Computer Communications 34 (2011) 1539–1548

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

# Separating computation and storage with storage virtualization

Yaoxue Zhang, Yuezhi Zhou [*]

*Key Laboratory of Pervasive Computing, Ministry of Education, Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, People's Republic of China*

ARTICLE INFO

*Article history:*
Available online 3 July 2010

*Keywords:*
Transparent computing
PC management and security
Virtual storage
Virtual disk

ABSTRACT

Recent advances of hardware, software, and networks have made the management and security issues increasingly challenging in PC usage. Due to the tight coupling of hardware and software, each one of the hundreds or thousands of PCs connected in a networked environment has to be managed and administrated individually, leading to a high Total Cost of Ownership (TCO). We argue that a centralized storage of software and data, while distributed computation in clients, i.e., transparent computing, can address these challenges potentially and reduce the complexity with reduced software maintenance time, improved system availability, and enhanced security.

This paper presents a novel approach, named StoreVirt, to realize transparent computing, which separates computation and storage from inside a single physical machine to different machines with a storage virtualization mechanism. With virtualization, all the OSes, applications, and data of clients are centered on the servers and scheduled on demand to run on different clients in a "block-streaming" way. Therefore, due to the central storage of OSes and applications, the installation, maintenance, and management are also centralized, leaving the clients light-weighted. Further, due to timely patching and upgrading, the system security can be improved. Experimental and real-world experiences demonstrate that this approach is efficient and feasible for real usages.

## 1. Introduction

The advent and advance of desktop/personal computers has greatly improved end user productivity and flexibility by enabling a richer set of applications to be installed and executed locally. Now, they have been ubiquitously deployed in enterprise network environments (typically LANs), such as universities, corporations, and governmental organizations. However, the great success of the distributed PC has also brought many challenges for system management and security.

Consider a typical scenario of PC usage in educational classrooms, where tens of computers are connected through a local area network. Given various course requirements, students need to use different OSes such as Linux, Windows, and Solaris, and diverse applications such as office software (e.g., MS Office or Open Office), image/audio/video editors (e.g., Adobe Photoshop, Adobe Premiere, 3dMAX), and program developing tools (e.g., Microsoft C# or GCC). Thus, to satisfy these diverse requirements, various types of OSes and applications have to be installed on each PC. Other PC-based systems in governmental organizations or enterprises are similar to the above usage scenario, as illustrated in Fig. 1(1).

In such typical scenarios, although end users can leverage the computation and storage resources of distributed client computers to achieve flexibility and enhanced productivity, there are mainly two categories of challenges: management and security. With respect to management, there are at least two following challenges:

- *Software consistence:* Since each PC has a local hard disk to store all the required software and data, the tasks of installation, patching, and upgrading have to be carried out on every client to keep a correct, consistent, and up-to-date system state across the entire environment. Automatic management tools such as Marimba can help reduce the manual efforts of administrators by automatically pushing new software images or patches to distributed clients. However, as clients may fail to respond to these tools due to hardware, software, and user errors, or malicious attacks, these tools cannot address the consistency problem fundamentally.
- *Heterogeneous OS and application support:* As described in the above scenario, multiple types or versions of OSes and applications may need to co-exist to support educational requirements, or to support legacy applications and other new requirements. This diversity of software further increases the management complexity. Administrators need accurate knowledge of the correct versions of software to update for each machine. Thus more sophisticated tools are required to push packages automatically in a heterogeneous environment.

* Corresponding author. Tel./fax: +86 10 62782118.
*E-mail addresses:* zyx@moe.edu.cn (Y. Zhang), zhouyz@mail.tsinghua.edu.cn (Y. Zhou).

0140-3664/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2010.06.024

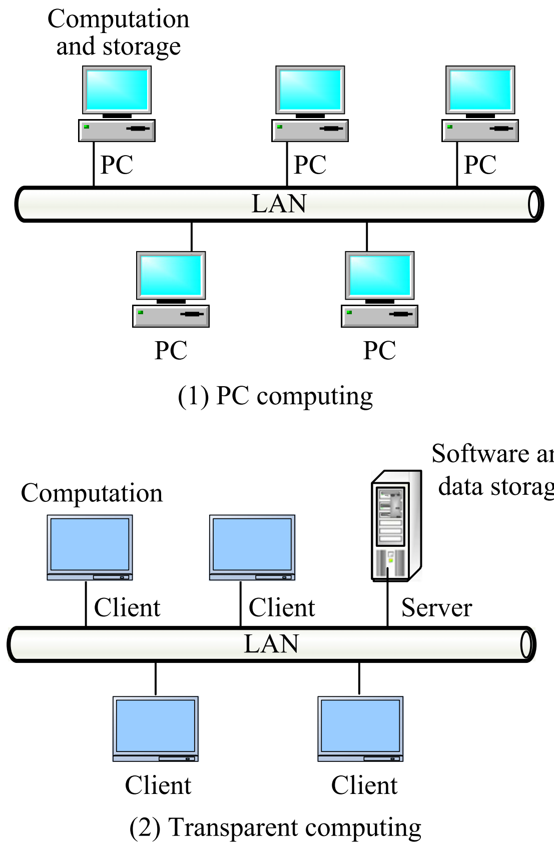

**Fig. 1.** (1) Each PC machine is installed with all needed software and data and executes them locally. (2) Each client machine does not hold any desired software and data locally, which resides on the central server, while being streamed to and executed with the client's local resources.

The next category of challenges concerns the security issue:

- *Malware threat:* The first security risk is associated with malicious attacks such as virus, worms, spyware, and other malware that target the normal functions of individual machines. Once the corresponding client is compromised or damaged, the installed software and data may be lost or corrupted, requiring expensive distributed backup and restoration services.
- *Data protection:* The second security issue is concerned with the data protection. As the data is distributed in the typical scenario, thus the distributed data backup, is time consuming and not reliable due to the same difficulties as that in maintaining software consistency. A more serious data security risk is information leakage and data theft, which is in particular a big threat to the governmental or military organizations. If sensitive data are fetched and cached at local disks, they will be potentially available to the errant end users or intended attackers who have access to the client machines.

Due to these difficulties and challenges in PC management and security, much money and manpower are involved in dealing with these issues. As estimated in a typical scenario, the annual Total Cost of Ownership (TCO) of a PC has been around five times the purchase cost of the PC [1].

To address these challenges, a variety of approaches have been proposed, which can be classified into two categories: distributed client management tools and centralized computing paradigms.

Various client management tools (e.g., BMC BladeLogic Client Automation [2]) have been produced in the past years. These tools use two sets of software: one installed on the server that help administrators to monitor and update the other set of software, often called agents, distributed on all client machines. These agents can report client status to the server and carry out management tasks assigned by the server, such as patching, updating, and scanning. As mentioned before, it is challenging for these management tools to tackle the management and security problems fundamentally. With the constant change in the increasingly distributed and heterogeneous environment, multiple operating system images and hundreds of applications have to be maintained and the patch and system security assured. Moreover, if the connectivity of the client to the server is destroyed in any way, the management tools cannot function anymore, resulting in manual effort or other data or information lost.

