LVD: A Lightweight Virtual Desktop Management Architecture*

Xiaofei Liao, Xianjie Xiong, Hai Jin, and Liting Hu

Services Computing Technology and System Lab
Cluster and Grid Computing Lab
School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, 430074, China
{xfliao,hjin}@hust.edu.cn

Abstract. 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. We present the LVD, a system that combines the virtualization technology and inexpensive personal computers (PCs) to realize a lightweight virtual desktop system. Compared with the previous thin client systems, LVD supports the backup, mobility, suspending and resuming of per-user's working environment; it supports the customization of operating system and applications for each user; it supports synchronous using of incompatible applications on different platforms; it achieves great saving in power consumption. LVD consists of five modules--- the template-based VM repository, the data center, the application cluster, the VM central manager, and the client terminal, in which we have proposed wRFB protocol, magnet algorithm and the like to perform the above functions. We have implemented LVD in a cluster with VMs and compared its performance against widely used commercial approaches. Experimental results demonstrate that LVD is effective in performing the functions while imposing little overhead.

Keywords: RDP, desktop virtualization, virtual machine.

1 Introduction

Although server virtualization is now a mainstream technology for data centers and cluster computing communities, desktop virtualization has attracted considerable attention for recent years, both academically [9] and commercially. Desktop virtualization has the potential to offer a new, cost efficient paradigm shift to ease the demand for the resources while maximizing return on investment. Combining the potential cost

* This work is supported in part by National Natural Science Foundation of China (NSFC) under grants No.60703050, National 973 Basic Research Program of China under grant No.2007CB310900, Hubei Natural Science Foundation under grant No.2007ABD009, the Ministry of Education-Intel information technology special research fund under grant No.MOE-INTEL-08-06 and Wuhan Chengguang Plan under grant No.200850731350.

L. Boursas et al. (Eds.): SVM 2008, CCIS 18, pp. 25–36, 2008.

savings with other advantages such as disaster recovery, robustness, scalability, and security make this an attractive computing model to deploy.

However, the architecture of current virtual desktop systems imposes fundamental limitations on the customization, convenience, acceleration of pixel data transfers and energy-savings, so the challenges remain as below.

First, most virtual desktop systems fail to support users to customize their own environments and back up them, not to mention the function of suspend/resume. The following scenarios happen frequently: people have different working environments at home, at the office and during the trip. At home, one may want a PC with more entertainment software, while at the office a PC with corporate applications is needed. Furthermore, people should logically suspend a machine at one Internet site, then travel to some other sites and resume on another machine. We call the hypothetical capability *suspend/resume*. Nevertheless, some previous work ignores the above needs. Collective [4] 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.

Second, it is impossible to interact with incompatible applications of different operating systems synchronous without switching the platforms. The reason lies in that 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.

The third shortcoming follows from the second. As a large granularity transferring unit, full-screens increase the amount of data transmitted, resulting in more bandwidth consumption and performance degradation. As a matter of fact, only the application GUI that the user interacts with needs to be transmitted.

Finally, most virtual desktop systems fail to consider the reduction of power consumption of the data center. Undoubtedly, *green computing* has been a hot idea in cluster computing for recent years. Anecdotal evidence from data center operators [7] 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.

For the first challenge, WebOS is a good option to get rid of the space limit. However, since all operations are performed by JAVASCRIPT which is less than 1/50 of Java, 1/200 of php and 1/500 of C for speed, WebOS can only be applied to deal with lightweight programs. Besides, high network latency will lead to slow response time. For the second challenge, VDI [14] combines server virtualization with remote presentation technology. However, the user still has to face two operating platform screens while interacts with applications on different platforms. Little previous work concerns the third challenge. For the fourth challenge, lots of previous works have been done to reduce the power consumption, from the perspective of both local techniques and cluster-wide techniques [11]. Nonetheless, previous schemes seldom take advantage of the virtualization technology and the characteristic of particular platform to save the energy.

We propose *lightweight virtual desktop* management architecture (LVD) to address the challenges. Our model consists of five modules: the template-based VM repository (TVR), the data center (DC), the application cluster (APPC), the VM central manager (VCM), and the client terminal. The key technologies behind these modules include *wRFB protocol*, *Magnet algorithm* and the like. LVD provides a comprehensive suite of important functions:

- Backs up, suspends, resumes per-user working environment (see TVR).
- Supports rolling back to different working environment by continuous backup at different working locations (see DC).
- Supports using incompatible application of heterogeneous platforms in a single platform (see APPC).
- Saves energy in a dynamical and global way (see VCM).

