TransCom: A Virtual Disk-Based Cloud Computing Platform for Heterogeneous Services

Yuezhi Zhou, *Member, IEEE,* Yaoxue Zhang, Yinglian Xie, Hui Zhang, Laurence T. Yang, *Member, IEEE,* and Geyong Min, *Member, IEEE*

***Abstract*—This paper presents the design, implementation, and evaluation of TransCom, a virtual disk (Vdisk) based cloud computing platform that supports heterogeneous services of operating systems (OSes) and their applications in enterprise environments. In TransCom, clients store all data and software, including OS and application software, on Vdisks that correspond to disk images located on centralized servers, while computing tasks are carried out by the clients. Users can choose to boot any client for using the desired OS, including Windows, and access software and data services from Vdisks as usual without consideration of any other tasks, such as installation, maintenance, and management. By centralizing storage yet distributing computing tasks, TransCom can greatly reduce the potential system maintenance and management costs. We have implemented a multi-platform TransCom prototype that supports both Windows and Linux services. The extensive evaluation based on both test-bed experiments and real-usage experiments has demonstrated that TransCom is a feasible, scalable, and efficient solution for successful real-world use.**

***Index Terms*—Centralized management, distributed platforms, cloud computing, virtual disks, heterogeneous services.**

I. Introduction

RAPID advances in hardware, software, and networks in the past decades have made personal computer (PC) a great success. They have been ubiquitously deployed in enterprise environments such as universities, corporations, and government organizations, where many PCs are networked with a few application servers. A great challenge for these systems is the high management overhead for system administrators and end users to maintain the software and to back up and secure the distributed data, resulting in the high total cost of ownership. It is shown that the annual cost of managing a PC can be up to five times the cost of deploying it [1]. Meanwhile, as all files and data are stored on the local disks of individual machines, they may be lost once the corresponding device is damaged or compromised, requiring distributed data backup and restoration services. More seriously, if sensitive server data are fetched and cached at local disks, they will be potentially available to the public or to attackers who have access to the machine.

Recently, cloud computing [2], [3] has become a popular topic in the academe and industry, aiming to provide application software, data, and even hardware as a service hosted in data centers. The power of cloud computing has also been recognized to address the above challenges faced by the PC paradigm. While there are different types of usage, the models can be roughly classified into the following two categories.

The first category is to host the applications, such as Salesforce [4], Google Docs [5], in data centers, and then deliver them to end users through the web browser or other special utilities. This new paradigm can sharply reduce the cost of software maintenance and management by centralizing all of them in the data centers. However, these application programs in cloud computing are specialized and dedicated, making it very difficult for traditional applications (e.g., MS Windows Media Player 10, and Quake 4) to be hosted and delivered. In addition, this just solves the maintenance and management issues of specific applications, which are not concerned with traditional OSes, e.g., Windows.

The other category is a virtual machine (VM) based thin-client approach emerging as virtual desktop solutions in data centers, such as Xen Desktop [6] and VMware View [7], which create virtual PCs/desktops (i.e., instances of Windows) on the server or server blade with virtualization technology. Thus, the user has a complete virtual PC in the data center or cloud, but only consumes a fraction of the resources of the server. The virtual desktop can be accessed from any client devices, such as normal PCs, thin clients, and mobile devices, through a remote display protocol, e.g., remote desktop protocol (RDP) [8], independent computing architecture (ICA) [9], virtual network computing (VNC) [10]. Compared with traditional thin-client systems, a virtual PC/desktop can guarantee and isolate user performance and improve security. However, as a type of thin client, it is very hard to support graphics-intensive multimedia applications due to the huge network bandwidth needed to transfer video display data, even in an enterprise environment.

This paper presents TransCom, a new cloud computing architecture for supporting heterogeneous OSes. Our key

Manuscript received October 29, 2012; revised October 8, 2013. The special issue guest editors coordinating the review of this paper and approving it for publication were G. Martinez, R. Campbell, and J. Alcaraz Calero.

Y. Zhou is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, P. R. China (e-mail: zhouyz@mail.tsinghua.edu.cn).

Y. Zhang is with the School of Information Science and Engineering, Central South University, Changsha, Hunan 410083, P. R. China (e-mail: zyx@csu.edu.cn).

Y. Xie is with the Microsoft Research Silicon Valley, CA 94043, USA (e-mail: yxie@microsoft.com).

H. Zhang is with the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA (e-mail: hzhang@cs.cmu.edu).

L. T. Yang is with the Department of Computer Science, St. Francis Xavier University, Canada (e-mail: ltyang@stfx.ca).

G. Min is with the Department of Computing, University of Bradford, UK (e-mail: g.min@brad.ac.uk).

Digital Object Identifier 10.1109/TNSM.2013.122613.120358

technique for addressing the above challenges is the use of virtual disks (Vdisks), each of which simulates a physical block-based storage device using a disk image file located on the centralized server and accessed via a network connection. Since disk operations are located beneath file systems and applications, TransCom can support heterogeneous OSes and applications, including Windows (it does not have a single kernel file) and its applications whose source code is not available for modification.

With all computing tasks still carried out by the clients, TransCom retains the high performance of PCs, yet significantly reduces the management overhead by completely eliminating the end user management tasks and reducing the inconsistent or faulty software states on local disks. Meanwhile, disk and network bandwidth has been increasing, especially with the advent of RAID technology [11], [12]. Although high-performance storage technologies may be expensive to deploy on PCs, it is realistic to deploy them on servers by amortizing the costs over clients. Such technology trends may lead to a powerful server with faster I/O access that makes access to network-based Vdisks faster than accessing low-performance client local disks. Specifically, TransCom has the following desirable features:

- *Heterogeneous OS support:* TransCom supports heterogeneous OSes with no or minimum OS modification. TransCom clients can choose to boot the desired Oses flexibly via the same remote OS boot process.
- *Flexible software and data sharing:* Our design enables both data and application software sharing. System and application file sharing is transparent to users with a novel file redirector mechanism.
- *User and application transparency:* The use of Vdisks is transparent to users and applications, and requires no application modification. From the perspective of applications and users, there is no difference between accessing data from Vdisks and that from local hard disks.

We have implemented a multi-platform prototype system for running both Windows and Linux at TransCom clients. The Windows-based system has already been deployed at universities' e-learning classrooms for daily use. Our extensive evaluation, consisting of both test-bed experiments and real-usage experiments, has shown that by using a powerful server, TransCom clients can achieve comparable or even better disk and application performance than regular PCs with local hard disks. By centralizing storage yet distributing computing, TransCom is more responsive and scalable than existing thin-client systems or VM-based thin-client-like cloud computing approaches.

The remainder of this paper is organized as follows. Section II presents an overview of the TransCom system architecture. Sections III, IV, and V describe the system in details, including the remote OS boot, Vdisk access and sharing, and its implementations. In Section VI, we present extensive test-bed and real-usage experiment evaluation of TransCom. Section VII discusses possible extensions and optimizations. Section VIII describes the related work before providing the conclusion in Section IX.

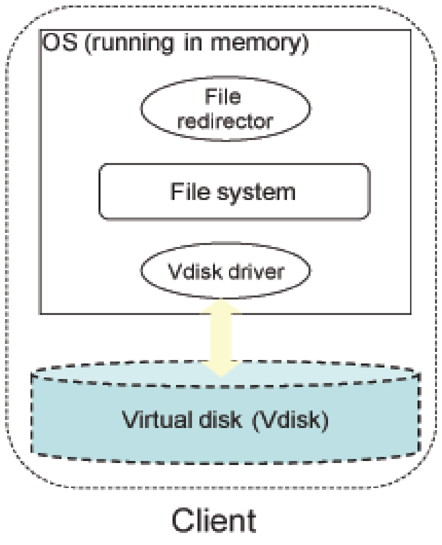


Fig. 1. Overall architecture of a TransCom system with a server and a single client.

II. System Overview

TransCom adopts the conventional client and server architecture, where a single server supports up to tens of client machines connected in a network system. A client machine can be any regular PC without local storage devices. The server can be either a regular PC or a higher-end dedicated machine. In our current design, the TransCom server uses MAC address to identify a unique client. The entire system should reside in a local area network, protected from other networks by network address translator (NATs) or firewalls for security.

Figure 1 illustrates the overall architecture of a TransCom system with a server and a single client. Without local hard disks, each client accesses software (including OSes) and data from one or more Vdisks that simulate physical block-based storage devices. A Vdisk, in essence, is one or more disk image files located on the server and accessed by the client remotely via a Vdisk access protocol (VDAP). Access to Vdisks can be generally supported across different OSes with a Vdisk driver - a specialized device driver running on the TransCom client. Disk buffer requests and page faults are handled regularly through the Vdisk driver as if there existed a local hard disk.