To overset the distributed model of PC, new centralized computing paradigms, such as thin client [3–5] in the past decade and new emerging virtual desktop [6,7], try to get PC off the desktop by centralizing both computation and storage on the server and only delivering the keyboard and mouse input and display output between the client and server, which is similar to the mainframe computing a long time ago, but can support desktop operating systems and applications. Due to the centralization of computation and storage, both management and security tasks are also centralized on the server, reducing the overall management efforts in half [7]. However, the large video display data transferred from the server to the client will consume much network bandwidth, it is very limited for these computing paradigms to support multimedia applications, such as video playback and 3D games. Further, the computing power of clients will be underutilized and wasted.

We believe that a new Transparent Computing paradigm with distributed computing, while centralized storage of all software and data, can achieve the best benefits of both the distributed computing model of PC and the above centralized computing paradigm. Without local storage, clients keep no persistent states and execute programs with local resources, while administrators can ensure centralized control of all software and data at a small number of servers, hence effectively addressing various challenges associated with distributed and inconsistent system states.

In this paper, we present a transparent computing system, namely StoreVirt, that can provide desirable features, such as heterogeneous OS support, user transparency, and flexible software and data sharing, by separating computation from storage. StoreVirt decouples software, data, and states from the underlying client hardware. The StoreVirt clients perform all the computing tasks, while all required OSes, applications, and data will be located at centralized servers and streamed to the clients on demand. The key technique is the virtual storage/disk mechanism which simulates the physical block-based storage devices using disk images located on the server and accesses them via network communication.

The remainder of this paper is organized as follows. In Section 2, we introduce the concept of transparent computing as a background and discuss some related work. In Section 3, we provide the detailed ideas and design of StoreVirt. In Section 4, we present the implementation and real experiences of StoreVirt. In Section 5, we study the performance of StoreVirt through several experiments and compare it with other similar approaches. Section 6 discusses possible extensions and optimizations. In Section 7, we conclude this paper.

## 2. Background and related work

### 2.1. Concept of transparent computing

To address the challenges faced by today's personal computers as mentioned above, we proposed a new computing paradigm, termed as transparent computing [8,9]. Its aim is to realize the vision advocated by ubiquitous or pervasive computing [10,11], in which users can demand any computing services via any available device, at any time and any place, with no concerns about these issues such as service installation, maintenance, management, and security.

Specifically, in contrast to PC paradigm, where computation and storage are coupled via inside bus in a single physical machine, as shown in Fig. 1(1), in transparent computing, the computation and storage are separated into different machines via outside networks, as shown in Fig. 1(2). Thus, in transparent computing, users can only focus on their desired computing tasks through easily obtainable devices, while leaving other non-relevant technical machine-specific and management details to system administrators or professional staffs. In transparent computing, all software services, including OSes, middleware, application programs, status, and data, are stored in dedicated servers (transparent servers), while they are delivered in a on-demand and streaming way to terminal devices (transparent clients) and executed mainly with the clients' local resources. In such a centralized storage of services while distributed computation mode, transparent computing paradigm, not only centralizes the maintenance, management, and security-related issues, but leverages the cheap and underutilized resources of different types of client devices.

First, with full control of all software, administrators can enforce OS and application patching and upgrading at the earliest available time. Only a small number of centralized servers have to be managed and maintained, as opposed to tens or hundreds of client machines. The savings of installation and configuration time can ensure software consistency significantly.

Second, centralized storage of software potentially opens up great opportunities for sharing OSes and applications to reduce the complexity of managing heterogeneous software. Administrators can install and support only one copy of each software at the centralized repository. Thus there is no need to keep track of the detailed configuration knowledge of each client machine.

Third, since software patching and upgrading can be performed in a more timely fashion, the time window of clients being vulnerable to malicious attacks will be shorten. Thus virus and worms could have little chance to infect client computers. Information leakage will no longer happen and there is no need to scrub client hard drivers at the end of their usage life. Consider the centralized servers can be better protected, for example, by being locked in more secure locations, it can reduce the end users' opportunities of introducing attacks into the system.

Fourth, without the need of transferring data back and forth between clients and servers, centralized data backup and recovery is faster and more reliable by simply preserving and recovering the snapshots of server repository images.

In all, by ensuring centralized control of software and data, centralized management and security with storage virtualization is a promising solution to address the challenges associated with distributed, inconsistent system states. For an in-depth discussion of the transparent computing, please refer to [8,9].

However, with centralized storage, many issues of managing software at server repositories still need to be addressed, for example, developing tools to update OSes and applications across various disk images or database entries. The centralized and virtualized storage also creates a number of new challenges, for example, software sharing and customization, access control, and in particular performance isolation and guarantee. For some of these issues, we present our solutions to address them in the StoreVirt system that will be described in Section 3.

### 2.2. Related work

There has been extensive research on distributed and pervasive computing platform. Our work is mostly related to systems such as network computers, thin-clients, network file systems, and virtual machine based systems.

At the end of last century, to deal with the management challenge of personal computers, network computers, such as the Java Station by Sun [12], are proposed to replace the personal computers. Such solutions support WWW & Java applications, but do not work with general commodity OSes or other applications such as Microsoft Office. So their usage is very limited in the markets.

Thin client systems have been very popular, by providing a full featured desktop to users with low management costs. Example systems include Microsoft RDP [3], Citrix ICA [4], Sun Ray 1 [5], and VNC [13]. In the thin-client system paradigm, all computation and storage tasks are performed at the central server with a multiple-users enhanced Windows OS (e.g., Windows 2003), while a client works only as a user interface by performing display, and keyboard/mouse input/output functions. Although such systems also achieve centralized management, they greatly increase the server resource requirements with limited scalability. Applications with heavy computing requirements (e.g., multimedia applications) usually cannot be supported by thin-client systems efficiently. Furthermore, user performances are hard to be guaranteed and isolated in Windows-based terminal services.

Network file systems, such as NFS [14], AFS [15], and NAS [16], are popular solutions for sharing data in distributed enterprise environments. Although these systems can be used to share user files flexibly, they generally do not support sharing system files for the reasons described in Section 3.6.

Our idea of centralizing storage while distributing computation is similar to the concept of diskless computers (e.g., [17,18]) in early years. Without local hard disks, a diskless computer usually downloads an OS kernel image from the remote server. It thus cannot support OSes that do not have clear kernel images, e.g., Windows. Neither does it support booting from heterogeneous OSes. Further, virtual disks perceived by StoreVirt users can be flexibly mapped to virtual disk images on the server. Such flexibility allows StoreVirt to share OS and application software across clients to reduce the storage and management overhead, while still isolating personal files for user privacy.

The concept of resource virtualization was introduced long ago and recently has been adopted to address security, flexibility, and user mobility. For example, commercial products such as VMware [19] have extended the concept of virtual machines to support multiple commodity platforms. The disks in these virtual machines are also virtualized, but they reside in local host machine and need to be accessed through the file system of host OS. In contrast, virtual disks in StoreVirt are located in the remote server, with different types of virtual disks for sharing and isolating data among users.