We have implemented LVD in the cluster with VMs and measured its performance on real applications. We also have compared our LVD prototype system against the most current and widely-used virtual desktop systems, including Microsoft Remote Desktop, Citrix MetaFrameXP and Sun Ray. Experimental results demonstrate that LVD is effective in performing the functions while imposing little overhead.

The rest of this paper is organized as follows. Section 2 discusses the related work. The LVD architecture is described in Section 3. Section 4 evaluates the performance. We conclude our work in Section 5.

2 Related Works

Virtual machine can simulate all the hardware components, like memory, disk, CPU and so on. A computer with virtual machine monitor installed can simulate several virtual machines at the same time. Each of these simulated virtual machines can independently install operating system, run applications simultaneously. The virtual machine monitor could suspend the virtual machine managed by the VMM, and resume one virtual machine which has been suspended before. The VMM could take a snapshot of a virtual machine's memory state, and restore the machine state to the state when the snapshot is created.

We divide 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 benefits from better fault isolation and higher resource utilization.

Remote display protocol based technique. As the earliest desktop virtualization technique, 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. Typical thin client protocols include VNC [12], RDP, THINC [1], pTHINC [8] and the like. 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 is 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.

WebOS based technique. WebOS is another fashion that can quickly build on-demand virtual desktop environments using web technology. A WebOS is also known as a webtop, and Ebrahim Ezzy's definition probably explains it best: "A webtop (derived

from "desktop") pushes that replication to its limit. Also known as a WebOS, it is basically a virtual desktop on the web. It is a simple, less bloated, less featured and remotely accessible operating environment that runs in a browser. It delivers a rich desktop-like experience, coupled with various built-in applications." Typical WebOS systems include YouOS [5], EyeOS, Glide [6] and Orca desktop [10]. The advantage lies in that less bandwidth consumption 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 the reprogramming of them.

In sum, PM-based technique has drawbacks including that (1) the user's working environment can not 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.

Server virtualization based technique. To simplify administration and to reduce management 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) 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 the full features of the VMs.

Client virtualization based technique. The typical systems are *Internet Suspend/Resume* (ISR) [13] designed by CMU and Collective designed by Stanford. ISR is a mobile computing technology which can preserve one's uniquely customized computing environment as one package and then moves 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, server virtualization based technique (1) trades off the user's ability to customize their own environment; (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 LVD Design

The motivation of designing LVD is to solve the challenges discussed above: (1) how the applications be customized and configurable freely by the end user; (2) how per-user states be backed up in a storage-saving way and an optional data-sharing way; (3) how the incompatible applications running on different operating systems be used by the end user at the same time without switching the platforms; (4) how the great savings be achieved in power consumption by our scheduling policy; (5) how the application display commands be transmitted effectively.

3.1 Overview

As illustrated in Fig.1, LVD system consists of *template-based VM repository* (TVR), *data center*, *application cluster* (APPC), *VM central manager* (VCM), and *client terminal*. We start by showing how LVD works from a user's perspective.

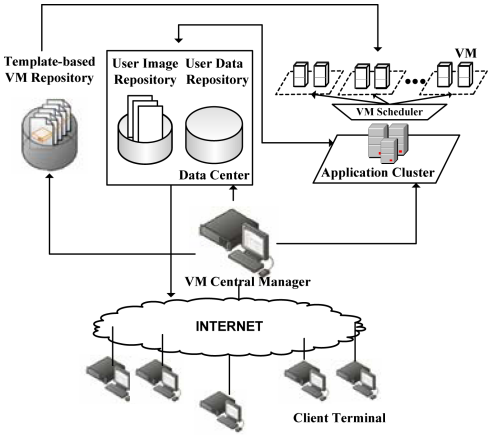


Fig. 1. Architecture of LVD system

When a user logs in at the first time, firstly, 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 template-based VM repository 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 data center.

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

3.2 Template-Based VM Repository (TVR)

The template-based VM repository is responsible for customized VM image provision and updating. TVR contains all versions of various operating systems 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 need to be updated constantly. All software upgrades, be they small or big, is 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 the TVR can even force a reconstruction to disallow use of the older version.

This update approach has some advantages over package and patch systems like yum [15], RPM [2], Windows Installer, and Windows Update. 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.

