits the commands to the VM and the VM returns the application window updates to the user interface. Finally, when the end user exits, his working environment, consisting of the user image and user data, is backed up in the DC.

If the user has already logged in previously, he fetches and runs the latest copies of VM image from the DC over the Internet, or resumes the suspended VM image locally when the network is unavailable.

4.2. Template-based VM repository (TVR)

The TVR is responsible for customized VM image provision and updating. TVR contains various operating systems, for example, Linux, Windows including most versions of them, and software. Once a user's request arrives, the ordered operating platform and applications (or components) will be encapsulated and thus a unique VM image is created.

Moreover, TVR needs to be updated constantly. These upgrades include security patches to operating systems or installed applications, installations of new software, upgrades of the operating system to a new version and finally re-installation of the operating system from scratch. All software upgrades, whether small or big, are accomplished in our system with the same mechanism. The system administrator prepares a new version of the appliance and deposits it in the repository. TVR can inform the user that a new version of the appliance is available, encouraging the user to reconstruct the working environment, or TVR can even force a reconstruction to disallow the use of the older version. This update approach has some advantages over package and patch systems such as yum [26], Windows Installer [27]. Patches may fail on some users' computers because of interactions with user-installed software. Our updates are guaranteed to move the appliance to a new consistent state. Users running older versions are unaffected until they perform a reconstruction.

4.3. Data center (DC)

The DC performs the following functions: (1) backups per-user state in terms of copies of VM images; (2) separates per-user VM image into system data and user data that simplifies the management of user data, for example, simplifying the private data sharing among multiple users.

From the view of the data organization, we use the NFS protocol to store the data, which is fast, reliable and simple to support demand paging of large objects, such as images. For our prototype, NFS has the following advantage: in a typical computing session, a user may not access most of the virtual disk image. We arrange the disk data in a tree with small files. Only a small number of these files need to be transmitted to the booted VM in the application cluster. The on-demand transmission mode helps a lot when the bandwidth is limited.

From the view of the data type, the data in DC is divided into two parts: system data and user data. The system data consists of an operating system and all installed applications. User data consists of a user's profile, preferences and private files. The separation has the following advantages: (1) the files in the user data repository could be classified and protected according to their security sensibility; (2) if a user wants to share his data, for example, movies, others can get access to them through NFS conveniently.

4.4. Application cluster (APPC)

After the customized VM image is created by TVR, or fetched from DC, APPC will boot a customized VM on one of its physical machines to provide service to the remote end user.

The highlights of the APPC module are:

- An advanced remote display protocol wRFB is established to address the limit of simultaneously interacting with incompatible applications of different operating systems.

In the previous virtual desktop systems, such as VDI, Sun Desktop Virtualization Solution [28] and Windows Vista enterprise centralized desktop (VECD) [29], each user is assigned to one virtual desktop. If the customer wants to use applications of other platforms, he faces multiple desktops from different VMs and has to switch among them.

Unlike the previous work, we have established a new remote display protocol called wRFB, which grabs and transfers the application GUI images from multiple VMs to the client side. Then

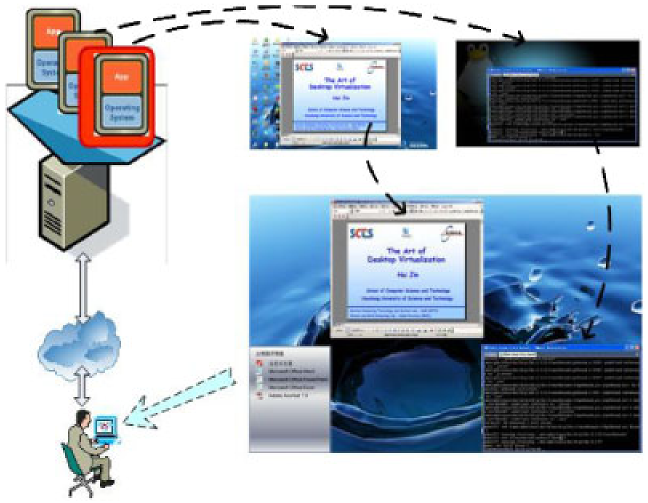


Figure 3. Example of the remote display using wRFB protocol.

all these window images will be merged and displayed on a single virtual desktop, as illustrated in Figure 3.

- A 'zero-copy' technique is proposed to improve the efficiency of the image transferring, which is similar to the share memory extension (SHM) approach in Linux.

The usual method to get the window image is to copy from the process space of the X server to the process space of the virtual desktop system. The 'zero-copy' technique creates a sharing memory segment for the X server and tells to place the image data of the particular window to the sharing memory segment, which is visible for both the X server process and the LVD process. Therefore, the LVD process can get access to the image data directly and thus save the time of copying, as illustrated in Figure 4.

4.5. VM central manager (VCM)

VCM provides the follows interfaces:

- Client-side interface: authenticating the user and keeping the working records of each user. For example, a user may login and work for a while in Hong Kong, then he or she will logout and travel to Seattle. In Seattle, he or she will login and work for a while, then logout, and come back to Hong Kong. It is highly possible that he or she wants to resume the working environment in Hong Kong a few days ago rather than Seattle. Therefore, it is necessary to trace each user's records at different locations.
- Server-side interface: locating and scheduling the VMs in APPC dynamically in a load-balancing and energy-saving way. Undoubtedly, the concept of 'green computing' has attracted much attention recently in cluster computing. The migration of the VM inspires us with a novel

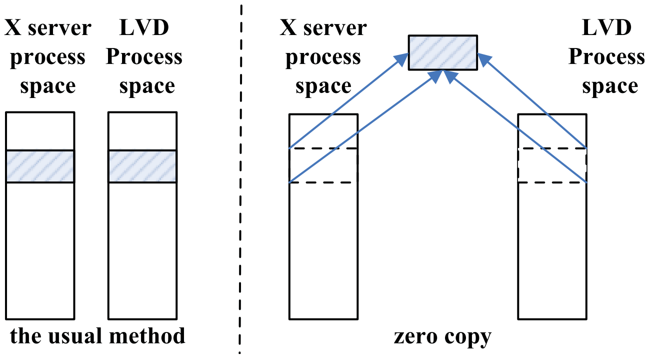


Figure 4. Comparison the usual method with zero copy.