VM-based stateless thick client approaches, e.g., ISR (Internet Suspend/Resume [20]), use virtual machine technology (e.g., VMware) together with a network file system (e.g., Coda [21]) to support user mobility. Each ISR client runs OS and applications in a virtual machine provided by a preinstalled VMware on the host OS. The use of virtual machine can support heterogeneous OSes as well, but it also introduces additional performance overhead due to its virtualization of all hardware resources, including CPU and memory, while in StoreVirt, client OSes are running directly on top of the CPU, memory, and graphics resources of the client machines.

VM-based thin client approaches emerging as virtual desktop solutions in cloud computing, such as Xen Desktop [6] and VMware View [7], create virtual PCs/desktops (i.e., an instance of Windows) on the server or server blade with virtualization technology. The user thus has a complete PC in the data center or cloud, but only consume a fraction of resources of the servers. The virtual desktop can be accessed from any client devices, such as normal

PC, thin client, and mobile devices, through a remote display interface. Compared with traditional thin client systems, the virtual desktop can guarantee and isolate the user performance and security concerns. However, as a type of thin client, it is very hard to support graphics-intensive applications, such as multimedia applications, due to the much network bandwidth needed to transfer video display data. Also, it cannot leverage the cheap and underutilized computation resources of client machines that can be fully beneficial in StoreVirt.

## 3. StoreVirt mechanism and method

To realize the transparent computing described above, we propose a StoreVirt mechanism. The main enabling technology for StoreVirt is the storage virtualization whose primary function is to provide a client machine with virtual disks (Vdisks) instead of local physical hard disks, whose actual contents are located and stored on a remote server. Through storage virtualization, the OS and application programs, the status, and the data stored on local disks are now processed on the central server, fetched to, and executed on the StoreVirt clients via networks, hence the separation of computation and storage is achieved. With the virtualization of storage, the system bus is naturally extended to as a network. The goal of this paper is to present and validate the StoreVirt mechanism.

### 3.1. Assumptions and environments

We made the following four assumptions in the StoreVirt model. First, StoreVirt model operates based on the spatio-temporally extended von Neumann-based computer (transparent client) and uses the remote storage of other computer (transparent server) for storing instructions and data. The transparent client has both CPU and memory to complete the needed computation locally. Here, we assume that the client has enough computing power and volatile memories to carry out the needed computation and the server is a traditional von Neumann architecture-based commodity computer.

Second, it is assumed that the transparent client has no local large storage, such as hard disks. Hard disks are mechanical devices; they consume several tens percent power, generate noise, and are fragile to failures that result in program breakage or data losses. Thus, this assumption can reduce the cost and improve the system security.

Third, we assume there is a strong network connection between the transparent client and server. This means that the network in a transparent computing system can have enough capability and speed for timely transferring instructions and data from servers to clients to satisfy the requirements of computation on clients. This assumption is realistic, since the modern network, especially local area network, has low latency and high bandwidth. Also, our experimental results have shown that a local Ethernet-based network can satisfy these requirements.

Fourth, it is assumed that the transparent computing system is setup and maintained by professional staff and trusted by users. When fetching instructions and data from these servers, users trust in these servers. Because the servers are commonly locked in a dedicated room and maintained by experts, they are more reliable and secure and thus can be trusted.

### 3.2. StoreVirt model

The overview of StoreVirt model is illustrated in Fig. 2. As mentioned above, in a transparent computing system, there are two roles of machines. One is the transparent client, which is a client device with local CPU, memory, and other related devices, but with no large persistent storage of instructions and data, such as hard disks. The other is the transparent server, which is a commodity server computer and holds all the needed OS and application programs to run on the transparent clients, whose main function is to manage the program depositories and serve the clients' requests of instructions and data. Accordingly, also there are two parts of the StoreVirt model, i.e., StoreVirt client and server, which works on the transparent client and server, respectively.

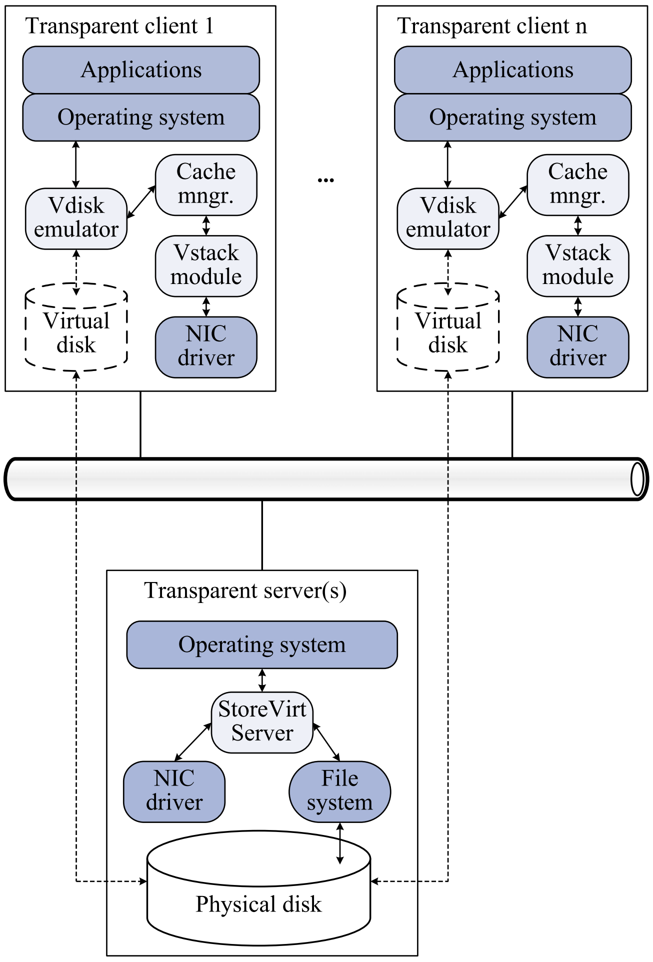

**Fig. 2.** Overview of the StoreVirt model.

As shown in Fig. 2, the StoreVirt client consists of three components, including virtual disk (Vdisk) emulator, Cache manager, and Vstack module. The Vdisk emulator is to simulate a normal hard disk for transparent clients, contents of which are stored on the transparent server's hard disks. We call the emulated hard disk in the transparent client as a virtual disk. To communicate with the transparent server independent of the client OSes, StoreVirt uses a virtual network stack (Vstack) to multiplex the physical network card with other regular network protocol stacks in an OS, for examples, TCP/IP stack. In addition, to alleviate the effects of latency of network delivery, there is a Cache manager in the StoreVirt client. It caches the written results by users for deferring them to be synchronized with the servers, or for serving the succeeding reading.

The StoreVirt server's function is simple to be described. It just listens, receives, and queues the requests from the StoreVirt clients via the server OS, handles them and then returns the corresponding results to the StoreVirt clients. For the virtual disk reading or writing, it first consults the relevant database, and then decides where and how to handle the requests with the hard disks of transparent server via the file system of the server OS. Next, we will discuss the relevant components and techniques in more details.