The TransCom client boots from the server remotely to load the desired OS, including the Vdisk driver. The client issues separate requests for OS boot and disk access. When a client is powered up, it makes a boot request to the TransCom server, and uses a remote boot protocol to first download and enable Vdisk access functions. The client then issues Vdisk access requests to launch the desired OS from its own Vdisks. During this OS loading process, a Vdisk driver will be loaded in place of the traditional hard disk driver. Further disk requests will go through the Vdisk driver, and the control of the hardware will be handed to the OS for the boot process to finish regularly as if with a local hard disk (Section III).

By using Vdisks, TransCom enables not only data sharing, but also software sharing for OSes and popular applications. Specifically, multiple user-perceived Vdisks on different

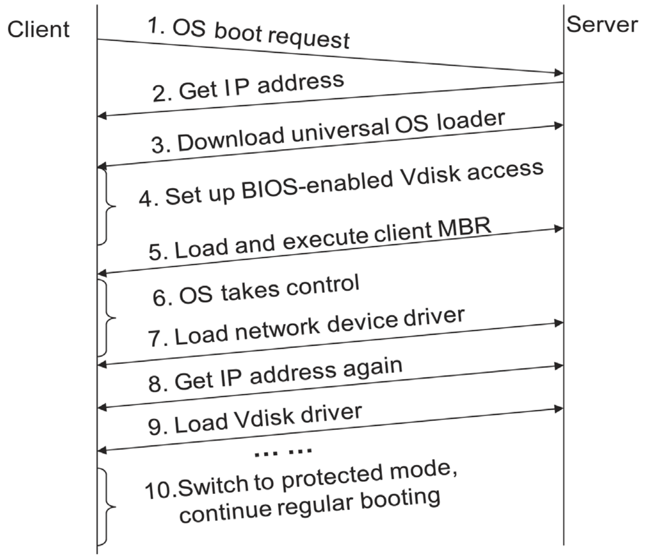

Client
Server
1. OS boot request
2. Get IP address
3. Download universal OS loader
4. Set up BIOS-enabled Vdisk access
5. Load and execute client MBR
6. OS takes control
7. Load network device driver
8. Get IP address again
9. Load Vdisk driver
...
10.Switch to protected mode, continue regular booting

Fig. 2. The remote OS boot process of a TransCom client.

clients can be mapped to the same Vdisk image on the TransCom server. In order to avoid the conflict of multiple users' write and then read operations on the file of the same shared Vdisk, each client uses a file system agent called file redirector to redirect the access of written files on user-perceived Vdisks to the access of different shadow files on user-specific server-perceived Vdisks (Section IV-B). Eventually, every file access request will be converted into one or more disk block access requests, handled by the corresponding Vdisk driver through communication with the server (Section IV-A).

The TransCom server is running as an application daemon. It maintains a client management database, a disk management database, and a list of all Vdisk image files belonging to all clients in the system. The client management database is configured with a list of mappings between clients' IP addresses and their MAC addresses. The Vdisk management database maintains a mapping between clients' IP addresses and the corresponding Vdisk image files. Given a disk access request, the TransCom server first looks up the Vdisk management database to find the corresponding Vdisk images and then performs the requested operations before sending replies back to the client (Section V-B).

III. OS-independent Remote Boot

Figure 2 lists the steps involved in the TransCom remote boot process. As a first step in OS boot, each TransCom client submits an OS boot request to the server and obtains an IP address for subsequent communication. This step can be supported by hard coding a remote boot protocol into the client's BIOS beforehand. In our current implementation, we adopt the existing Intel PXE protocol [13] and tools. The TransCom server maintains a client MAC address to the IP address mapping table in the client management database. Given the boot request, the server assigns the corresponding IP address to the client via DHCP [14] based on the client MAC address.

The client then downloads a universal OS loader from the server using TFTP [15], also supported by PXE (step 3). The purpose of the universal OS loader is to replace the BIOS hard disk access function (e.g., INT 13 in x86 architecture) with a customized one that redirects all disk access requests to the server using VDAP. Once the universal OS loader is up and running, the client will have the ability to access Vdisks remotely from the server (step 4). Thus, the immediate next step is to read and execute the master block record in the client's Vdisk, which is usually the first step in the regular OS boot process (step 5). After this point, the OS takes control to resume initializing various modules, including device drivers and file systems (step 6).

However, as this regular OS boot process progresses, the BIOS-enabled Vdisk access will no longer work due to its real-mode memory management. Most modern OSes access memory in protected mode. Once in control, the modern OS will bypass the BIOS real-mode disk access functions and replace them with its disk device drivers for improved performance. Hence, the booted OS will not be able to access those memory segments that store the BIOS functions in real mode, including the specialized Vdisk access instructions installed by the universal OS loader.

To solve this problem, we replace the normal disk driver with a Vdisk driver in the boot process so that the client machine can continue accessing Vdisks after the OS switches to the protected mode of memory access. Note that disk drivers are typically loaded before the network device drives in the normal OS booting process. Since the Vdisk driver relies on the network functions, we switch the device driver loading order so that the network devices are launched first for establishing connections to the server (steps 7 and 9).

Another important detail is to enable the DHCP client before the Vdisk driver is activated. This is because the IP address obtained under real mode will no longer be accessible after the OS switches to the protected mode. Therefore, after loading network drivers, the TransCom client needs to contact the server again using DHCP to re-obtain its IP address (step 8). By the time that the real-mode access fails, all the required modules for accessing Vdisks, including the network interfaces and the Vdisk driver, will have been loaded into the memory. Once the OS switches to the protected mode, all disk access goes through the Vdisk driver as if there existed a physical local disk (step 10). The OS can resume the rest of the boot procedures until all the other required modules are up and running normally.

IV. Accessing Vdisks

The core concept of TransCom is the notion of Vdisks. From a user or application's perspective, there is no difference between accessing data from a Vdisk and from a local hard disk. The actual contents in Vdisks reside at the remote server, and will be fetched to the client on demand. In this section, we describe the detailed process of accessing data from Vdisks as well as how data are organized and shared across different clients.

A. *Vdisk and Vdisk Access Protocol*

Vdisks are flat addressable block-based virtual storage devices located beneath the file systems. Each block has 512 bytes. A TransCom client can be configured to access data from one or more Vdisks, with each corresponding to a Vdisk image located on the server. A Vdisk image consists of one or more files, depending on the required disk size. To manage the Vdisks of different clients, the TransCom server sets up a Vdisk management database to maintain the mappings between a client's IP address and the corresponding Vdisk images. Administrators can configure the quotas of client Vdisks using a TransCom application tool that updates the server's Vdisk management database.

Vdisk images are regular flat files located at the TransCom server. The contents of a Vdisk image exactly simulate those of a hard disk, by fragmenting the file contents into fixed 512 byte records. Each record corresponds to a block of Vdisk. Note that there exist 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.

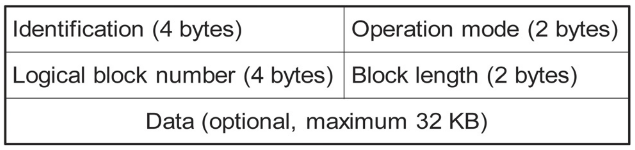
Every Vdisk data access request goes through a specialized Vdisk driver on the client. There are two types of disk requests involved in accessing Vdisk data:

- A *virtual disk request* is a disk access request issued by the file system to the Vdisk driver. It is of the same format as a disk access request to regular SCSI or IDE disk devices. Each request consists of the disk block number, the start offset, and the length of data to be read or written.
- A *remote disk request* is a disk access request issued by the Vdisk driver to the remote TransCom server through network communication. Given each virtual disk request received from the file system, the Vdisk driver will compose one or more remote disk requests to send to the server.

Each Vdisk driver maintains two request queues, one for the virtual disk requests received from the file system and another for the remote disk requests to be sent to the server. For a simple client failover, the network communication between the Vdisk driver and the server uses a UDP-based VDAP. Thus, the TransCom server maintains no client states or active connections, and processes requests individually as they arrive. For reliable data transmission, the TransCom server will send an explicit acknowledgment for each received remote disk request. If the request is a data read request, the acknowledgment can be piggy backed with the returned data.

Vdisk drivers use time-outs to detect lost remote disk requests or replies for request retransmission. Each remote disk request has a unique ID so that both the client and the server can detect and drop duplicate packets. For efficient performance, we would like a small time-out value in case of network loss. Our empirical experience has shown that most disk access involves a small amount of data (see Section VI-G2). Hence, the maximum size of data is set to be read or written by a remote disk request to 32 KB.