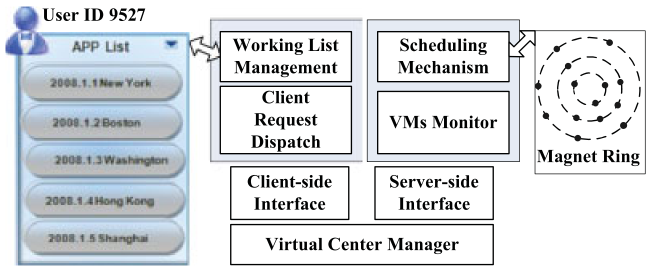


Figure 5. The logic diagram of VCM.

method called ‘Magnet’ to reduce the power consumption, which has been introduced in detail in our previous work [6]. The key technique is to consolidate the load among the nodes on a multi-layer ring-based overlay and then keeps the redundant nodes in ‘low power’ state like ‘deep sleep’ or ‘shut down’.

4.6. Client terminal

The LVD’s client terminal is very simple. LVD presents the user with a list of VM images that he or she has access to, along with the time and locations (see Figure 5). Choosing from a menu, the user can resume any of his working environments.

The client terminal can be any form of computing hardware, from desktops to handheld or wearable computers. The highlights of the client terminal module lie in as follows.

On high bandwidth (e.g. 100 Mbps) networks, the system performs well, the challenge here is that a terminal may be disconnected from the network or the bandwidth is highly occupied. We provide the following solutions: (1) if the terminal is a mobile device, such as a laptop, to maintain the latest VM image locally is necessary; (2) if the terminal is a thin-client, compare the standard operating system and application software (e.g. Windows 2000 and the Microsoft Office suite), the user-specific files are a very small fraction of the image size. Therefore, we distribute the large standard disk image-based blocks widely over the network that can be accessed at high bandwidth from a nearby site. Only the much smaller user-specific state needs to be transmitted at low bandwidth from the DC.

5. KEY TECHNOLOGIES

5.1. VM template scheme

To deploy a virtual machine for an application, the traditional procedure is extremely time-consuming. The process goes like this: first, create a virtual machine with a blank disk image; second, install an operating system (Guest OS) on this virtual machine; and finally, setup the application and its requirements. Our system uses templates to simplify the disk image preparing process. A template is a disk image with pre-installed operating system with or without certain application software. The user can pick a proper template according to the requirements and make a duplicate of it as his/her own disk image.

Though installing an operating system from the beginning for a virtual machine is replaced by a much timesaving cloning, it is still a long-lasting process to duplicate dozens of disk images. With shared external storage, our system could generate a disk image from a virtual machine template in a couple of seconds by utilizing incremental file technology, which is often used in database backup.

An incremental disk image can be used to store the changes to another disk image, without actually affecting the contents of the original image. Figure 6(a) illustrates how incremental disk image works. As it shows, incremental disk image and base disk image are combined to be an integrated disk image for a virtual machine. Base disk image is read-only, which means that all write operations act on incremental disk image. When you want to read an area, you first check to see if that area is allocated within the incremental disk image. If not, you read the area from the base disk image.

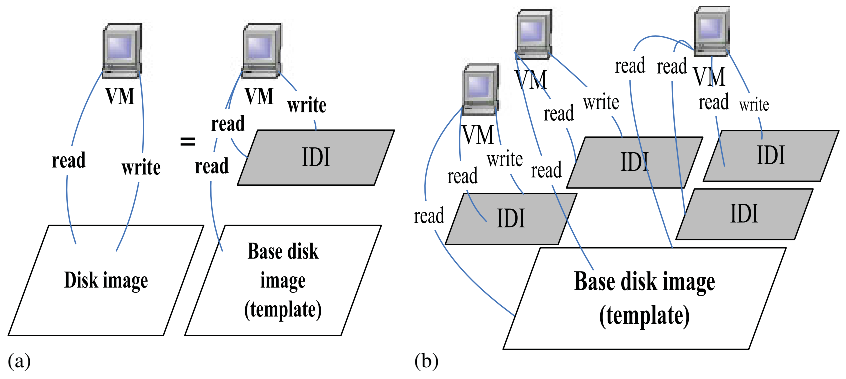


Figure 6. (a) Working principle of incremental disk image and (b) Shared base disk image and multilevel incremental disk images (IDI: incremental disk image).

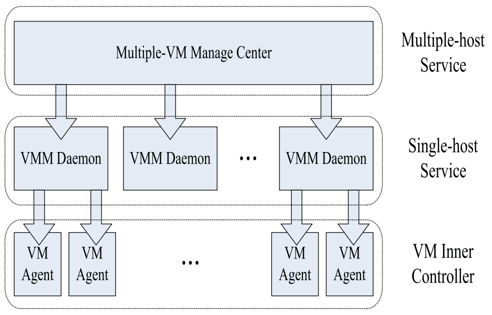


Figure 7. Multiple-VM manager architecture.