## 3.3. Virtual disk and access method

The core idea of StoreVirt is the notion of virtual disks. From the perspective of a user or application, there is no difference in accessing data from Vdisks or local hard disks. However, Vdisk is just a virtualized logical disk device in transparent clients; its actual contents reside at the transparent server(s) and are fetched to the clients for execution on demand. To use the virtual disks whose contents are located on the transparent server, there are two set of access protocols needed in the StoreVirt model. The first set of protocols is concerned about how to access the contents of Vdisk before a client OS starts, termed as MRBP (Multi-OS Remote Booting Protocol). The second set is about how to access the Vdisk when the client OS runs after starting, termed as NSAP (Network Service Access Protocol).

In order to start an OS from the transparent server, the transparent client must read the OS instructions from the Vdisk images, instead of from hard disks as that in traditional computers. First, the transparent client needs to establish a network link to the transparent server and then a virtual disk (through emulating the access interface of traditional hard disks for the OS bootstrap program). Second, the transparent client needs to discover the supported OSes, and then display the OS lists to end users for their selections. Third, after the user's selection, the Vdisk on the transparent client will be mapped to the dedicated Vdisk image that holds the corresponding OS instructions. Finally, the OS bootstrap program can read the OS instructions from the Vdisk as from a normal hard disk. To implement this set of protocols, it is needed to extend the traditional BIOS function [22].

However, after the client OS is booted up, the Vdisk access interface established through extending the BIOS function is to be disrupted in modern PCs (e.g., x86-based machines), due to the different memory access method before and after the booting up of common OSes. Therefore, to continue providing instructions and data after the OS initiates, the OS-specific StoreVirt client is loaded and run as kernel services for streaming the OS instructions and data continually.

As mentioned above, a virtual disk request issued by the above OS or file system will be intercepted by the StoreVirt client, i.e., the Vdisk driver, and then be changed into one or more NSAP packets that are sent to and responded from the remote transparent server. Thus, each given virtual disk request received from the file system, the OS-specific Vdisk emulator will compose one or more remote disk requests in the format of NSAP to be sent to the server.

The first function of NSAP is to establish a unique connection between the Vdisk in transparent client and the Vdisk image in server side. Consequently, each client can maintain two request queues: one for the virtual disk requests received from the file system, and the other for the remote disk requests to be sent to the server. The second function of NSAP is to deliver the instructions and data including OS codes from transparent servers to clients or the computation results from transparent clients to servers. These transmissions occur when interruptions or I/O requisitions are made in the transparent client.

## 3.4. Flexible mapping from virtual to physical

As mentioned in the above section, it is first needed to map a logical virtual disk in the transparent client to the physical hard disk (via Vdisk image) at the transparent server before the virtual disk can be accessed by users. In SotreVirt, there are two different levels of mapping in establishing such mapping relationships. The first level is to map the users' virtual disk to its authorized Vdisk image on the transparent server; the second level is to map the logical blocks of Vdisk images to the physical blocks of hard disks of transparent servers.

This corresponding relationship between the user's Vdisk and Vdisk image can be expressed as a mathematical mapping:$f{:}VD \rightarrow VDI$, in which *VD* means the aggregate of Vdisks and *VDI* the aggregate of Vdisk images. Under the control of StoreVirt, though the driver letters of the Vdisks accessed by different users of transparent clients are the same, after being transformed by the StoreVirt, they can be mapped to the same or different Vdisk images located on the transparent server.

Actually, there are two types of mapping from the user's Vdisks to Vdisk images: one-to-one mapping and many-to-one mapping, as shown in Fig. 3(1) and (2), respectively. The one-to-one mapping is mainly used for mapping to the Vdisk image that stores private user information or data, while the many-to-one mapping is for mapping to the Vdisk image that is shared by multiple users. In fact, other corresponding relationships also exist from *VD* to *VDI*, such as one-to-many mapping. Although this is not a mapping relationship in mathematical term, it can be used to develop the concurrent operation capability for the Vdisk.

At the second level, the occupation of Vdisk image resources can be described with different storage granularity, such as "block" of one data block or "chunk" of multiple blocks. The organization of its resources can be in the form of a table or tree. Different resource granularity and different organization forms may lead to different searching performance, which will in turn affect the overall read/write performance of Vdisk.

Fig. 4 shows a table-based resource mapping of Vdisk images in a block granularity. In the resource table, each horizontal row demonstrates the mapping of a virtual block address (VBA) of virtual

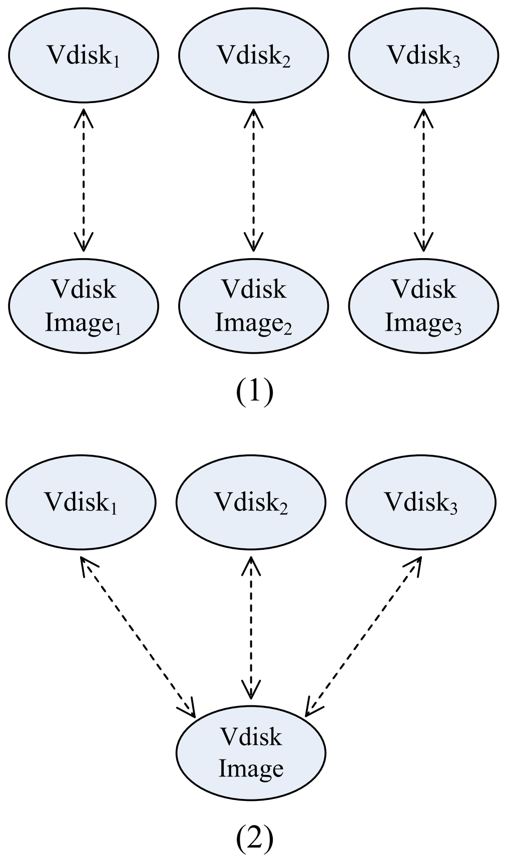

**Fig. 3.** Different types of mapping from Vdisks to Vdisk images.

| | | |
|---|---|---|
| $VBA_1$ | $PSD_1$ | $PBA_i$ |
| $VBA_2$ | $PSD_1$ | $PBA_j$ |
| ... | ... | ... |
| $VBA_{n-1}$ | $PSD_2$ | $PBA_k$ |
| $VBA_n$ | $PSD_2$ | $PBA_m$ |

**Fig. 4.** Resources of Vdisk image organized in a table form.

disk block to its physical block address (PBA), including the physical storage device (PSD) where the corresponding physical block locates. The number of items in the table is determined by the number of logical blocks in a Vdisk image. We can see from this figure that the Vdisk image shown here occupies the resources across two physical devices, namely, PSD1 and PSD2.

Obviously, the organization of Vdisk images' resources in a table form is quite simple and easy to understand, but it may be less efficient. In order to enhance the searching efficiency, the resources of Vdisk images can be organized in the form of a tree, such as a radix tree implemented in parallax [23]. In a resource tree, each leaf node represents the PSD where each real physical block corresponding to the VBA is located, as well as the specific PBA of the logical block.

When an end user logins, the StoreVirt client will ask for his name and password. After the user logs in and selects the desired OS, the StoreVirt server will then establish his virtual disks to the virtual disk images and related physical storage devices. This process is illustrated in Algorithm 1.