To ensure that there are no out-of-order requests, each Vdisk driver uses a simple wait-and-send solution to send remote disk requests. After a request is removed from the queue and sent to the server, the Vdisk driver will not send the next request until it receives the response to the last message. Hence, a read request following a lost write request will not return stale data, and repeated read/write requests are guaranteed to generate the same effects.



Identification (4 bytes)	Operation mode (2 bytes)
Logical block number (4 bytes)	Block length (2 bytes)
Data (optional, maximum 32 KB)	

Fig. 3. Format of VDAP request/response packets.

Figure 3 shows the format of each remote disk request and reply in an application-level packet. Each packet starts with an identification field of 4 bytes to denote the unique packet sequence number. The 2-byte operation mode is used to distinguish between three different types of remote disk packets: read request, write request, and server acknowledgment. The logical block number and block length specify the start block number and the total number of blocks to be read or written on the corresponding Vdisk. If the packet is a write request or a read response, then the actual disk data to be written or read will follow.

Given a client remote disk request, the server first looks up the corresponding Vdisk image based on the client IP address. If the client is allowed to perform the requested operation (see Section IV-B), the server retrieves or updates the requested disk image and sends a response to the client, acknowledging the completion of the process. If the client request is a read request, the server will also attach the requested disk data in the reply.

B. *Disk Organization and Data Sharing*

The disk-level remote data access provides a wide spectrum of design space to share data across different clients at the server. At one extreme end, we can perform coarse-grained sharing based on an entire disk. This solution is simple but not efficient in disk utilization. At the other extreme end, we can perform fine-grained sharing based on each disk block, which could be efficient in disk utilization, but may result in high management overhead due to information booking and access control.

TransCom shares data at the granularity of files since they are the most natural units for sharing and access control in convention. However, since the TransCom server handles disk-level access requests, it is difficult to directly support file-level sharing. To maximize disk utilization, we would like TransCom to support the sharing of system files, including OS and application software, which would be used by multiple clients, while isolating user files that involve private user data. TransCom uses different mechanisms to share system files and user files:

1) Sharing user files: TransCom uses the existing remote file system solutions such as AFS [16], NFS [17], or common internet file system (CIFS) [18] to share user files. This solution gives users flexibility in choosing different remote

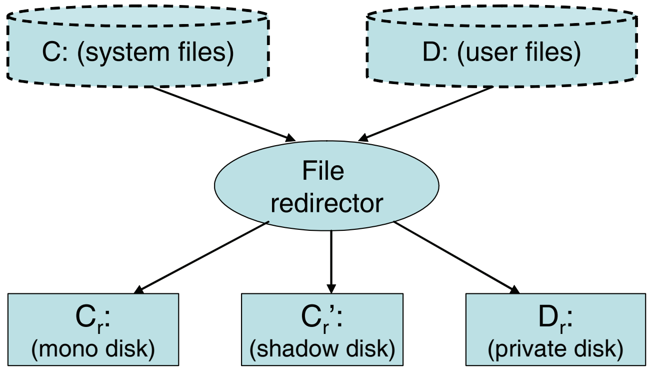


Fig. 4. Vdisk mappings: C and D are user-perceived Vdisks for storing system files and user files, respectively. C_r, C'_r ,and D_r are server-perceived Vdisks for storing shared system files, customized system files, and user files, respectively. The file redirector is a file system agent for redirecting user-perceived Vdisk requests to server-perceived Vdisk requests.

file systems based on the desired degree of sharing. It is also simple, as access control and user authentication will be performed at the client file systems, reducing server management overhead.

2) Sharing system files: However, it would be difficult to use the existing remote file system solutions to share the above defined system files, mostly because certain system files are also user-specific, containing customizable configuration entries, yet having fixed path names for OSes or applications to work correctly. However, client users need to write or modify these system files, resulting in conflicts. For example, the World Soccer Winning 11 game must save the user-specific game progress information in a subdirectory. In order to share such software with NFS-like solutions, users will need to change the directory locations. This potentially requires OS or application software modification and is often impossible in practice.

TransCom uses a file system level agent called file redirector to map user-perceived Vdisks into server-perceived Vdisks for sharing system files. The set of system files is specified by the system administrator, with prior knowledge of potential TransCom usage scenarios. Figure 4 illustrates this idea. From a user's perspective, each client is configured with two types of Vdisks, one for storing shared system files and another for user files. From the server's perspective, each client has three types of Vdisks. The existence of these three types of Vdisks is transparent to the users.

- *Mono disk:* It is used to store OS and application files that are shared across all clients in a TransCom system. The mono disks of different clients are mapped to a single Vdisk image at the server. Thus, mono disk is for read-only files.
- *Shadow disk:* Each client has a shadow disk that is used to store the customized or written system files. The shadow disk maps to a client or user-specific Vdisk image at the server. The corresponding data are private and will not be shared by other client machines. Therefore, the shadow disk is, in essence, a copy-on-write (COW) disk for isolating written user-specific configuration files and system files.
- *Private disk:* Each client has one or more private disks that are used to store private user data. Similar to shadow disk, each client's private disks map to client-specific Vdisk images at the server, and will not be shared.

The file redirector translates the file access requests on user-perceived Vdisks into those on server-perceived Vdisks by intercepting all file system calls. If the file to be accessed locates on the private disk (user-perceived), the redirector simply maps the request to the same file on the server-perceived private disk. If the file to be accessed is on the system disk, the file redirector will redirect the request to the shadow disk in the following two cases: (1) a read request to a system file that already has a customized copy on the shadow disk, and (2) a write request to a system file (in this case, a copy of the file will be created first on the shadow disk before being written). Otherwise, the file redirector will redirect the request to the mono disk. The file redirector therefore supports the dynamic redirection of system files for enabling file system level COW semantics. This software agent will be loaded from the mono Vdisk as part of the underlying file system. We defer the discussion of its implementation in Section V.

V. TransCom Implementation

We have developed a multi-platform (Windows and Linux) prototype of TransCom based on the Intel x86 architecture. In this section, we present the implementation details.

A. Client Implementation

We choose Windows XP as the Windows client OS to show back compatibility on low-end hardware. For Linux client OS, we adopt RedFlag 4.1 Desktop [19] (Linux kernel 2.4.26-1). There are three major implementation modules for supporting TransCom clients under different OSes: (1) universal OS loader, (2) Vdisk driver, and (3) file redirector.

1) Universal OS loader: Our current implementation uses Intel PXE as the remote boot protocol for sending boot requests. The PXE tools are burned into the client BIOS beforehand. With the x86 architecture, the universal OS loader will replace the INT 13 interrupt code to enable real-mode Vdisk access. It is implemented as 15,000 lines of ASM code. The implementation of the universal OS loader is independent of the client OS to be loaded.

2) Vdisk driver: Because device drivers are platform dependent, we implement two different Vdisk drivers, customized for Windows and Linux, respectively. The implementations are in C++. The loading procedure of the Vdisk driver is also non-trivial and platform dependent. This is because the TransCom client must load the network device drivers before loading the Vdisk driver in the remote OS boot procedure, as discussed in Section III.

Since Windows XP is a modified microkernel, we simply modify the corresponding Windows registry files to ensure that the network device driver gets loaded before the Vdisk driver. There is no need to change or recompile the kernel. However, as Linux is a monolithic kernel, the Vdisk driver has to be compiled into the kernel. We also change the initialization sequence of the device drives by modifying the related kernel source code before recompilation.

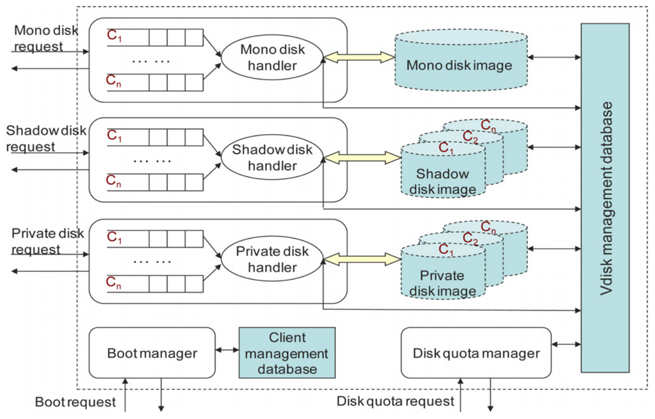


Fig. 5. TransCom server architecture.