3.3 Data Center (DC)

The data center performs the following functions:

- Backs up per-user state in terms of copies of VM images.
- Separates per-user VM image into system data and user data.

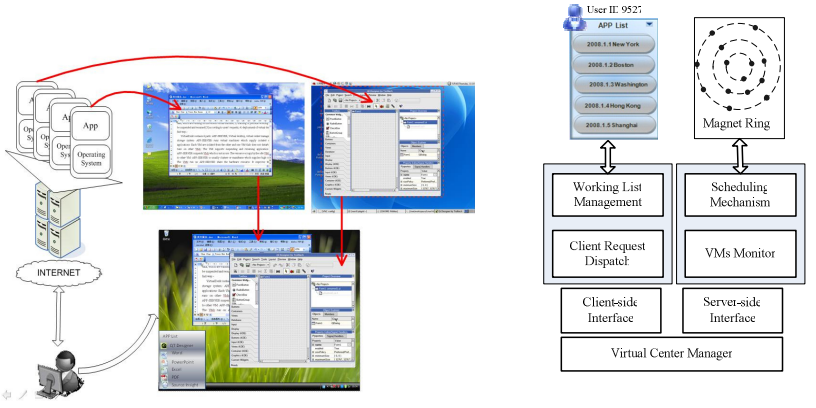
From the view of the data organization, we use the NFS protocol to store the data, which is fast, reliable and simple to support demand paging of large objects, like the images. For our prototype, NFS has the following advantage: in a typical computing session, a user may not access most of the virtual disk image. We arrange the disk data in a tree of small files, only a small number of these files will 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, e.g. movies and files, others can get access to them through NFS conveniently.

3.4 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 machine to provide service to the remote end user. The highlights of APPC module is an advanced remote display protocol *wRFB* (window-based remote frame buffer).

In the previous virtual desktop systems, such as VDI, Sun Desktop Virtualization Solution and Windows *Vista Enterprise Centralized Desktop* (VECD) [4], each user is assigned to a 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, wRFB grabs and transfers the application GUI images from multiple VMs to the client side. Then all these window images will be merged and displayed on a single virtual desktop, as illustrated in Fig.2(a).



(a) (b)

Fig. 2. (a) wRFB protocol; (b) The logic diagram of VCM

3.5 VM Central Manager (VCM)

VCM provides the follows interfaces, as illustrated in Fig.2(b): 1) client-side interface: authenticating the user and keeping the working records of each user. For example, a user may login to the client in Hong Kong, work for a while, log out, travel a great distance to Seattle, login, work, log out, and come back to Hong Kong. It is highly possible that he 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; 2) 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 method called *Magnet* to reduce the power consumption, which has been introduced in detail in our previous work [16]. The key technique is to consolidate the load among the nodes on a multilayer ring-based overlay and then keep the redundant nodes in *low power* state like *deep sleep* or *shut down*.

3.6 Client Terminal

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., 100Mbps) 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 mobile computers like laptop, store the latest VM image locally; (2) if the terminal is a thin-client, compare to 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 which 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 data center.

4 Performance Evaluation

We provide some quantitative measurement of the system to give a sense of how the system behaves and study the system performance in a various network environment.

As shown in Fig.3, our testbed consists of six computers connected on a switched Fast Ethernet network: two thin clients, a data center, a virtual central manager, a packet monitor, a template-based VM repository, 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 1GB 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 512MB of RAM, the memory page size is 4KB, page fault service time is 10ms, and the context switch time is 0.1ms.

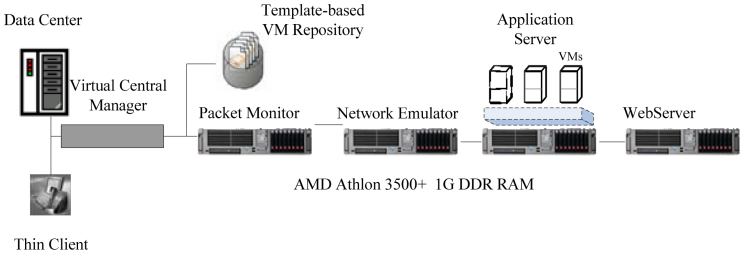


Fig. 3. Tested configuration

We have assessed the LVD display performance in a various network environment. 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

Table 1. 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 54 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)

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.

As illustrated in Table 1, we use a simple Java latency benchmark to measure the latency of basic operations on a thin client platform and a selection of benchmarks from the Ziff-Davis i-Bench benchmark suite to provide a measure of web-based and multimedia-oriented application performance.