Furthermore, as Figure 6(b) shows, multiple incremental images can share one base image, which itself could be an incremental image based on other images. The incremental image files are very small and easy to create and transfer, hence they distinctively reduced the space consuming, and the VM deployment time is lesser than copying the whole raw image file.

5.2. Multiple VM management scheme

Since each desktop user needs several virtual machines in DC, it is necessary to effectively manage huge amounts of virtual machines on multiple physical machines. Hence our multiple-VM management system should have features such as easily monitoring the status of VMs and hosts as well as related information; simultaneously deploying many VMs according to the user's requirement; and automatically allocating proper resource to each VM to balance the workload.

We design a three-layer multiple-VM management architecture in which each layer is loosely coupled with its neighbors. As Figure 7 shows, the underlying layer provides services for the upper neighboring layer. Communications between neighbored layers are mostly through TCP/IP with the exception of a few data link layer packets streamed between VMM Daemon and VM Agent.

Take full virtualization of Xen for example, the monitor and input peripherals are emulated by QEMU that converts the virtual machine's screen output, keyboard input and mouse action into network stream packaged by RFB (Remote Frame Buffer) protocol based on TCP/IP. An RFB client such as VNC Viewer can get the graphic console directly by establishing a connection with the RFB server created by VMM. Unfortunately, as cluster nodes usually share a private internal network isolated from external Internet for security reasons, the user interface might be unable to directly connect to RFB servers running on such nodes. To solve this problem, Manage Center should act as a gateway to forward RFB packets from internal nodes to the user interface.

VMM Daemon is the implementation of single-host service layer providing primitive functions manipulating VMM and virtual machines on local physical machine. It usually works in the background as a system service that receives remote instructions from Manage Center, carries out the tasks assigned and returns the results.

Single-node service layer is the only layer that deals directly with several heterogeneous VMMs such as Xen, VMware, KVM, thus VMM Daemon must hide most trivial details of various VMM interfaces and expose a unified high-level interface to multi-node service layer. Additionally, it is worth noting that different versions of VMM may also have a wide range of distinctions.

Virtual machine agent is installed inside Guest OS. This layer is optional so that the whole multiple-VM management system could work well without it, but if the VM agent is on duty, much more features would be available to administrators. To manage a large number of virtual machines, the VM agent performs the following functions: (1) reports system information including CPU usage, memory allocation, network status, process information, etc.; (2) changes system configuration including network configuration, user password, system services, etc.; (3) controls system including safety shutdown, safety rebooting, formatting disk, installing software, etc.

5.3. Multiple checkpoints scheme

In this paper, we propose a full-system checkpointing mechanism that supports time-travel operations in desktop environments. Virtual machine technology is adopted to implement the complete checkpoint/restart mechanism. The checkpoint consists of all the transient and persistent dates that compose the whole state of a running user execution environment. The Xen mechanism of system snapshot is extended to save the transient data, for example, memory, CPU states; the persistent state (e.g. file system) is incrementally checkpointed with COW mechanism. The implementation of the Xen VM checkpoint mechanism is implemented as the following: (1) change the VM identifier to avoid conflicts with other VM management tasks; (2) stop the VM; (3) dump the VM memory into a file; (4) flushs I/Os and save the CPU state.

Xen currently only supports saving the transient state of a virtual machine, hence only the latest snapshot can be used to rollback to the prior saving state, that is, it does not support multiple checkpoints. Xen assumes that the disk image, that is, the VM file system, is unique for a given VM. As VM restart requires the snapshot of transient state and VM disk image to be at a consistent state to avoid corruption/errors, this assumption implies that Xen is not suitable for providing VM time-travel mechanism, that is, it is not possible to take a complete checkpointing of a VM. To implement a multiple checkpoint, the version of transient state snapshot must strictly associate with the version of VM file system.

However, Xen VM's file system represents the complete OS file system and the installed application data, which may include several gigabytes of data, hence to checkpoint/restart of the whole file system may be an expensive task. We design an incremental checkpoint for the VM disk image using COW mechanism. Our approach reduces the checkpointing size by only capturing the data that has been changed from a prior disk state (possibly the original installation state). Figure 8 presents a structure of incremental image tree for multi-checkpoint. We choose some most popular OS images as basic image files, and then backup these image files into COW format as incremental image files. When the user installs new software, the application files would be saved in the incremental image file. Once a file becomes a basic image, the file is set to read-only authority, and all the write operation would only take effect on the incremental image file, hence the basic file will not be corrupted. A checkpoint version includes the incremental image file, user privacy data, the snapshot file of the transient state, which points the current version of incremental image file. The incremental image file can also be the basic file of another one for mostly sharing the data. But note that once an incremental image file becomes a basic file, its file attribute should be set to read-only, that is, only the leaf node in the tree can be written. When the user execution environment is rolled back to prior versions, a new incremental file will be created to save all the modifications of the system, for example, user profile data, application configuration data. The incremental backup file