**Algorithm 1.** Algorithm for establishing users' virtual disk mappings.

```
Input:
        The end user name: $N_u$;
        The end user password: $P_u$;
Output: Virtual disk mappings: $M_i$;
1:      The end user input $N_u$ and $P_u$; {client}
2:      Authenticate the end user with $N_u$ and $P_u$; {server}
3:      if success then
4:        The end user select the desired OS; {client}
5:        Get the id of the selected OS; {server}
6:        Look up for the set of $VD_n$ has been assigned;
7:        Initialize $M_i$;
8:        for $j = 1$ to $n$ do
9:          Map $VD_j$ to $VDI_j$;
10:         Map $VDI_j$ to $PSD_j$;
11:         Fill $M_j$;
12:       end for
13:       return $M_i$;
14:     else
15:       return error;
16:     end if
```

## 3.5. Software and data separation

As described in Section 3.4, there are two levels of mapping from the virtual disks on the StoreVirt client to the physical devices on the StoreVirt server. The first level mapping mechanism from Vdisks to Vdisk images provides a very flexible approach for software and data sharing by mapping different users' Vdisks to a same Vdisk image that contains the software or data to be shared. The second level of mapping from logical blocks of Vdisk images to the physical blocks of real hard disks also provides a mechanism for flexible resource management of Vdisk images and advanced features for reading or writing, for example, providing concurrent reading/writing operations from/to different hard disks through mapping different ranges of virtual blocks of a Vdisk to different physical hard disks. We will illustrate the advantage of the first level mapping by separating software and data below.

To facilitate effective management of centralized Vdisk images and support heterogeneous OSes and applications with reduced complexity, StoreVirt classifies Vdisks in the transparent client into four different categories to enable sharing and isolation, based on the flexible mapping mechanism described in Section 3.4.

We separates software from data in StoreVirt based on the observation that, many users will use the same OS and application software and thus they can be shared among users, while data are often user-specific and cannot be shared directly. There are mainly four categories of Vdisks:

*System Vdisk*: It is mapped in a many-to-one mode to the "golden image" that stores the OS and application programs. The corresponding system Vdisk images are created by administrators and shared by all transparent clients. They can only be modified by the administrators. More details will be presented in Section 3.6.

*Shadow Vdisk*: It is a user-specific Copy-On-Write (COW) Vdisk for a system Vdisk to enable write access to the System Vdisk contents. Each Shadow Vdisk is mapped to a user-specific Vdisk image in a one-to-one mode. The COW semantics can be supported at the granularity of files through a file redirector, which is a file system level software agent as mentioned before. When a user needs to modify a file on the System Vdisk, a COW copy of the file will be created on the shadow Vdisk for any subsequent access. The use of Shadow Vdisks is transparent to end users.

*Profile Vdisk*: Each client also has a Profile Vdisk to store user-specific persistent data such as customized user settings for OS and applications. Similar to Shadow Vdisks, the existence of Profile Vdisks is also transparent to end users.

*User Vdisk*: Each client has one or more User Vdisks that are used to store the private user data. Each Profile or User Vdisk will be mapped to a user-specific Vdisk image.

It is the classification of Vdisks that greatly simplifies software management tasks, especially for system recovery. For example, if a transparent client is corrupted by accidental errors, software bugs, or malicious attacks such as viruses, worms, and spyware, system administrators can quickly clear the COW Vdisk contents to return a clean system image for end users.

## 3.6. Software sharing, isolation, and recovery

To enable software sharing, StoreVirt maintains a "golden image" of a clean system that contains the desired OS and a common set of applications. This "golden image" is thus immutable and can be shared by all transparent clients. However, some applications must write to the directories where they reside to function properly, e.g., creating temporary files. To support such applications, StoreVirt adopts a COW approach by having a user-specific COW Vdisk (i.e., shadow Vdisk) image corresponding to the "golden image" for each client user.

The COW operations can be implemented through a file system redirector at the file level on the transparent client. It filters the file written operations and redirects them to the COW Vdisk or images. It also needs to carry out the reading operations by combining the contents of the original and COW Vdisk.

Specifically, the file redirector translates the file access requests on user-perceived Vdisks into those on server-perceived Vdisks by intercepting all the file system calls. If the file to be accessed locates on the user-perceived private Vdisks (i.e., shadow Vdisk, profile Vdisk, and user Vdisk), the redirector simply maps the request to the same file on the server-perceived private Vdisk. If the file to be accessed is on the system Vdisk, the file redirector will redirect the request to the shadow Vdisk in the two following cases: (1) a read request to a system file that already has a customized copy on the shadow Vdisk, and (2) a write request to a system file (in this case, a copy of the file will be first created on the shadow Vdisk before written). Otherwise, the file redirector will redirect the request to the system Vdisk. The file redirector therefore supports dynamic redirection of system files for enabling file system level COW semantics. This software agent will be loaded from the system Vdisk as part of the underlying file system. Algorithm 2 shows the process for the file redirector to handle an open file system call.

**Algorithm 2.** Algorithm for redirecting users' open file request mappings.

```
Input:
        The file name to be opened: $F$;
        The file operation mode: $mode$;
Output: The pointer to the file opened: $Fp$;
1:      if $F$ locates on user-perceived private_Vdisk then
2:          $Fp$ = open_file ($F$, $mode$, "private_Vdisk");
3:          return $Fp$;
4:      end if
5:      if $F$ has a copy $F'$ on the shadow_Vdisk then
6:          $Fp$ = open_file ($F'$, $mode$, "shadow_Vdisk");
7:      else if $mode = read$ then
8:          $Fp$ = open_file ($F$, $mode$, "system_Vdisk");
            {no customized copy of $F$ exists, open directly}
9:      else if $mode = write$ then
10:         $Fp$ = open_file ($F'$, $mode$, "shadow_Vdisk");
            {need to customize $F$, open a new file in the
            shadow_Vdisk}
11:     end if
12:     return $Fp$
```

Of course, the COW operation can also be implemented at the block level. The block redirector can be carried out by the transparent server and transparent to the client and thus can alleviate the dependence on the client's computation and the network delivery.

With the above COW operation of system Vdisk, not only can the system Vdisk and hence the software be shared among different end users, but also their performance and experiences are isolated and guaranteed. Further, if an end user gets attacked or makes a mistake, it is very easy and quick to recover and return him a consistent and usable clean system state just by cleaning his COW Vdisk content of the corresponding system Vdisk.

## 3.7. Virtual network stack

As discussed in Section 3.2, in StoreVirt model, the Vdisk emulator needs to submit read/write requests to the remote StoreVirt server for further handling. Thus, the StoreVirt client needs to share the underlying physical network with other regular OS components or applications. To implement this, we adopts a virtual network stack technique to multiplex and share the network between the StoreVirt and the normal TCP/IP stack in a client OS, as shown in Fig. 5.