There is also need to enable the client DHCP service for re-obtaining the client IP address before loading the Vdisk driver. While the Linux kernel already provides DHCP facility, the DHCP client provided by Windows XP operates in user mode, and will not be launched until the kernel boots successfully. Therefore, for Windows, we also implement a DHCP client driver in the kernel mode to enable IP configuration before the kernel initializes TCP/IP.

The currently implemented Vdisk driver uses a time-out value of 100 ms for sending remote disk requests to the server. It does not support device driver level cache or read ahead functions. Such optimization can potentially increase performance, and is a topic of our ongoing work.

3) File redirector: The file redirector is a file system agent that intercepts all file system calls. For Windows XP, it is implemented as a file system driver sitting on top of the device driver chain for handling file system operations. It intercepts the I/O request packet (IRP) from the built-in Windows I/O manager to perform the redirection, as described in Section IV-B. For Linux, we modify the `open()` file system call provided by the virtual file system of the Linux kernel, by maintaining a dirty table that records the mappings between the modified system files and their corresponding shadow disk versions.

B. Server Implementation

The TransCom server is implemented as a multi-process program, with each process listening on a dedicated port, handling different types of client requests (Figure 5).

Specifically, there are three disk request handler processes handling the requests for three different types of disks. Each process maintains a request queue per TransCom client, and serves queues in a round-robin fashion. Conceptually, there is only one Vdisk management database at the TransCom server. However, in the implementation, each disk request handler deals with a dedicated Vdisk management database implemented as a file. As discussed in Section IV-B, there exists only one mono Vdisk image for all TransCom clients. For shadow disks and private disks, the TransCom server maintains one Vdisk image per client disk. The default mono disk size is 8 GB, and the default shadow disk and private disk sizes are set to 1 GB per disk.

The boot manager handles boot requests with a DHCP service and a TFTP service for clients to obtain IP addresses and to download the universal OS loader, respectively. The disk quota manager is responsible for processing disk quota-related requests. System administrators can log in from a pre-configured special client to submit disk quota requests to the server and change the number of user disks. After that, a client must then reboot to reflect the changes.

The server is implemented in C++ for both Windows and Linux. The Windows server OS is Windows 2003 Server (SP1) edition, and the Linux server OS is RedFlag DC Server 5.0 (Linux kernel 2.4.21-32)[1].

VI. Performance Evaluation

This section evaluates TransCom performance using both test-bed experiments and real-usage experiments. We focus on the TransCom Windows prototype, which has already been deployed so that it is feasible to collect real-usage data.

We are primarily interested in the following questions: (1) What is the Vdisk access performance? And how does it compare against the performance of accessing local disks and VM disks? (2) What is the TransCom client OS boot latency? (3) How does Vdisk access impact file system and application performance? (4) What is the scalability of TransCom, and how does it compare against other existing solutions? (5) What is the TransCom performance in real-world usage?

A. Experiment Setup

We use five sets of experiments to investigate the above questions. In all the test-bed experiments, TransCom clients are configured as Intel Celeron 1 GHz machines, each with 128 MB DDR 133 RAM and a 100 Mbps onboard network card. The server is configured as an AMD Athlon 64 3000+ machine, with 2 GB Dual DDR 400 RAM, two 80 GB Seagate Barracuda 7,200 rpm soft RAID0 hard disks, and a 1 Gbps onboard network card. The clients and the server are connected by an Ethernet switch with 48 100 Mbps interfaces (used for clients) and two 1 Gbps interfaces (one used to connect the server). We also compare the TransCom client performance with a regular PC, which has the same client hardware configuration but with an additional local hard disk (80 GB Seagate Barracuda 7,200 rpm), and a VM emulating the stateless thick-client-like approaches [20], [21], which is virtualized with 128 MB memory and 8 GB (dynamically expanding) SCSI hard disk using VMWare Workstation 5.0 hosted by Windows XP SP2 with NTFS 5.0 file system on the same regular PC hardware (but with 512 MB physical memory). The server OS is Windows 2003 Server (SP1) edition running an NTFS 5.2 file system and Redflag DC Server 5.0 running EXT3 file system. The TransCom clients, the regular PC, and the stateless thick clients all use Windows XP SP2 with NTFS 5.0 file system. Among the 128 MB client memory, about 90 MB is used for the working set of Windows, 8 MB is used as the video frame buffer, and the remaining 30 MB is free after the OS boot.

[1]In fact, these two servers can be merged into one. We separate them here just for consideration of user customization in real deployment. The single one-server version is also available in the lab.

TABLE I
AVERAGE TIME SPENT AT VARIOUS STEPS IN PROCESSING A DISK READ OR WRITE REQUEST OF TWO DIFFERENT SIZES (μS).

Operation	Vdisk read (4 KB)		Vdisk read (8 KB)		Vdisk write (4 KB)		Vdisk write (8 KB)	
	Windows	Linux	Windows	Linux	Windows	Linux	Windows	Linux
Queuing	6.44	1.02	6.38	1.03	6.35	1.02	6.51	1.13
Request pack	2.58	1.02	2.39	1.06	17.04	1.27	32.57	1.29
Reply parse	9.92	4.51	15.69	4.82	5.53	2.15	5.74	2.85
Server parse	5.25	1.06	5.04	1.08	7.43	9.82	8.57	11.84
File system I/O	18.53	372.67	19.99	432.05	541.49	531.53	2,407.2	642.66
Reply pack	1.60	1.04	1.56	1.06	1.62	1.03	1.66	1.06
Network	676.45	657.54	1,050.85	1,071.92	571.04	669.23	1,076.05	1,167.80
Total	720.77	1,038.86	1,101.9	1,513.02	1,150.5	1,216.05	3,538.3	1,828.63
Local disk	7,407.41	8,443.80	7,462.69	8,557.61	5,208.33	7,928.82	5,347.60	8,264.10
VM-based disk	8,264.46	9,097.50	8,474.57	9,317.59	5,464.48	8,759.97	5,617.97	8,887.60

B. *Vdisk Performance*

We evaluate the Vdisk access performance in terms of latency and throughput in a single-client TransCom system.

1) Vdisk access latency: We first examine the latency spent on various steps of accessing Vdisk data. We disabled the client file system cache, and used the Microsoft I/O performance evaluation tool SQLIO [22] in Windows (with the testing file size being 512 MB) and Iometer [23] in Linux environment to submit random disk access requests of different sizes to the client. Because the request data size is small, each disk access request triggered only one remote disk request in the experiments.

Table I shows the average measured latencies of 20 disk requests. The standard deviations are small (less than 10%) and are omitted. The "total" latency corresponds to the time elapsed between the Vdisk driver receiving a disk access request from the file system and returning the results back to it. The first three steps are processed by the client Vdisk driver. The "queuing" step measures the queuing delay before the Vdisk driver starts processing the request. The "request pack" step corresponds to the time the Vdisk driver spends parsing the request and packing it into remote disk requests for sending to the server. The "reply parse" step corresponds to the latency for the Vdisk driver to parse the server reply and return the results back to the file system.

The next three steps are processed by the server, including parsing the received requests and looking up the corresponding Vdisk image ("server parse"), performing the requested disk access operation ("file system I/O"), and packing replies for sending to the client ("reply pack"). Finally, the "network" step is the time for sending both the request and the reply over the network.

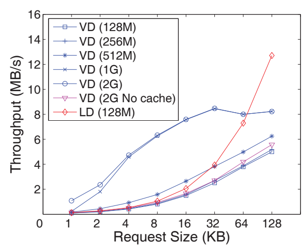
For Vdisk read requests, the majority of time is spent on network transmission. The server file system I/O accounts for only a small fraction of latency. This is mostly due to the server-side memory cache, which is further explored in the next section. For write requests, in addition to the network communication, the server file I/O is also a bottleneck, particularly when we increase the amount of data to be written each time. The total latency is on the order of milliseconds, which is acceptable to most users and applications today. Because the bottleneck of read access is the network communication, we expect the performance to be improved when we increase the network speed (e.g., using 1 Gbps network card) and use TCP/IP off-load engine (TOE) technology. To reduce the disk write latency, we can additionally optimize the server Vdisk write operation by increasing the I/O speed or implementing application-level write optimization schemes such as lazy write.

We also compare the total latency with that of local disks and VM-based disks in stateless thick-client-like systems. The latency of the VM disk access is a little more than that of the local disks, showing a virtualized overhead of around 10% in read operation and 5% in write operation, respectively. However, these two latencies are much more than that of TransCom. This is because the virtualization of disk in the TransCom is not only implemented inside the operating system, but also can leverage the high-performance network and server capacity.