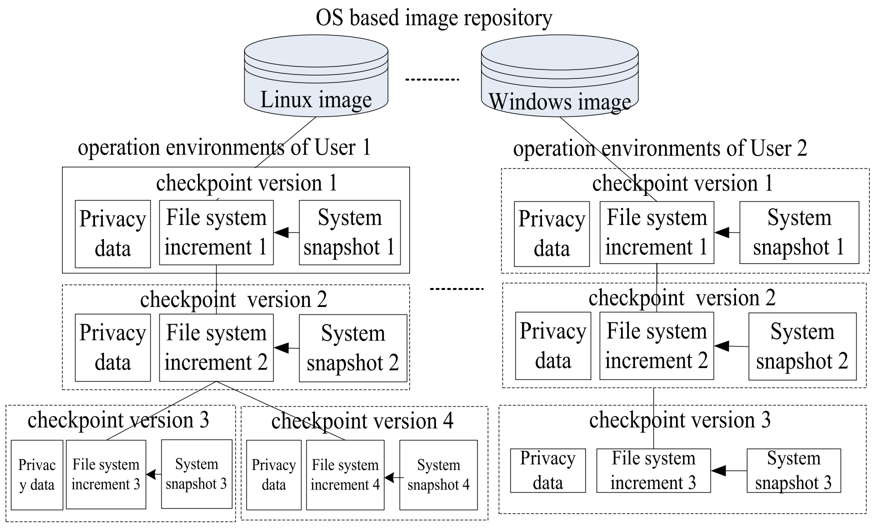


Figure 8. Multi-checkpoint file structure.

is very small, and it is easy to create and transfer, hence it distinctively reduces the disk space consuming.

From the view of the desktop users, their private data should be accessed anywhere with uniform remote view. We use the NFS protocol to store user persistent data, that is, films, photos, archives, which are mostly considered as large file size. For our prototype, NFS has much advantage: it is fast, reliable and simple to support on-demand paging of large objects. The on-demand transmission mode helps a lot when the bandwidth is limited. It also supports multi-platform access transparently.

6. FUNCTION EVALUATION

We have provided some quantitative measurements of the system to give a sense of how the system behaves. In this section we study the feasibility of the system functions.

6.1. Experimental setup

As shown in Figure 9, our test-bed consists of six computers connected by a switched Fast-Ethernet network: two thin clients, a DC, a virtual central manager, a packet monitor, a TVR, a network emulator for emulating various network environments, an application server and a web server used for testing web applications. All computers have an AMD Athlon 3500+ processor and 1 GB DDR RAM. Storage is accessed via iSCSI protocol from a NetApp F840 network attached storage server (NAS). The guest kernel is Linux 2.4.18 ported to UM-Linux, and the host kernel for UM-Linux is a modified version of Linux 2.4.18. The virtual machine is configured to use 512 MB of RAM, the memory page size is 4 kbytes, page fault service time is 10 ms and the context switch time is 0.1 ms.

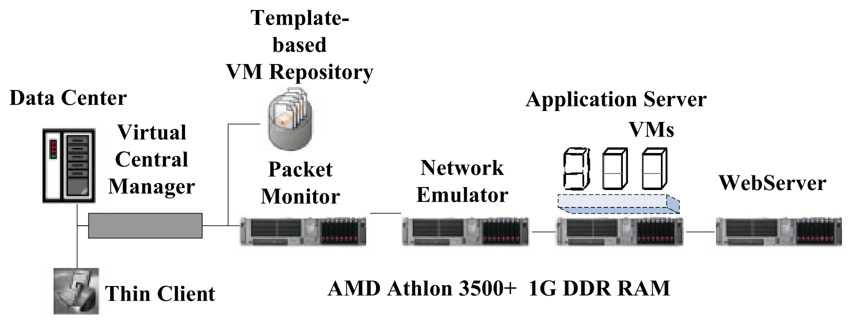


Figure 9. Test-bed configuration.

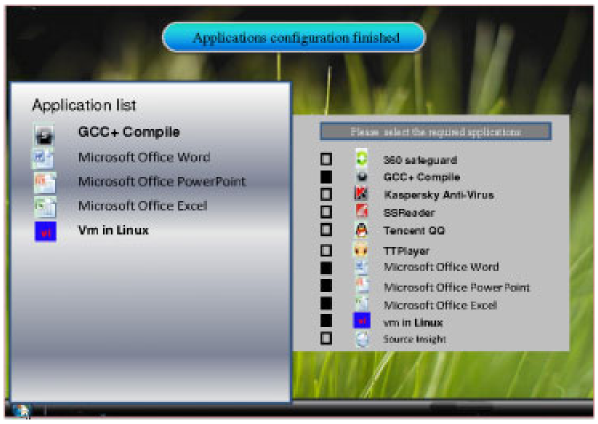
As shown in Figure 9, the network simulator machine is placed in the middle of the network between the thin client and server machines to control the available network bandwidth between the thin client and server. This is done using a dedicated PC installed with two 100 Mbps Ethernet interfaces running the Cloud [30], a software network bandwidth simulator that could vary the effective network bandwidth between the two network interfaces. The thin clients and servers are separated from one another on isolated 100 Mbps networks, which are then connected via the network simulator. Here, the server-side network is connected to one of the network interfaces PC and the client-side network is connected to the other network interface in the network simulator. We refer to the server-side interface as the East gateway and the client-side interface as the West gateway. The Cloud software can then be used to vary the bandwidth on either of the two gateways from the maximum of 100 Mbps to as little as 2400 bps. To ensure that this simulator does not itself introduce extra delay into our tests, we measure round-trip ping times with and without the simulator between the client and the server. There are no significant differences and round-trip ping times are roughly 0.57 ms in both cases.

A packet monitor is used in the test-bed to monitor and record traffic on the network between the client and the server. The packet monitor used is a PC running Etherpeek 4, a software packet monitor that timestamps and records all packet traffic visible by the PC. As shown in Figure 9, the packet monitor is generally attached to the client-side network to monitor client-side network traffic, though it can be moved to other parts of the test-bed to monitor server-side traffic as well.