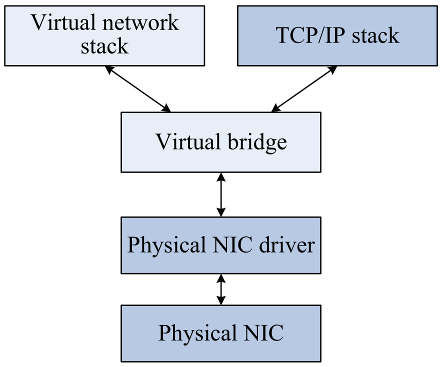

**Fig. 5.** Virtual network stack in StoreVirt client.

The virtual network stack implements the high and low-level network protocols needed to communicate with the StoreVirt server, specifically, NSAP and others. It receives the Vdisk access requests from the Vdisk emulator, interprets and encapsulates them in well-formed packets, and then delivers them to the virtual bridge.

The virtual bridge is the most important component to multiplex the underlying physical network. It binds to the network driver to receive every packet that the physical NIC has received. It then decides to route the packet to one of the above network stacks. This decision can be made through using different IP address between the virtual network and the TCP/IP stack or through different UDP/TCP ports if needed.

It should be noted that, with virtual network stack, other network functions can be implemented, such as firewall, monitoring, and auditing to improve the security of StoreVirt. But these are beyond the discussion of this paper.

## 3.8. Cache and buffer management

Due to the extending of local I/O operations to networked I/O operations, the Vdisk I/O access path involves travelling from the transparent client, through network, and then to the transparent server, thus, the access performance of Vdisk will be affected by conditions of the network and transparent server. To lower the dependence on network and reduce the number of requests for Vdisk I/O operations sent to the transparent server and thus reduce the corresponding response time, the cache manager buffers part of the response data. Similarly, to further reduce the operations to Vdisk images, a Vdisk image cache can also be established in the transparent server to buffer the Vdisk image status and relevant data.

The first function of the cache manager of StoreVirt client is to cache the request or response data from the client OS or the remote StoreVirt server, respectively, and thus reducing the I/O response time.

In case, the requests sent from client OS are for data reading, the Vdisk emulator will first send the request to the cache manager for searching the local cache for the data requested. If it is found there, the cache manager will operate on the local cache and return the results to the Vdisk driver. Otherwise, the cache manager will send the requests to the transparent server. Upon receipt of the results returned by the transparent server, the cache manager will parse the results and return them to the Vdisk emulator in a form that can be understood by the latter. At the same time it will send the same results to the local cache so that, relevant contents in the latter may be updated. This process can be illustrated in Algorithm 3.

**Algorithm 3.** Algorithm for reading blocks of Vdisk with local cache.

```
Input:
        Vdisk Id:: $id$;
        Block offset to read: $st$;
        Block length to read: $le$;
Output: The data to be read: $D[0..n-1]$;
1:      initialize $D[0..n-1]$;
2:      for $i = 0$; $i < le$; $i++$ do
3:        Look up for $Block_{id}^{st+i}$ in local cache;
4:        if find in cache then
5:          $D[i] = Block_{id}^{st+i}$;
6:        else
7:          Fetch Block from the remote server;
8:          Prefetch related blocks from the remote server;
9:          Update the local cache;
10:       end if
11:     end for
12:     return $D[0..n-1]$;
```

In case of requests for data writing, also the Vdisk emulator will first send the written requests to the cache manager. The cache manager will update the local cache with the written data and then send the written request to the StoreVirt server through NSAP. After getting the successful response from the transparent server, it delivers it to the Vdisk emulator to indicate that the written operation is completed.

In addition to caching the written results for succeeding reading operations, the cache manager can also prefetch some instructions or data in advance, thus reducing the response time of reading data sharply. However, this prefetching may bring waste of network bandwidth if the prefetched data is not needed in short time.

## 4. Implementation and deployment experiences

We have implemented a prototype of StoreVirt that supports both Windows 2000/XP and RedFlag Linux Desktop 6.0 (Linux kernel 2.6) [24]. Our implementation of MRBP is based on the Intel PXE protocol [25] for sending boot requests. The implementation of NSAP is based on the UDP [26]. Because, device drivers are platform dependent, we implemented two different Vdisk emulators, customized for Windows and Linux, respectively. The Vdisk emulator is implemented as a SCSI port device driver at the block level. The cache manager is integrated within the Vdisk driver. We have implemented the Vstack module as a filter intermediate driver, which binds to the network miniport driver.

The implementations are in C++. Since Windows 2000/XP is a modified microkernel, we modified the corresponding Windows Registry files for the OS to load these added drivers. Thus there is no need to change or recompile the kernel. However, since Linux is a monolithic kernel, we compiled the Vdisk and Vstack drivers into the kernel by modifying the related kernel source code before recompilation.

The technology of StoreVirt has been transferred to several companies for industrial products. These systems based on StoreVirt have been deployed across many universities, enterprises, and other organizations for daily usages.

Take the typical deployment for an interactive English learning class as an example, the transparent clients are Intel Atom 1.60 GHz machines, each with 512 MB DDR2 666 MHz RAM and 100 Mbps onboard network card. The server is a Dell PowerEdge 840 machine with an Intel Xeon Dual Core 1.6 GHz CPU, 2 GB DDR2 333 MHz RAM, a 1 Gbps network card, and a 160 GB

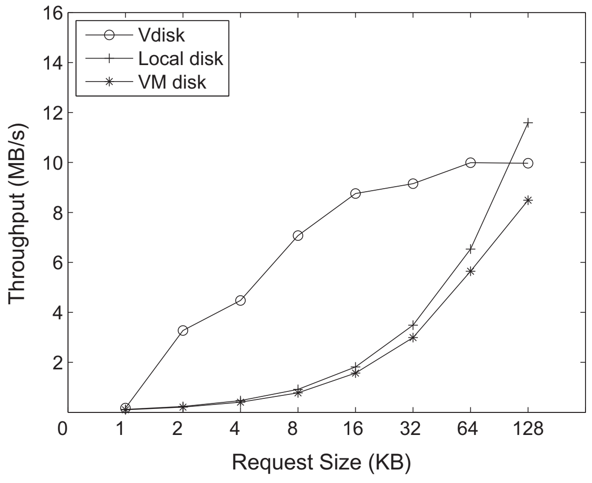

**Fig. 6.** Random read throughput.

Samsung He160hj 7200 rpm SATA hard disk. The clients and the server are connected by an Ethernet switch with 48 100 Mbps interfaces (used for clients) and 2 1 Gbps interfaces (used for the server). The server OS is Windows 2003 Standard (SP2). The transparent clients use Windows XP Professional (SP3).

These real deployed systems have been observed to run stable most of time and have achieved at least the following benefits:

*Reduced system maintenance time*: Previously, administrators spent on average one or at least a half of a day in a week to clear every machine regularly even with the help of automatic tools to fix problems caused by user faults or malicious attacks. Using StoreVirt, the system cleaning and upgrading time is reduced to 30 min per week, due to both the reduced number of malicious attacks and the centralized operations.

*Improved availability and usability*: Before using StoreVirt, the 4–8 hour system maintenance took place every week. No class can be arranged to use the classroom during this maintenance day. After deploying StoreVirt, the classroom can be used everyday without weekly service interruption.

*Improved security*: After deploying the transparent computing system, there have been less reported virus or worm attacks than before. Even with errors, the transparent system resumed operations quickly.