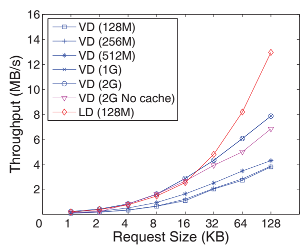
2) Vdisk throughput: We evaluate the Vdisk throughput, also using SQLIO for Windows and Iometer for Linux. Previous studies have shown that most file access involves random disk access with small request sizes [24]. We thus focus on the random disk read/write, and compare the performance with the local hard disk throughput measured from the regular PC and that of the VM-based disk in stateless thick-client-like systems (see Section VI-A).

Figures 6(a)-(f) show the Vdisk read and write throughput in various scenarios in Windows. For read access, the Vdisk throughput ("VD") increases with the request size, but saturates when the request size is larger than 32 KB, which is the maximum size handled by a remote disk request. This is because the network communication time dominates read latency, and a large request size will result in multiple remote disk reads.

When the request sizes are small ($\leq$ 64 KB), the Vdisk read

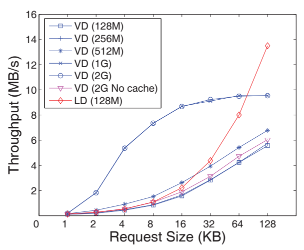


(a) Unbuffered



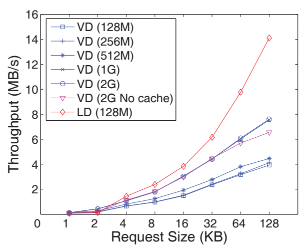
(c) Buffered

Read, RAID0



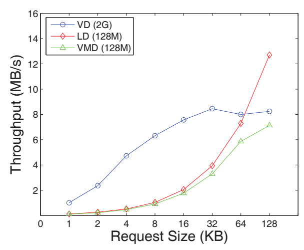
(e) Unbuffered

Read, IDE

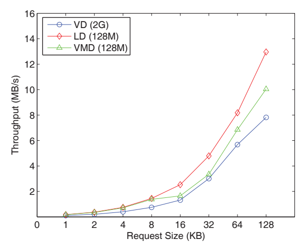


(g) Unbuffered

Read, Linux

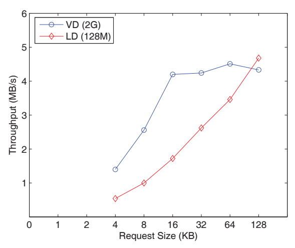


(b) Unbuffered



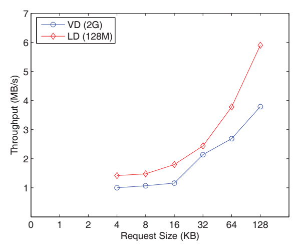
(d) Buffered

Write, RAID0



(f) Unbuffered

Write, IDE



(h) Unbuffered

Write, Linux

Fig. 6. Random Vdisk throughput: "buffered" means the client file system and device driver buffers were enabled, while "cache" means the server file system cache was enabled.

throughput obtained using the 2 GB memory server is higher than the local disk throughput ("LD"). We vary the server memory size and observe that decreasing the server memory size significantly reduces the performance. When the server is configured with the same 128 MB memory as the regular PC, the Vdisk read throughput is lower than the hard disk throughput. For further investigation, we disable the server file system cache in the 2 GB memory case and observe a significant performance drop. Thus, a large server memory cache is the key factor for explaining the high Vdisk read throughput.

For write requests, the throughput increases monotonically with larger request sizes as the server file system I/O performance is the dominant factor for Vdisk write. The server memory cache does not play a significant role. Interestingly, for local disk access, the write throughput is higher than the read throughput. One reason could be the lazy write optimization implemented by the OS and the hard disk, in which case data are written to the cache instead before the operations are returned successful. This reason may also explain why a larger server memory increases the Vdisk write throughput.

We also compare the disk throughput with and without client file system and device driver buffers. When the buffers were enabled, the performance increased slightly as expected. There is no significant difference between accessing RAID0 and IDE disks at the server, with the former having slightly higher performance.

Comparing the throughput with that of local disk ("LD") and VM-based disk ("VMD") in Figures 6(e)-(f), we can see that the read and write performance of the TransCom is better.

We also measure and compare the disk throughput with local disk in the Linux environment. As shown in Figures 6(g)-(h), the trend is very similar to that in Widows, but only with a relatively low value. This may be due to the difference in OSes and evaluation tools.

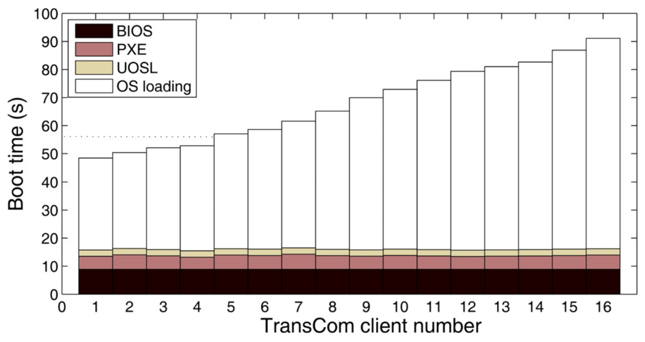


Fig. 7. Average time spent at various steps for a TransCom client to boot remotely from the server. We vary the number of simultaneously booting clients and report the average latency observed among all clients.

Note that TransCom cannot achieve better performance than local disks or even VM-based disks all the time. In fact, we find that in sequential read/write cases, TransCom perform worse than local disks. TransCom can only achieve better disk access performance when the request size is small and the fast server disk and server memory cache can make up the network transportation latency in random read/write cases.

C. Remote Boot Performance

In this section, we evaluate the remote OS boot latency, and compare it against a regular PC boot latency. To time-stamp the boot process of a client, a dedicated monitor machine was linked to the client with a hub that is connected to the server via the Ethernet switch. The monitor machine collected all observed network packets to and from the client using the EtherPeek tool [25]. For each experiment, we repeat the above

TABLE II
AVERAGE TIME SPENT AT VARIOUS PHASES OF THE APACHE WINDOWS SOURCE TREE BENCHMARK (SECOND).

Phase	TransCom	Local disk	CIFS	VM-based disk
mkdir	0.93	1.03	2.93	1.69
cp	27.58	58.36	81.96	49.01
scan dir	92.41	89.88	156.43	283.61
cat	175.56	214.64	296.78	550.13
make	330.82	319.89	535.37	628.10
Total	627.30	683.79	1,073.47	1,512.54

process to measure the boot latency of every client in the system and compute the average latency across all clients. Figure 7 shows the time spent on various steps by varying the number of clients booting simultaneously in the system.

The "BIOS" step is the time spent by the client launching the hardcoded BIOS program. "PXE" is the time spent in obtaining an IP address and downloading the universal OS loader. "USOL" is the time for initializing the universal OS loader to read the Vdisk MBR. These three steps take constant amount of time regardless of the number of clients, and account for a small fraction of the total boot latency. The majority of boot time is spent in loading the OS remotely from the server ("OS loading"). This portion of time increases when the number of clients is increasing.

Overall, with a small number of clients (fewer than five), the total boot latency of a TransCom client is even smaller than that of a regular PC (The dotted line marked value denotes the total latency) is due to that its Vdisk throughput is higher than that of the local disk in our experiments. The total latency increases roughly in line with the number of clients within our range, and is on the order of tens of seconds.

D. File System Performance

In this section, we evaluate the overall file system performance of a TransCom client using a modified Andrew benchmark [16], [26]. We compare its performance against the file system performance of a regular PC with a local disk, the CIFS [18], and the VM-based disk in stateless thick-client-like systems. For CIFS, the same TransCom client and server hardware configuration are used. In our benchmark, the Windows Apache 2.0.53 source tree is used, which has 39.3 MB data before compilation and 42 MB data after compilation. Table II shows the average performance over five runs. For each run, we rebooted both the client and the server to clean various caches.

TransCom achieves better performance than both a regular PC and CIFS, except for the "scan dir" phase, which requires accessing a large number of directories and files. With Vdisks, this phase will result in a large number of remote disk requests, and thus incurring larger network communication overhead.

Our file system performance evaluation shows that, by using a more powerful server and fast network access, TransCom can achieve file access performance that is comparable to a regular PC, and can potentially perform better than other remote file system solutions and that of VM-based approaches in stateless thick-client-like systems. Note that the file performance of TransCom depends on the file size and access model due to the different undergoing disk read/write size and access model, as shown in Section VI-B. The result here cannot indicate that TransCom can achieve comparable file access performance on any other set of files.