4.1 Latency Benchmark Results

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 Fig.4(a) and Fig.4(b), respectively.

Advantage. Compared with other platforms, LVD is able to complete any operations within 100ms, including the basic draw letter, fill box, and scroll text operations. Comparing Fig.4(a) with Fig.4(b), we can see that the latency and the data transferred are not at all correlated in many cases. While LVD had less latency for most of the tests, it also requires more data transfer.

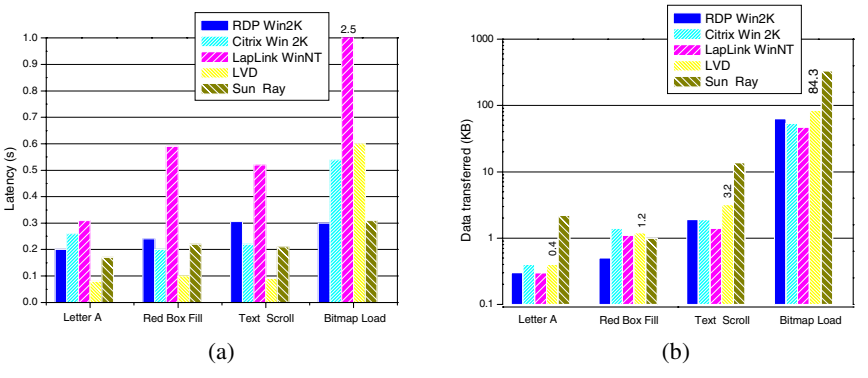


Fig. 4. (a) Latency test time to completion; b) Latency test data transferred

4.2 Web Page Load Benchmark Results

We measure the total time required to display all 109 web pages of the web benchmark and logs complete packet traces using the packet monitor. Fig.5(a) and Fig.5(b) illustrate the total normalized web page download times and total data transferred for each run, respectively.

As shown in Fig.5(a), almost all of the platforms complete the web benchmark in less than 50 seconds at network bandwidths of 4 Mbps or greater, corresponding to an average of less than half a second per page.

Limits. At network bandwidths below 4 Mbps, the performance of the thin client platforms begins to degrade. As shown in Fig.5(a) and Fig.5(b), LVD does not take

longer to complete the web benchmark at lower bandwidth, but instead lose data resulting in missed or incomplete screen updates. On the other hand, RDP and Citrix behave in a similar manner by taking longer to complete the web benchmark as the network bandwidth decreases, but continuing to send the same amount of data even at lower bandwidths.

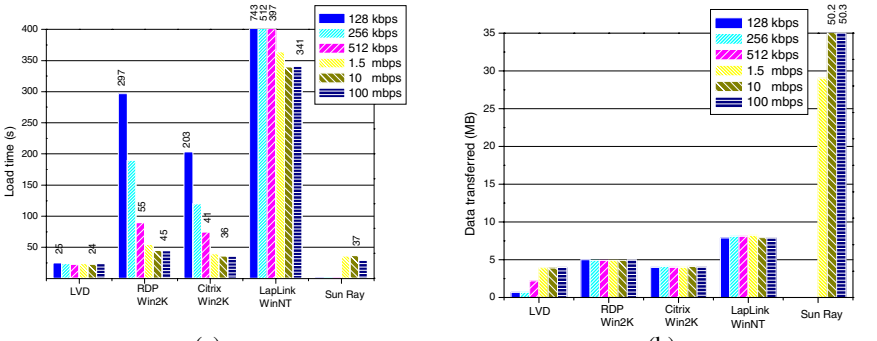


Fig. 5. (a) Web page load time; (b) web page data transferred

In general, 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.

4.3 Flash Benchmark Results

We run the flash animation test on each of the thin client platforms via network, and also run the benchmark on the baseline platforms for comparison. The results are illustrated in Fig.6(a) and Fig.6(b).