6.2. Function evaluation

In function evaluation, we have evaluated three main functions of the LVD system: customizing per-user's working environment, suspending/resuming per-user's state and the synchronous use of incompatible applications on different platforms.

If it is the first time for a user to login, first, he or she is presented with a list of appliances by the VCM on which he can click to begin constructing his personal working environment, as illustrated in Figure 10. If the user wants to leave for a while, he or she can suspend his working environment. When he or she comes back, the user can resume it. Finally, as Figure 10 shows, synchronous use of Microsoft Office Word 2007 on Window XP and Qt designer 3.3.3 on Linux Redhat 7.0 can be obtained.


Applications configuration finished
Application list
GCC+ Compile
Microsoft Office Word
Microsoft Office PowerPoint
Microsoft Office Excel
Vm in Linux
Please select the required applications

- ☐ 360 safeguard
- ■ GCC+ Compile
- ☐ Kaspersky Anti-Virus
- ☐ SSReader
- ☐ Tencent QQ
- ☐ TTPlayer
- ■ Microsoft Office Word
- ■ Microsoft Office PowerPoint
- ■ Microsoft Office Excel
- ■ vm in Linux
- ☐ Source Insight

Figure 10. The customization of operating system and applications.

7. PERFORMANCE EVALUATION

In this section, we assess the LVD display performance in various network environments. We run each of the three application benchmarks on the baseline PC platform and the five thin-client platforms on different operating systems. The five specific platform configurations considered are PC running LVD on Linux (LVD), Microsoft Terminal Services RDP 5.0 on Windows 2000 (RDP Win2K), Citrix Metaframe 1.8 running on Windows 2000 (Citrix Win2K), LapLink 2000 on Windows NT 4.0 Terminal Server Edition (LapLink WinNT) and Sun Ray.

7.1. Application benchmarks

To measure the performance of the thin-client platforms, we use a simple Java latency benchmark and a selection of benchmarks from the Ziff-Davis i-Bench benchmark suite, version 1.01, as shown in Table I. The Java latency benchmark is used to measure the latency of basic operations on a thin-client platform, such as responding to a single keystroke. The i-Bench benchmarks used are the Web Text Page Load and Flash benchmarks, which can be used to provide a measure of web-based and multimedia-oriented application performance. All the benchmarks are designed to be executed within a web browser to provide a common application environment across different platforms.

For our evaluation study, we run the three application benchmarks on each platform and use the network simulator to vary the network bandwidth between client and server to examine the impact of network bandwidth on the thin-client performance. All the tests are run on each system at the bandwidth listed in Table II. Eight different network bandwidth configurations are considered, representing LAN, T1, DSL and ISDN. We focus our evaluation on network bandwidth and do not consider the different network latencies associated with different network technologies, which is beyond the scope of this paper.

Table I. Characteristics of the application benchmarks.

Benchmark type	Application benchmark	Operation
Latency benchmark	Java applet	Typing a character, scrolling text, filling a screen region. Downloading an image
Web-based benchmark	Web text page load	Downloading a sequence of 109 web pages. Scrolling down 200 pixels
Multimedia oriented	Benchmark Flash benchmark	Streaming a 98 kB Macromedia Flash animation chip from the server side (uses vector graphics and contains 315 550*400 frames)

Table II. Bandwidth tested.

Network type simulated	Bandwidth tested
LAN	100 Mbps, 10 Mbps
T1	1.5 Mbps
DSL	512 kbps
ISDN	256 kbps, 128 kbps

7.2. Latency benchmark results

We run the latency benchmark using each thin-client platform at the full 100 Mbps network bandwidth to measure the latency of each of the thin-client platforms when the network capacity is not the limiting factor. We measure both the latency and the data transferred for each of the four operations, draw letter, fill red box, scroll text and load bitmap. These results are shown in Figures 11 and 12, respectively.

Limits. To achieve a good subjective performance, the latency between user input and system response should be below the threshold of human perception. For simple tasks such as typing, cursor motion or mouse selection, the system response time should be less than 50–150 ms to keep users from noticing a delay.

Compared with other platforms, LVD is able to complete any operations within 100 ms, including the basic draw letter, fill box and scroll text operations. Comparing Figures 11 and 12, we can see that the latency and the data transferred are not at all correlated in many cases. While LVD has less latency for most of the tests, it also requires more data transfer.

7.3. Web page load benchmark results

We run the Web benchmark on each of the thin-client platforms using network bandwidth from 128 kbps to 100 Mbps, as listed in Table II. We also run the Web benchmark on the baseline PC platform for comparison. For each run, we measure the total time required to display all 109 web pages of the Web benchmark and logged complete packet traces using the packet monitor. We find that at high bandwidth, the page load times are limited by the server speed as the server is completely busy. Figures 13 and 14 illustrate the total normalized web page download times and total data transferred for each run, respectively.

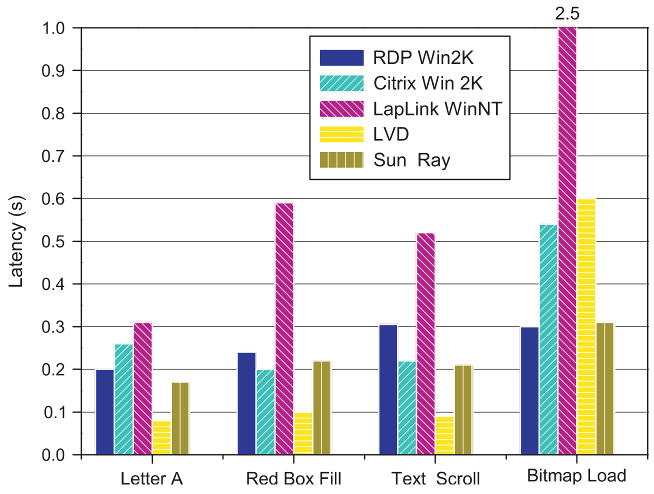


Figure 11. Latency test time to completion.