## 5. Experimental evaluation

In this section, we will evaluate the StoreVirt in Windows XP with several experiments. We evaluated the storage virtualization performance in disk and file system levels and compared it with other popular approaches.

### 5.1. Experiment setup

In our experiments, we used the same hardware configurations as the above mentioned real deployment but with a more powerful server of Dell PowerEdge 1900 machine. It is configured with an Intel Xeon Quad Core 1.6 GHz CPU, 4 GB Dual DDR2 666 MHz RAM, one 160 GB Hitachi 15,000 rpm SATA hard disk, and a 1 Gbps onboard network card. We also compared the StoreVirt performance with a regular PC, which has the same hardware configurations but with an additional local hard disk (80 GB Seagate Barracuda 7200 rpm SATA), and a virtual machine that emulates the VM-based like approaches, which is virtualized as with 512 MB memory and a static 8 GB SCSI hard disk using VMWare Workstation 6.5 hosted by Windows XP Professional (SP3) with NTFS V3.1 file

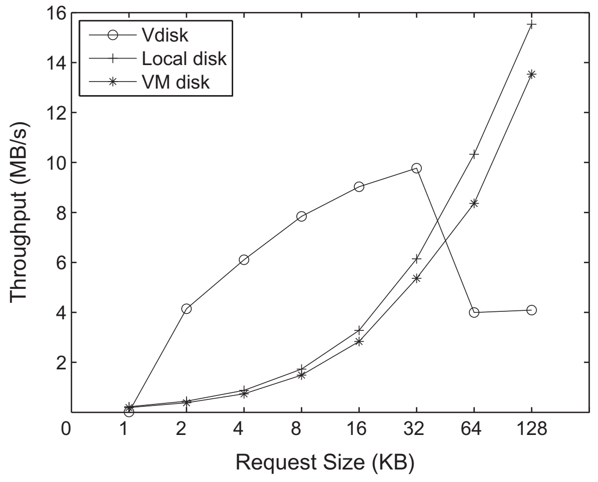

**Fig. 7.** Random write throughput.

system on the same regular PC hardware (but with 1 GB physical memory, a half of it is used by the Host OS). The server OS of StoreVirt is Windows 2003 Enterprise (SP2) running a NTFS v3.1 file system. All the StoreVirt clients, the regular PC, and the virtual machine use Windows XP Professional (SP3) with NTFS v3.1 file system.

### 5.2. Vdisk access performance

We first evaluate the Vdisk access performance in terms of throughput in a single transparent client setup. The experiment is carried out by using the Iometer tool [27] to submit random disk access requests of different size to the machine, with the filesystem caches disabled.

The results shown in Figs. 6 and 7 are the average of five trials. Because the standard deviations are small (less than 10%), they are not plotted here. As mentioned above, we also compare the throughput with that of a regular PC's local hard disk and virtual machine's virtualized hard disk. As seen from Fig. 6, for read access, the Vdisk throughput in StoreVirt increases with the request size and is higher than the local disk, but decreasing when the request size is larger than 64 KB, which is the maximum size delivered by one NSAP service. When the request size is larger, the network communication dominates the latency, for that a large request size will cause several service requests. At the same time, because the response in StoreVirt can be satisfied with the server's memory cache, the Vdisk performance is higher than local disk when the request size is small. The write access shown in Fig. 7 is similar to that of the read access, but decreasing at the size of 32 KB, which is also the maximum size delivered by NSAP writing service. Note that the throughput of write access is bigger than read access in all cases, this may be due to the embedded hardware cache of hard disk or the server cache of StoreVirt server returns success without carrying the real disk operations.

### 5.3. File system performance

In this section, we evaluate the overall file system performance of a StoreVirt client, using a modified Andrew benchmark [28]. We compared the performance against the file system performance of a regular PC with a local disk, the CIFS (Common Internet File System) [29], and the virtual machine in the VM-based approaches. For CIFS, we used the same PC and the same StoreVirt server hardware configuration. In our benchmark, we used the Windows Apache 2.0.53 source tree. This source tree has 39.3 MB data before compilation, and 42 MB data after compilation. Table 1 shows the average performance over 5 runs. For each run, we rebooted both the client and server to clean various caches.

We observe that StoreVirt achieves a little better performance than regular PC in the "mkdir", "scan dir", and "cat" phases, which is aligned with the better Vdisk accessing performance. However, the StoreVirt performance in "cp" and "make" phases is a little worse than the regular PC. In StoreVirt, these phases may require accessing continuous blocks of large size, resulting in a large number of remote disk requests and thus involving much CPU and communication overhead. Even though, the StoreVirt performance is much better than the CIFS and virtual machine.

Our file system performance evaluation shows that, using a more powerful server and fast network access, StoreVirt can achieve comparable file access performance to a regular PC, and can potentially perform better than other remote file system solutions and VM-based approaches.

## 6. Discussion

In this section, we discuss possible extensions and optimizations to StoreVirt for enhancing the system performance, robustness, and security.

The use of explicit caches at both client and server side can potentially enhance performance significantly. At the client side, Vdisk driver cache can reduce the number of network communications. This latency is the current performance bottleneck. At the server side, we can exploit the locality of read requests across different clients using a Vdisk image cache and optimize write request using application level write optimization schemes (e.g., lazy write). The Vdisk access protocol can be further optimized to improve performance. In our current implementation, remote disk requests are sent in sequential order. As future work, we can enhance the disk access latency and throughput by sending multiple remote disk requests concurrently.

So far, we have not discussed how to generate or update the contents of Vdisk images at StoreVirt server. In our current implementation, the contents of Vdisk images exactly simulate those of a hard disk, with several special, initial blocks proceeding the data blocks. These special blocks are hardware dependent, containing disk parameters such as disk capacity, cylinders, heads, and sectors. Thus Vdisk image files, in particular the system disk image, can work for only homogeneous hardware machines. Supporting machines with heterogeneous hardware is our ongoing work.

**Table 1**
Average time spent at various phases of the modified Andrew benchmark with Apache Windows source tree. Times are reported in seconds and standard deviations are given in parentheses.

| Phase | StoreVirt | | PC | | CIFS | | Virtual machine | |
|---|---|---|---|---|---|---|---|---|
| mkdir | 0.77 | (0.07) | 2.44 | (0.12) | 2.92 | (0.59) | 1.39 | (0.09) |
| cp | 34.68 | (0.66) | 26.64 | (1.85) | 77.46 | (0.52) | 28.98 | (1.02) |
| scan dir | 187.26 | (3.25) | 205.95 | (1.02) | 234.59 | (1.48) | 771.22 | (2.85) |
| cat | 365.83 | (1.08) | 381.58 | (2.53) | 455.14 | (0.95) | 1463.66 | (12.82) |
| make | 323.60 | (0.82) | 254.79 | (1.98) | 403.39 | (0.52) | 709.40 | (4.76) |
| Total | 912.14 | (4.02) | 871.40 | (2.97) | 1173.50 | (1.69) | 2974.65 | (13.72) |

Our current prototype does not implement automatic server fail over. When server crashes or the system disk image needs to be updated, the entire system needs to be manually shutdown and reboot. As future work, we plan to use server replication mechanisms (e.g., [30,31]) to handle these scenarios, where clients can switch to an identical backup server when required.