E. Application Performance

In this section, we study how virtual disks impact real application performance. We compare TransCom with a regular PC, the VM-based stateless thick-client-like approach, and a number of thin-client systems, including Microsoft RDP 5.2 (included in Windows Server 2003) [8], Citrix MetaFrame XP Server for Windows Feature Release 3 (with a client of Citrix ICA 6.2) [9], and VNC Viewer free Edition 4.1.1 [10]. All the thin clients run on the same platforms as the regular PC, as described in Section VI-A. The corresponding servers use the same configuration as the TransCom server. Note that the performance of these thin-client systems also estimates the performance of VM-based thin-client-like systems in cloud computing.

We use web browsing and video playback as our two applications. Web browsing performance is measured with Microsoft IE 6.0 using the web text page load test provided by the i-Bench benchmark suite 5.0 [27]. This benchmark consists of a sequence of 30 different web pages, each containing mixed text and graphs. To compare the performance across other thin-client systems, we set the window resolution to 800×600. Meanwhile, video playback performance is measured using Windows Media Player 9.0 to play a 21-second (320×240 pixels, 24 frames per second) video clip displayed at 800×600 resolution. For both applications, we use a packet monitor to capture network traffic and measure the performance using slow-motion benchmarking [28], which allows the quantification of performance in a non-invasive manner.

For web browsing, we examine both the average page download latency and the average amount of data to be transferred between the client and the server with different network speeds. TransCom achieves similar performance ("TRC") as that of a regular PC ("PC"). Since RDP and VNC ran the web browsers on more powerful servers, they incur smaller client-perceived latency than TransCom, but require larger amount of data to be transferred between the client and the server. The ICA client experiences a significantly higher browsing latency, even though the system transfers the least amount of data by using a higher compression rate algorithm for the screen display. There is slightly higher latency than a regular PC but similar data size in the VM-based stateless thick-client-like approach ("STC"). This means that while it requires only a small data transfer as in PC and TransCom, the stateless thick-client approach decreases the computing performance due to the virtualization of both the CPU and memory, in addition to disks.

For video playback, we use the playback quality, defined in the slow-motion benchmarking, in addition to the data transfer size, as our performance metrics. Since the Vdisk read throughput with a powerful server is comparable or even better than the hard disk throughput on a regular PC, the TransCom client can achieve good playback quality ("TRC") as that

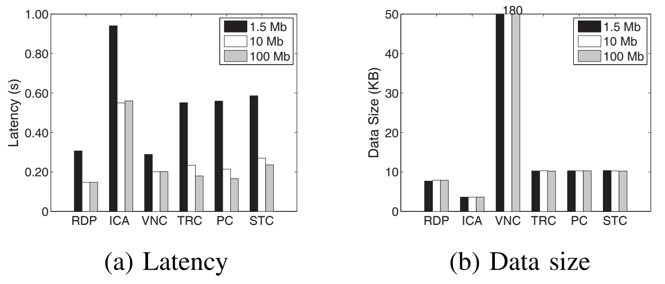


Fig. 8. Web access performance in terms of access latency and the total amount of data transferred.

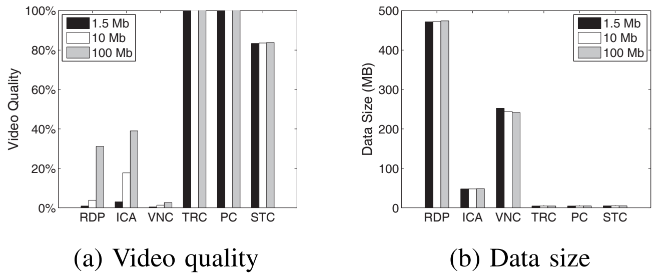


Fig. 9. Video playback performance in terms of video quality and the total amount of data transferred.

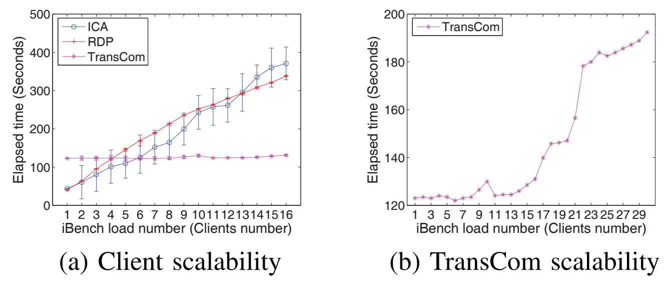


Fig. 10. (a) The client observed i-Bench run latency by varying the number of clients in the system; we plot both the mean and the standard deviations. (b) The TransCom client observed i-Bench run latency by varying the number of clients in the system; we plot only the mean.

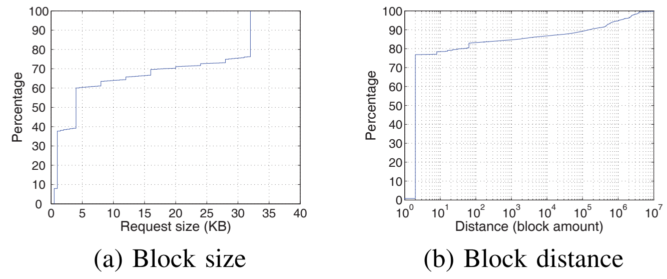


Fig. 11. (a) Distribution of the read request block size; (b) distribution of the initial block position distance between two consecutive read requests

of a regular PC (“PC”), significantly outperforming other thin-client solutions. In these thin-client systems, the servers need to perform expensive video clip decoding for sending display data to the clients, resulting in longer server processing latencies and larger amount of data to be transferred, which explains the low playback qualities at the clients. However, due to the virtualization of the CPU, memory, and graphics, the stateless thick-client approach results in a decrease in video quality (“STC”) while there is a similar amount of data to be transferred compared to the PC and TransCom.

In summary, the web browsing and video playback experiments demonstrate the robust performance of TransCom across different types of popular applications. They show that TransCom can achieve better performance than the virtualization of all hardware resources in the stateless thick-client approach with the same hardware configurations. They also indicate that TransCom can achieve better performance than thin-client solutions with the similar hardware configurations, especially for applications that require heavy computation.

F. Scalability

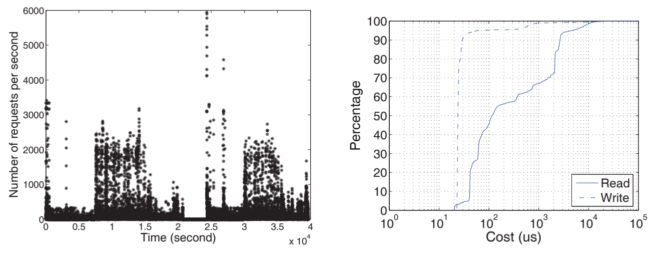
We study the scalability of TransCom by varying the number of clients in the system. We use i-Bench 5.0, but without slow-motion benchmarking, as our workload to evaluate the average client latency of the entire i-Bench run, and compare the performance with both ICA and RDP thin-client systems (Figure 10(a)). When the number of clients is smaller than four, the ICA and RDP achieve lower client latencies than TransCom. However, as the latencies increase linearly with the number of clients in both the ICA and RDP, the client latency in TransCom remains constant with small deviations, suggesting that TransCom is more scalable. This is because the TransCom server handles only Vdisk access requests, while thin-client servers perform both file access and computing tasks. In order to further investigate the scalability of TransCom, we also study the performance with a larger number of clients and present this in Figure 10(b) with a small scale. This shows that when the number of clients is small, the slope is also small, but when it is larger than 15, the slope gets bigger. Even in the case of large number of clients, the elapsed time of TransCom is still smaller than that of the ICA and RDP.

G. Real-usage Experiment

TransCom has been deployed in universities' e-learning classrooms and used by students in their daily lives. In this section, we study the real-world usage of TransCom by instrumenting the server and letting it run for three days in one such e-learning classroom. The classroom has more than 100 client machines, and is dedicated for students to learn English online from 8 am to 11 am and from 2 pm to 5 pm daily.

For our experiments, we isolated 20 clients (the typical number of clients connected to a server) and connected them to our instrumented server. The server is an Intel Pentium IV 2.8 GHz PC with 1 GB RAM, an 100 Mbps network card, and an 80 GB 7,200 rpm soft RAID0 hard disk. For each disk request, we recorded the requested initial block number, the block length, and the operation type. There are about five million requests collected in total, with 80.5% being read requests, and the remaining 19.5% being write requests.