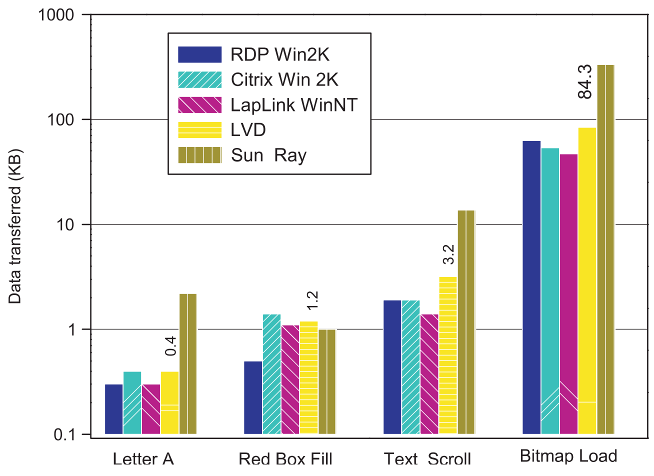


Figure 12. Latency test data transferred.

As shown in Figure 13, almost all the platforms complete the Web benchmark in less than 50 s at network bandwidth with 4 Mbps or greater, corresponding to an average of less than half a second per page.

Limits. At network bandwidth below 4 Mbps, the performance of the thin-client platforms begins to degrade. As shown in Figures 13 and 14, LVD does not take longer to complete the Web benchmark at lower bandwidth, but instead loses data resulting in missed or incomplete screen updates. On the other hand, RDP and Citrix behave in a similar manner to the baseline PC client by

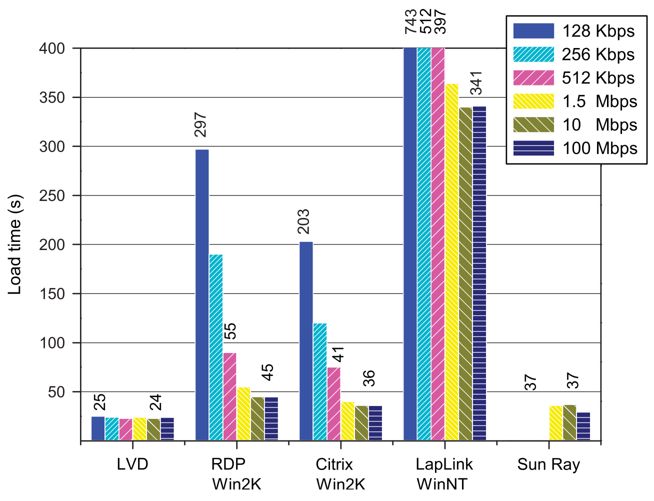


Figure 13. Web page load time.

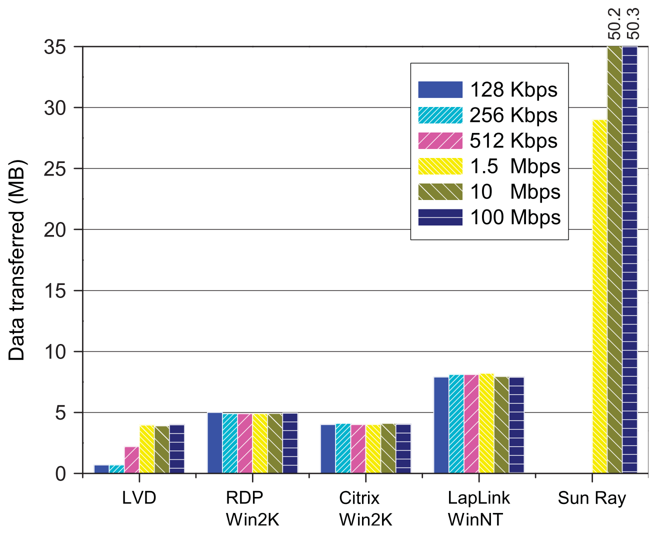


Figure 14. Web page data transferred.

taking longer to complete the Web benchmark as the network bandwidth decreases, but continuing to send the same amount of data even at lower bandwidth. As shown in Figure 13, LVD transfers less and less data at the lower network bandwidth unlike the other platforms that all transfer roughly the same amount of data.

In general, while a comparison of Citrix with Sun Ray shows that the graphics-based encoding approach is more efficient for screen updates we note that LVD needs to send less data than those of

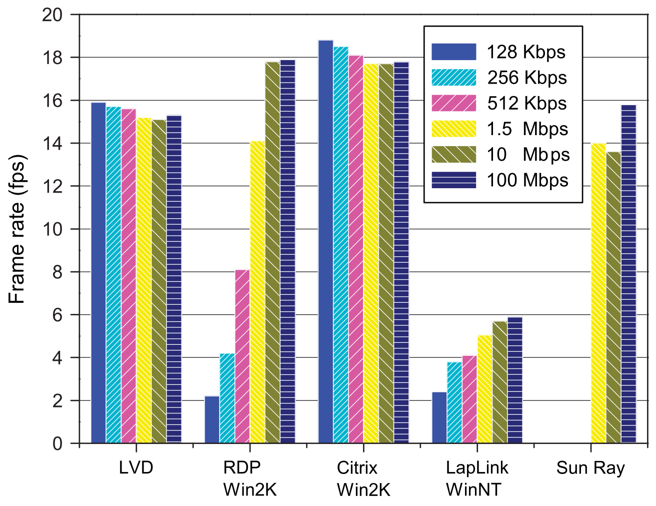


Figure 15. Flash test frame rate.