SotreVirt server needs to prevent and detect unauthorized access of information. The current use of user name and password authentication can only protect against malicious attackers that spoof client users, but can not protect with the content of each NSAP packet content. A network-level encryption mechanism (e.g., IPsec [32]]) might help mitigate the possibility of such attacks. We can also augment the current system using various encryption based approaches to protect the privacy of disk data access.

Our current prototype only support Ethernet-based LAN network. As the mobile network and even 3 G network emerge, it is more benefical to support them in transparent computing. Due to the network bandwidth is limited and vibrated, we need to explore its usage in transparent computing systems. This is one of our ongoing work.

In StoreVirt, we do not need to change the kernel of Windows or Linux OS, thus it does not touch the working mechanism of the supported OS, such as task/process, memory, and other I/O management. This compatibility is critical for successful adoption in real markets. However, as a future work, we may need to explore the impact of the storage virtualization on the normal functions of OS.

## 7. Conclusions

We have introduced a novel computing paradigm, transparent computing, which tries to solve the challenges faced by current computing systems based on a storage virtualization mechanism by separating the tight-coupled computation and storage in the past PC. To realize transparent computing system, we have developed StoreVirt, which stores all data and software on virtual disks that correspond to disk images located on a central server with a storage virtualization mechanism.

We have given real usage experiences and carried out several experiments to evaluate StoreVirt. We show that with a powerful server, StoreVirt can achieve comparable or even better disk and filesystem performance than regular PCs with local hard disks, networked file system and VM-based approaches.

Future work includes further optimizing performance, enhancing the system security, and supporting more types of computing devices, such as smart phones and digital appliances, and more types of networks, such as WiFi and 3G.

## Acknowledgments

This work is supported by the National High Technology Research and Development Program of China (2009AA01Z151). The authors would like to thank the students, especially Hao Liu, Li Wei, etc., for their contributions in the development and experiments. The authors thank other people for their helpful comments and fruitful discussions.

## References

[1] R. Bloor, Why the desktop is broken, 2007. <http://havemacwillblog.com/2007/11/21/why-the-desktop-is-broken/>.
[2] Bmc bladelogic client automation, 2009. <http://www.bmc.com/products/offering/configuration-automation-for-clients.html>.
[3] B. Cumberland, G. Carius, A. Muir, Microsoft Windows NT Server 4.0, Terminal Server Edition: Technical Reference, Microsoft Press, 1999.
[4] Boca Research, Inc. Citrix ICA Technology Brief, Technical White Paper, Boca Raton, 1999.
[5] Sun Ray Overview, White Paper, Version 2, 2004. <http://www.sun.com/sunray/whitepapers.html>.
[6] Citrix XenDesktop 4, 2009. <http://www.citrix.com/English/ps2/products/product.asp?contentID=163057>.
[7] VMware, Inc. VMware View 3, Brochure, 2009. <http://http://www.vmware.com/files/pdf/view_brochure.pdf>.
[8] Y.X. Zhang, Y.Z. Zhou, Transparent computing: a new paradigm for pervasive computing, in: Proceedings of the Third International Conference on Ubiquitous Intelligence and Computing, 2006.
[9] Y.Z. Zhou, Y.X. Zhang, Transparent Computing: Concepts, Architecture, and Implementation, Cengage Learning Asia Pte Ltd., 2009.
[10] M. Weiser, The computer for the twenty-first century, Scientific American 265 (3) (1991) 94–104.
[11] D. Saha, A. Mukherjee, Pervasive computing: a paradigm for the 21st century, IEEE Computer 36 (3) (2003) 25–31.
[12] R.G. Herrtwich, T. Kappner, Network computers – ubiquitous computing or dumb multimedia? in: Proceedings of the Third International Symposium on Autonomous Decentralized Systems, 1997.
[13] T. Richardson, Q. Stafford-Fraser, K.R. Wood, A. Hopper, Virtual network computing, IEEE Internet Computing 2 (1) (1998) 33–38.
[14] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, B. Lyon, Design and implementation of the sun network filesystem, in: USENIX Association Conference Proceedings, 1985.
[15] J.H. Howard, M.L. Kazar, S.G. Menees, Scale and performance in a distributed file system, ACM Transactions on Computer Systems 6 (1) (1988) 51–81.
[16] G.A. Gibson, R.Y. Meter, Network attached storage architecture, Communications of the ACM 43 (11) (2000) 37–45.
[17] D.R. Cheriton, W. Zwaenepoel, The distributed V kernel and its performance for diskless workstations, in: Proceedings of the Ninth ACM Symposium on Operating Systems Principles, 1983.
[18] B. Croft, J. Gilmore, Bootstrap Protocol (BOOTP) RFC 951 (1985).
[19] J. Sugerman, G. Venkitachalam, B.-H. Lim, Virtualizing i/o devices on vmware workstations hosted virtual machine monitor, in: Proceedings of the 2001 USENIX Annual Technical Conference, 2001, pp. 1–14.
[20] M. Satyanarayanan, B. Gilbert, M. Toups, N. Tolia, D.R. O'Hallaron, A.S. abd A. Wolbach, J. Harkes, A. Perrig, D.J. Farber, M.A. Kozuch, C.J. Helfrich, P. Nath, H.A. Lagar-Cavilla, Pervasive personal computing in an internet suspend/resume system, IEEE Internet Computing 11 (2) (2007) 16–25.
[21] M. Satyanarayanan, The evolution of coda, ACM Transactions on Computer Systems 20 (2) (2002) 85–124.
[22] Y. Zhang, Y. Zhou, 4VP$^+$: a novel meta os approach for streaming programs in ubiquitous computing, in: Proceedings from AINA07: The IEEE 21st International Conference on Advanced Information Networking and Applications, 2007, pp. 394–403.
[23] D.T. Meyer, G. Aggarwal, B. Cully, G. Lefebvre, M.J. Feeley, N.C. Hutchinson, A. Warfield, Parallax: virtual disks for virtual machines, SIGOPS Operating System Review 42 (4) (2008) 41–54.
[24] RedFlag Linux. <http://www.redflag-linux.com/en/index.php>.
[25] Intel Corporation. Preboot Execution Environment (PXE) Specification, Version 2.1, 1999.
[26] J. Postel, User Datagram Protocol RFC 768 (1980).
[27] Iometer. <http://www.iometer.org>.
[28] L.P. Cox, C.D. Murray, B.D. Noble, Pastiche: making backup cheap and easy, in: Proceedings of the Fifth USENIX Symposium on OSDI, 2002.
[29] P. Leach, D. Perr, CIFS: A common internet file system, Microsoft Interactive Developer (1996).
[30] R. Guerraoui, A. Schiper, Software-based replication for fault tolerance, IEEE Computer 30 (4) (1997) 38–74.
[31] A. Helal, A. Heddaya, B. Bhar, Replication Techniques in Distributed Systems, Kluwer Academic Publishers, 1996.
[32] S. Kent, R. Atkinson, Security Architecture for the Internet Protocol RFC 2401 (1998).