1) Vdisk access workload: To characterize the request workload, we first examine the read request block size distribution in Figure 11(a). The disk write request size is fixed to 2 KB in our real deployment. The majority of read requests involved only a small amount of data, with 60% of read access



(a) Request arrival rate (b) Server process latency

Fig. 12. (a) Request arrival rate over time; (b) distribution of server disk request process latency.

requesting less than 5 KB data each time. We have shown in Section VI-B2 that the TransCom client read throughput (with a powerful server) can be higher than that of a regular PC for small-size disk read access. Thus, in these real-world scenarios, TransCom users will experience similar or even better disk performance compared with a regular PC.

We further study how many read requests can benefit from the server memory cache, which is a key factor for achieving high Vdisk performance. We plot in Figure 11(b) the distribution of the distance between the start block numbers requested by two consecutive read access. A small distance means that the two consecutive disk read requests have strong spatial locality in accessing data. After the first request, the data requested by the second read request are likely to be in the server memory. More than 75% of the consecutive read requests asked for data whose initial block numbers differed by only a few blocks, suggesting that disk read requests have strong temporal and spatial locality in real usage, where TransCom is the most helpful in terms of performance.

2) Real-usage Performance: Given 20 TransCom clients, we examine both the request arrival rate and the server process latency in real-world scenarios. Figure 12(a) plots the client request arrival rate over time for one of the three data collection days. The request patterns in the other two days look similar. For most of the time, the request rate was low, on the order of hundreds of requests per second. There were some request burstiness during the morning and afternoon active usage periods, with peak rate as high as 6,000 requests per second. The short quiet period in which no request was received was during the rest time at noon.

Figure 12(b) shows the server request process latency distributions for both the read and write disk access. Because the maximum write request size is 2 KB (four disk blocks), more than 90% of the write requests are finished in tens of microseconds. For read requests, the server process time varies due to the different request sizes, with the majority of them (more than 60%) finishing within 1 ms. While there was a small percentage ($\leq 10\%$) of requests having a relatively long processing latency, which may be due to the burstiness in the number of requests, all disk access requests are finished within 100 ms.

The strong locality of disk access patterns suggests that TransCom may outperform regular PCs with local disks in terms of real-usage.

VII. Discussion

This section discusses the possible extensions of and optimization to TransCom for enhancing cloud services, system performance, scalability, robustness, and security.

In TransCom, each different computing environment, including OS and its above applications (one or more) can be encapsulated into a Vdisk image, which can be started up and used on any client as a cloud service. Users can subscribe and then be authorized to use these cloud services. Therefore, there are several ways to provide cloud services based on TransCom system. For one example, a school administrator can create several different Vdisk images, each of which consists a specific OS and the necessary applications for different classes. Thus, for each class, students can only choose and use the assigned services, concentrating on the teaching objects. For another example, an enterprise administrator can create different Vdisk images for the staffs. Each Vdisk images consists the just OS and applications. The staff can only run the authorized services. If a staff has more than one roles, he/she can be authorized to run different cloud services.

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

In order to break through the capacity of a single server and support large-scale users, we can employ the existing cluster computing and cloud computing technologies as complementary solutions. For example, we can use a load balance approach (e.g., LVS [29]) to support multiple TransCom servers. Also, we can use HDFS [30] or other large-scale distributed file systems to store and manage a huge number of Vdisk images. We are considering to integrate these technologies into TransCom in future.

Our current prototype does not implement automatic server failover. When the server crashes or the mono disk image must be updated, the entire system needs to be manually shut down and rebooted. As future work, we plan to use server replication mechanisms (e.g., [31], [32]) to handle these scenarios, wherein clients can switch to an identical backup server when required.

The TransCom server needs to prevent and detect unauthorized access of information. The current use of MAC addresses for authentication cannot protect against malicious attackers that spoof clients' MAC addresses. A user-level authentication mechanism (e.g., [33]) might help mitigate the possibility of such attacks. We can also augment the current system by using various encryption-based approaches to protect the privacy of disk data access.

VIII. Related Work

There has been extensive research on distributed and cloud computing platforms. Our work is mostly related to systems such as cloud computing systems, thin clients, network file systems, and VM-based systems.

The cloud services, such as Salesforce [4], Gmail [34], Google Docs [5], centralize both computation and storage in data centers and then deliver the applications to end users through the web browser or other special utilities. This new paradigm can sharply reduce the cost of software maintenance and management by centralizing all of them in data centers. However, these application programs in cloud computing are specialized and dedicated, making it very difficult for traditional applications to be hosted and delivered. In addition, this just solves the maintenance and management issues of specific applications, which are not concerned about traditional OSes, such as Windows.

Thin-client systems have been very popular, providing a full-featured desktop to users with low management costs and pervasive access. Example systems include Microsoft RDP [8], Citrix ICA [9], Sun Ray 1 [35], VNC [10], and MobiDesk [36]. In the thin-client system paradigm, all computing tasks are performed at the central server, 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) cannot be supported by thin-client systems efficiently. Furthermore, there is no isolated user performance guarantee.

Recently developed VM-based thin-client approaches in cloud computing, such as Xen Desktop [6] and VMware View [7], also leverage the thin-client model to access the virtual PC or desktop hosted in data centers. This virtual PC or desktop is hosted by a VM created and run within a server. Although approach can guarantee and isolate user performance and achieve higher security with the help of VM technologies, it is still hard to support graphics-intensive applications due to the large network bandwidth needed to transfer video display data over the Internet.

Network file systems and devices, such as NFS [17], AFS [16], and NAS [37], 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 we described in Section IV-B.

Our idea of centralizing storage while distributing computing is similar to the concept of diskless computers (e.g., [38], [39]) in the early years. Without local hard disks, a diskless computer usually downloads an OS kernel image from the remote server. Hence, it cannot support OSes that do not have clear kernel images, such as Windows, and does not support booting from heterogeneous OSes. Furthermore, the Vdisks perceived by TransCom users can be flexibly mapped to the Vdisk image files on the server. Such flexibility allows TransCom to share OS and application software across clients to reduce the storage and management overhead while still isolating personal files for user privacy.

The iSCSI protocol has been used to access disk blocks through network communication [40]. In particular, the iBoot [41] project at IBM has proposed a method that can remotely boot a commodity OS through iSCSI. An iBoot client needs a special type of BIOS ROM to carry out the OS boot process, hence it is not generally applicable. For better performance, TransCom can potentially adopt iSCSI to replace the current VDAP, but may need to modify it in order to fit the small-size client BIOS memory.

The concept of resource virtualization has been introduced a long time ago, but has just been recently adopted to address security, flexibility, and user mobility. For example, commercial products such as VMware [42] have extended the concept of VM to support multiple commodity platforms. The disks in these VMs are also virtualized, but they reside in the local host machine and are accessed through the file system of the host OS. In contrast, TransCom virtual disks are located in the remote server, with different types of Vdisks for sharing and isolating data among users.

VM-based stateless thick-client approaches, such as internet suspend/resume (ISR) [21], use virtual machine technology (e.g., VMware) together with a network file system (e.g., Coda [43]) to support user mobility. Each ISR client runs OSes and applications on top of a pre-installed VMware on the host OS. The use of VM supports heterogeneous OSes, but it also introduces additional performance overhead due to the virtualization of all hardware resources, including the CPU and memory, whereas in TransCom, client OSes are running directly on top of the CPU, memory, and graphics resources.

SoulPad [44] is another project that uses VM concept for mobility, with a portable storage device for storing the entire VM image. The collective project [45] proposed a cache-based system management model based on VMs to reduce the management tasks of desktop computing. Similar to TransCom, it also uses different types of virtual disks, among which an immutable system disk is presented to protect it against outside threats. Compared with collective, TransCom uses a COW semantics file instead of COW disks. Moreover, TransCom adopts on-demand block-level disk access instead of using network file systems, such as NFS, to access and cache disk images.

IX. Conclusion

We have developed TransCom, a novel virtual disk based cloud computing architecture for enterprise environments. TransCom clients store all data and software on virtual disks that correspond to Vdisk images located on a centralized server. By using disk-level remote data access, TransCom can flexibly support running heterogeneous services like OSes, including Windows.

We performed both test-bed experiments and real-usage experiments to evaluate TransCom. The results demonstrate that by using a powerful server, TransCom clients can achieve comparable or even better disk and application performance than regular PCs with local hard disks. It is more scalable than traditional thin-client systems and other recently developed VM-based cloud computing systems.

Future work includes further optimizing TransCom performance, supporting multiple servers, increasing the system robustness, enhancing the system security, and supporting more types of devices and networks.

Acknowledgment

This work is supported by National High-Tech Research and Development Plan of China under Grant No. 2011AA01A203, and by the International Science and Technology Cooperation Program of China (Grant No. 2013DFB10070). The authors would like to thank the students, especially Guanfei Guo, Li Wei, etc., for their contributions in the development and experiments.