RDP in WinNT and Win2K, and Citrix in WinNT. Instead, the LVD results indicate that for Web-based applications, a pixel-based encoding approach could encode screen updates with comparable efficiency as graphics-based encoding approaches.

7.4. Flash benchmark results

We run the Flash animation test on each of the thin-client platforms using the network bandwidth listed in Table II, and also run the benchmark on the baseline platforms for comparison. The results are illustrated in Figures 15 and 16.

We find that a frame rate, about 16 fps, produces good subjective results, with smooth display, no skipped screens and no tearing or jerky movement. The 100 kB animation is downloaded completely to the server prior to playback, hence the bandwidth does not affect the frame rate on the baseline. However, we examine the time required for this download in the baseline case and find that there are no significant delays except a brief one when the bandwidth is limited to 128 kbps. In general, the LVD system experiences a similar fall-off in the performance at 128 kbps bandwidth or below, indicating that LAN bandwidth is required to support multimedia applications.

Limits. As with the web test results, LVD is slightly less efficient than the platforms with graphics-based encodings. LVD maintains an almost constant rate for the benchmark at all bandwidth, but drops many screens on the client's display at the low end of the range. Its performance is only comparable in smoothness to the baseline when the data transferred reaches a plateau.

7.5. Power reduction results

After analyzing the characteristics of workflow of LVD, we propose a new schedule policy called Magnet for cluster with VMs to achieve the energy savings, which can be accessed in [6]. Magnet

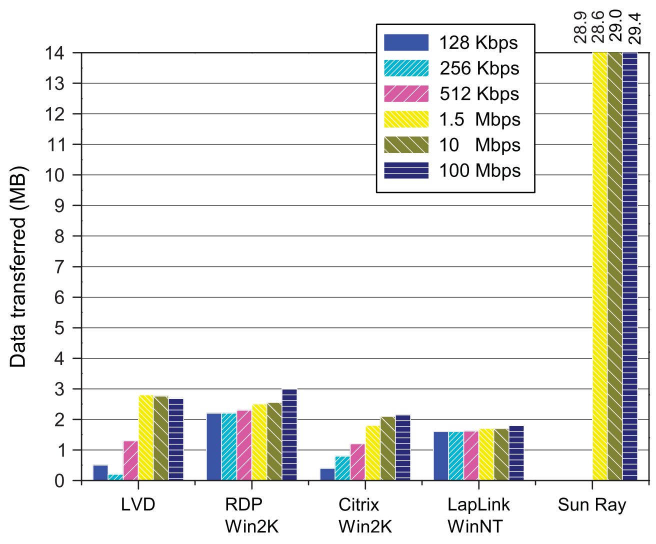


Figure 16. Flash test data transferred.

keeps track of all active nodes and organizes them into concentric, non-overlapping rings in terms of gradually decreasing the workload, hence it is easy to squeeze the existing running jobs that are widely distributed among lightweight nodes and then deliver them to a subset of current active nodes and it is also easy to release the over-weighted nodes, thereby (1) turning off the redundant nodes to save energy when the system is in non-intensive computing state; (2) transferring violating jobs or big jobs to the free nodes when the system is in intensive computing state to obtain performance gains. We apply the policy to the application cluster. The experimental measurements show that the new method can reduce the power consumption by 74.8% over base at most with certain adjustably acceptable overhead.

8. CONCLUSIONS

This paper presents LVD, a prototype of a system management architecture for managing desktop computers. This paper concentrates on the design issues of a complete system. By combining 'wRFB protocol', 'zero-copy' and 'Magnet algorithm' with the technology of virtualization, LVD provides several novel functions including the backup, mobility, suspending and resuming of per-user's working environment, the customization of working environment and the synchronous using of incompatible applications on different platforms. In this paper, we also describe several key technologies in the lightweight virtualized desktop. Some experiments show the good performance of the system in addition to achieving great saving in power consumption. In the future, we will try to optimize the display performance by exploring more intelligent schemes.

The demo video for the lightweight virtualized desktop system can be obtained from [31].

ACKNOWLEDGEMENTS

This work is supported by National 973 Basic Research Program of China under grant No. 2007CB310900, Program for New Century Excellent Talents in University under grant NCET-08-0218, Hubei Nature Science Fundation under grant No. 2007ABD009, China National Natural Science Fundation (NSFC) under grant No. 60703050, Wuhan Chenguang Research Plan Foundation with No. 200850731350 and the Ministry of Education-Intel Information Technology Special Research Fundation under grant No. MOE-INTEL-08-06.