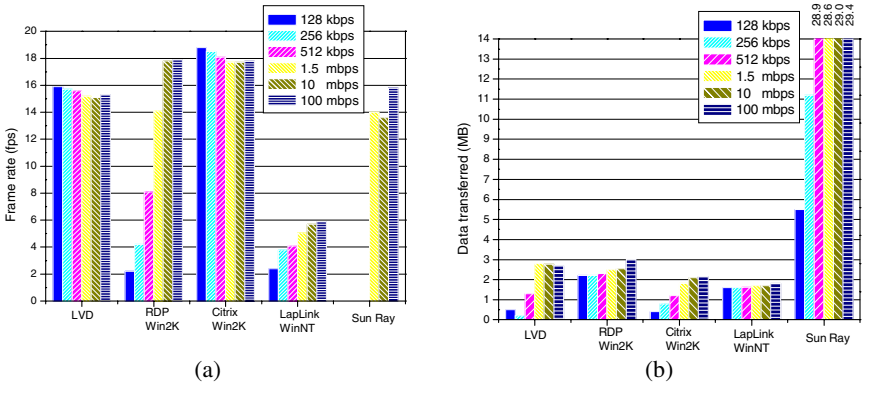


Fig. 6. (a) Flash test frame rate; (b) Flash test data transferred

We find that a frame rate close to 16fps produced good subjective results, with smooth display, no skipped screens, and no tearing or jerky movement. The 100KB animation is downloaded completely to the server prior to playback, so the bandwidth does not affect the frame rate on the baseline.

Limits. 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 bandwidths, 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. In general, LVD system experiences a similar fall-off in performance at bandwidth 128kbps or below, indicating that LAN bandwidths are required to support multimedia applications.

4.4 Power Reduction Results

After analyzing the characteristics of workflow of LVD, we have proposed a new schedule policy called *Magnet* for cluster with VMs to achieve the energy savings [16]. 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.

5 Conclusions

This paper presents LVD, a prototype of system management architecture for managing desktop computers. This paper concentrates on the design issues of a complete system. By combing *wRFB protocol* and *Magnet algorithm* and other techniques of virtualization, LVD provides several novel functions including 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. Besides, it achieves great saving in power consumption. In the future, we will try to optimize the display performance by exploring more intelligent schemes. Also, we will analyze the strategies of the migration of multiple VMs to achieve more energy savings and better load balancing.

References

1. Baratto, R., Kim, L., Nieh, J.: THINC: A Virtual Display Architecture for Thin-Client Computing. In: Proceedings of 20th ACM Symposium on Operating Systems Principles, pp. 277–290. ACM Press, United Kingdom (2005)
2. Bailey, E.: Maximum RPM. SAMS, Indianapolis (1997)
3. Ponder, W.: Sun Desktop Virtualization Solution: Desktop Virtualization Blueprint. Sun Microsystems (2006)
4. Chandra, R., Zeldovich, N., Sapuntzakis, C., Lam, M.S.: The Collective: A Cache-Based System Management Architecture. In: Proceedings of the 2nd Symposium on Networked Systems Design and Implementation, pp. 259–272. ACM Press, Boston (2005)
5. eyeOS project, `http://www.eyeOS.org/`

6. Glide project, http://www.glidedigital.com/
7. Hopkins, M.: The On Site Energy Generation Option. The Data Center Journal, http://datacenterjournal.com/NewsArticle.asp?article_id=66 (2004)
8. Kim, J., Baratto, R.A., Nieh, J.: pTHINC: a thin-client architecture for mobile wireless web. In: Proceedings of the 15th International World Wide Web Conference, pp. 143–152. ACM Press, Edinburgh (2006)
9. Nieh, J., Yang, S.J., Novik, N.: Measuring thin-client performance using slow-motion benchmarking. ACM Transactions on Computer Systems 21(1), 87–115 (2003)
10. Orca desktop, http://www.orcaa.com/
11. Pinheiro, E., Bianchini, R., Carrera, E., Heath, T.: Dynamic Cluster Reconfiguration for Power and Performance. In: Benini, L., Kandemir, M., Ramanujam, J. (eds.) Compilers and Operating Systems for Low Power. Kluwer Academic Publishers, Norwell (2003)
12. Richardson, T., Stafford-Fraser, Q., Wood, K.R., Hopper, A.: Virtual network computing. IEEE Internet Computing 2(1), 33–38 (1998)
13. Satyanaranyanan, M.: Pervasive Personal Computing in an Internet Suspend/Resume System. IEEE Internet Computing 11(2), 16–25 (2007)
14. VMware Virtual Desktop Infrastructure Partners, http://www.vmware.com/partners/alliances/solutions/
15. Yellowdog, http://linux.duke.edu/projects/yum/
16. Hu, L., Jin, H., Liao, X.: Magnet: A Novel Scheduling Policy for Power Reduction in Cluster with Virtual Machines. In: Proceedings of the 9th IEEE/ACM Conference on Cluster 2008. IEEE Press, Tokyo (2008)