References

[1] "Understanding Full Virtualization, Paravirtualization, and Hardware Assist," White Paper, VMware Inc., Sept. 2007.
[2] B. Hayes, "Cloud computing," *Commun. ACM*, vol. 51, pp. 9–11, July 2008. Available: http://doi.acm.org/10.1145/1364782.1364786
[3] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Proc. 2008 Grid Computing Environments Workshop*, pp. 1–10.
[4] "Salesforce platform," http://www.salesforce.com, 2012.
[5] "Google Docs," https://docs.google.com/, 2012.
[6] "Citrix XenDesktop," http://www.citrix.com/virtualization/desktop/xendesktop.html, 2012.
[7] I. VMware, "VMware View 4.5, Modernize Desktop and Application Management, V.2.0, Brochure," http://www.vmware.com/files/pdf/VMware-View-45-DS-EN.pdf, 2012.
[8] B. Cumberland, G. Carius, and A. Muir, "Microsoft Windows NT Server 4.0, Terminal Server Edition: Technical Reference," Microsoft Press, 1999.
[9] I. Boca Research, "Citrix ICA Technology Brief, Technical White Paper," Boca Raton, 1999.
[10] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper, "Virtual Network Computing," *IEEE Internet Computing*, vol. 2, no. 1, pp. 33–38, 1998.
[11] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: high-performance, reliable secondary storage," *ACM Computing Surveys*, vol. 26, no. 2, pp. 145–185, 1994.
[12] R. J. T. Morris and B. J. Truskowski, "The evolution of storage systems," *IBM Systems J.*, vol. 42, no. 2, pp. 205–217, 2003.
[13] "Intel Corporation. Preboot Execution Environment (PXE) Specification, Version 2.1," 1999.
[14] R. Droms, "Dynamic Host Configuration Protocol," RFC 2131, 1997.
[15] K. Sollins, "The TFTP Protocol," RFC 1350, 1992.
[16] J. H. Howard, M. L. Kazar, and S. G. Menees, "Scale and performance in a distributed file system," *ACM Trans. Computer Systems*, vol. 6, no. 1, pp. 51–81, 1988.
[17] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and implementation of the sun network filesystem," in *1985 USENIX Association Conference Proc.*
[18] P. Leach and D. Perr, "CIFS: A Common Internet File System," Microsoft Interactive Developer, 1996.
[19] "RedFlag Linux," http://www.redflag-linux.com, 2011.
[20] N. Tolia, D. G. Andersen, and M. Satyanarayanan, "Quantifying interactive user experience on thin clients," *IEEE Computer*, vol. 39, no. 3, pp. 46–52, 2006.
[21] M. Satyanarayanan, B. Gilbert, M. Toups, N. Tolia, A. Surie, D. R. O'Hallaron, A. Wolbach, J. Harkes, A. Perrig, D. J. Farber, M. A. Kozuch, C. J. Helfrich, P. Nath, and H. A. Lagar-Cavilla, "Pervasive personal computing in an internet suspend/resume system," *IEEE Internet Computing*, vol. 11, pp. 16–25, Mar. 2007. Available: http://portal.acm.org/citation.cfm?id=1256316.1256355
[22] L. Chung, J. Gray, B. Worthington, and R. Host, "Windows 2000 Disk IO Performance," Technical Report MS-TR-2000-55,Microsoft Research, 2000.
[23] "Iometer," http://www.iometer.org, 2011.
[24] W. Vogels, "File system usage in Windows NT 4.0," in *Proc. 1999 Symposium on Operating Systems Principles*.
[25] "Etherpeek 4," http://www.wildpackets.com, 2011.
[26] L. P. Cox, C. D. Murray, and B. D. Noble, "Pastiche: making backup cheap and easy," in *Proc. 2002 USENIX Symposium on OSDI*.
[27] "i-Bench," ftp://ftp.pcmag.com/benchmarks/i-bench/, 2011.
[28] J. Nieh, S. J. Yang, and N. Novik, "Measuring thin-client performance using slow-motion benchmarking," *ACM Trans. Computer Systems*, vol. 21, no. 1, pp. 87–115, 2001.
[29] "What is the Linux Virtual Server?" http://www.linuxvirtualserver.org/, 2011.
[30] "HDFS Users Guide," http://hadoop.apache.org/docs/stable/hdfs_user_guide.html, 2011.
[31] R. Guerraoui and A. Schiper, "Software-based replication for fault tolerance," *IEEE Computer*, vol. 30, no. 4, pp. 38–74, 1997.
[32] A. Helal, A. Heddaya, and B. Bhar, *Replication Techniques in Distributed Systems*. Kluwer Academic Publishers, 1996.
[33] B. C. Neuman and T. TSó, "Kerberos: an authentication service for computer networks," *IEEE Commun.*, vol. 32, no. 9, pp. 33–38, 1994.
[34] "Gmail," http://www.gmail.com, 2012.
[35] "Sun Ray Overview, White Paper, Version 2," http://www.sun.com/sunray/whitepapers.html, 2004.
[36] R. A. Baratto, S. Potter, G. Su, and J. Nieh, "MobiDesk: mobile virtual desktop computing," in *Proc. 2004 International Conference on Mobile Computing and Networking.*
[37] G. A. Gibson and R. Y. Meter, "Network attached storage architecture," *Commun. ACM*, vol. 43, no. 11, pp. 37–45, 2000.
[38] D. R. Cheriton and W. Zwaenepoel, "The distributed V Kernel and its performance for diskless workstations," in *Proc. 1983 ACM Symposium on Operating Systems Principles.*
[39] B. Croft and J. Gilmore, "Bootstrap Protocol (BOOTP)," RFC 951, 1985.
[40] J. Satran, C. S. K. Meth, M. Chadalapaka, and E. Zeidner, "Internet Small Computer Systems Interface (iSCSI)," RFC 3720, 2004.
[41] "iBoot-Remote Boot Over iSCSI," http://www.haifa.il.ibm.com/projects/storage/iboot/index.html, 2012.
[42] "VMware Workstation," http://www.vmware.com/products/workstation/index.html, 2011.
[43] M. Satyanarayanan, "The evolution of coda," *ACM Trans. Computer Systems*, vol. 20, no. 2, pp. 85–124, 2002.
[44] R. Caceres, C. Carter, C. Narayanaswami, and M. Raghunath, "Reincarnating PCs with portable SoulPads," in *Proc. 2005 ACM/USENIX MobiSys*, pp. 65–78.
[45] R. Chandra, N. Zeldovich, C. Saputzakis, and M. S. Lam, "The collective: a cache-based systems management architecture," in *Proc. 2005 NSDI.*

Yuezhi Zhou obtained his Ph.D. degree in Computer Science and Technology from Tsinghua University, China in 2004 and is now working as an associate professor at the same department. He worked as a visiting scientist at the Computer Science Department in Carnegie Mellon University in 2005. His research interests include cloud computing, distributed systems, mobile devices and systems. He has published over 80 technical papers in international journals or conferences.

Yaoxue Zhang received received his Ph.D. degree in computer networking from Tohoku University, Japan in 1989. Then, he joined Department of Computer Science, Tsinghua University, China. Currently, he is a fellow of the Chinese Academy of Engineering and the president of Central South University University, China. His major research areas include computer networking, cloud computing, transparent computing, and active services. He has published over 200 technical papers in international journals and conferences, as well 9 monographs and textbooks.

Yinglian Xie received Ph.D. in Computer Science from Carnegie Mellon University, USA. She joined Microsoft Research Silicon Valley in 2006. Her general research interests are in security, privacy, distributed systems, and networking. Her recent work focuses on improving online service security.

Hui Zhang received a Ph.D. in computer science from the University of California, Berkeley. He is Co-Founder and Chief Scientist of Conviva and Professor of Computer Science at Carnegie Mellon University. His research interests include Internet Quality of Service (QoS), video streaming, network control, and Internet architecture. He is an ACM Fellow.

Laurence T. Yang received B.E in Computer Science from Tsinghua University, China and Ph.D. in Computer Science from University of Victoria, Canada. His is a professor in Computer Science at St. Francis Xavier University, Canada. His current research includes parallel and distributed computing, embedded and ubiquitous/pervasive computing. He has published many papers in various refereed journals, conference proceedings and book chapters in these areas (including around 100 international journal papers such as IEEE and ACM transactions).

Geyong Min received the Ph.D. degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. He is a Professor of Computer Science in the Department of Computing at the University of Bradford, United Kingdom. His research interests include Next Generation Internet, Wireless Communications, Multimedia Systems, Information Security, Ubiquitous Computing, Modeling and Performance Engineering.