REFERENCES

1. Ruth P, McGachey P, Xu D. VioCluster: Virtualization for dynamic computational domains. *Proceedings of IEEE International on Cluster Computing* (*Cluster 2005*), Boston, U.S.A., September 2005.
2. Kim EJ, Yum KH, Link GM, Vijaykrishnan N, Kandemir M, Irwin MJ, Yousif M, Das CR. A holistic approach to designing energy-efficient cluster interconnects. *IEEE Transactions on Computers* 2005; **54**(6):660–671.
3. Nieh J, Yang SJ, Novik N. Measuring thin-client performance using slow-motion benchmarking. *ACM Transactions on Computer Systems* 2003; **21**(1).
4. Karissa M, Mahmoud P. Virtualization. Virtually at the desktop. *Proceedings of the 35th Annual ACM SIGUCCS Conference on User Services*, Florida, U.S.A., 2007.
5. Kozuch MA, Helfrich CJ, O'Hallaron D, Satyanarayanan M. Enterprise client management with internet suspend/resume. *Intel Technology Journal* 2004; **8**(4):313–323.
6. Hu L, Jin H, Liao X, Xiong X, Liu H. Magnet: A novel scheduling policy for power reduction in cluster with virtual machines. *Proceedings of 2008 IEEE International Conference on Cluster Computing* (*Cluster 2008*), Japan, September 2008; 13–22.
7. Lange JR, Dinda PA, Rossoff S. Experiences with client-based speculative remote display. *Proceedings of USENIX 2008 Annual Technical Conference on Annual Technical Conference* (*USENIX 2008*), San Diego, U.S.A., June 2005.
8. Mirakhorli M, Rad AK, Aliee FS, Mirakhorli A, Pazoki M. RDP technique: Take a different look at XP for adoption. *Proceedings of the 19th Australian Conference on Software Engineering* (*ACM ASWEC 2008*), Perth, Australia, March 2008.
9. Baratto R, Kim L, Nieh J. THINC: A virtual display architecture for thin-client computing. *Proceedings of the 20th ACM Symposium on Operating Systems Principles* (*SOSP 2005*), Brighton, U.K., 2005.
10. Kim J, Baratto RA, Nieh J. pTHINC: A thin-client architecture for mobile wireless web. *Proceedings of the 15th International World Wide Web Conference* (*WWW 2006*), Edinburgh, Scotland, 2006.
11. youOS project [Online]. Available at: http://en.wikipedia.org/wiki/YouOS [15 May 2008].
12. Pierce JS, Nichols J. An infrastructure for extending applications' user experiences across multiple personal devices. *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology* (*UIST 2008*), Monterry, U.S.A., October 2008.
13. Glide project [Online]. Available at: http://www.glidedigital.com/ [15 May 2008].
14. Orcaa desktop [Online]. Available at: http://www.orcaa.com/ [15 May 2008].
15. Ongaro D, Cox AL, Rixner S. Scheduling I/O in virtual machine monitors. *Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments* (*VEE 2008*), Seattle, U.S.A., March 2008.
16. VMware. VMware Infrastructure 3 VDI Server Sizing and Scaling, 2006.
17. Sathyanarayanan M, Gilbert B, Toups M, Tolia N, Surie A, O'Hallaron DR, Wolbach A, Harkes J, Perrig A, Farber DJ, Kozuch MA, Helfrich CJ, Nath P, Lagar-Cavilla HA. Pervasive personal computing in an internet suspend/resume system. *IEEE Internet Computing* 2007; **11**(2):16–25.
18. Satyanarayanan M, Kozuch M, Helfrich C, O'Hallaron DR. Towards seamless mobility on pervasive hardware. *Pervasive and Mobile Computing* 2005; **1**(2):157–189.
19. Chandra R, Zeldovich N, Sapuntzakis C, Lam MS. The collective: A cache-based system management architecture. *Proceedings of the Second Symposium on Networked Systems Design and Implementation* (*NSDI 2005*), Massachusetts, U.S.A., May 2005; 259–272.
20. Sapuntzakis C, Lam M. Virtual appliances in the collective: A road to hassle-free computing. *Proceedings of 9th Workshop on Hot Topics in Operating Systems* (HotOS 2003), Hawaii, U.S.A., 2003; 55–60.
21. Jim G. CTO virtualization roundtable: Part I. *Communications of the ACM* 2008; **51**(11):47–53.
22. Burd T, Pering T, Stratakos A, Brodersen R. A dynamic voltage scaled microprocessor system. *Proceedings of IEEE International Solid-State Circuits Conference*, San Francisco, U.S.A., 2000; 294–295.

23. Lai A, Nieh J. Limits of wide-area thin-client computing. *ACM SIGMETRICS Performance Evaluation Review* 2002; **30**(1):228–239.
24. Pinheiro E, Bianchini R, Carrera E, Heath T. Dynamic cluster reconfiguration for power and performance. *Compilers and Operating Systems for Low Power*, Benini L, Kandemir M, Ramanujam J (eds.). Kluwer Academic Publishers: Dordrecht, 2003.
25. Hopkins M. The onsite energy generation option. *The Data Center Journal* 2004 [Online]. Available at: http://datacenterjournal.com/News/Article.asp?articleid=66 [15 August 2009].
26. Yellowdog updater modified (yum) [Online]. Available at: http://linux.duke.edu/projects/yum/ [15 May 2008].
27. Wilson P. *Definitive Guide to Windows Installer* (1st edn). Apress Inc.: New York, U.S.A., 2004.
28. Ponder W. Sun Desktop Virtualization Solution: Desktop Virtualization Blueprint [Online]. Available at: http://www.sun.com/software/sdis/wp_desktop_virtualization_blueprint.pdf [11 October 2006].
29. Fritsche G. Windows Vista: Implementation challenges. *Proceedings of the 35th Annual ACM SIGUCCS Conference on User Services*, Florida, U.S.A., 2007; 113–117.
30. Shunra Software [Online]. The Cloud. Available at: http://www.shunra.com [15 May 2008].
31. Demo Video for The Lightweight Virtualized Desktop System [Online]. Available at: http://grid.hust.edu.cn/xfliao/LVD.html [15 May 2008].