Client Rendering Method for Desktop Virtualization Services

Su Min Jang, Won Hyuk Choi, and Won Young Kim

Cloud computing has recently become a significant technology trend in the IT field. Among the related technologies, desktop virtualization has been applied to various commercial applications since it provides many advantages, such as lower maintenance and operation costs and higher utilization. However, the existing solutions offer a very limited performance for 3D graphics applications. Therefore, we propose a novel method in which rendering commands are not executed at the host server but rather are delivered to the client through the network and are executed by the client's graphics device. This method prominently reduces server overhead and makes it possible to provide a stable service at low cost. The results of various experiments prove that the proposed method outperforms all existing solutions.

Keywords: Desktop virtualization, client rendering, cloud computing, software as a service.

I. Introduction

Desktop virtualization has recently had a major impact on the IT industry and is one of the representative technologies for cloud computing [1]. Desktop virtualization makes it possible to securely deliver desktops, applications, and data to many users through a network. Well-known solutions for desktop virtualization are Citrix XenDesktop [2], Microsoft RemoteFX [3], and rich client.

Citrix XenDesktop is a desktop virtualization solution that delivers a Windows desktop experience as an on-demand service to any user, anywhere. It is the only viable solution on the market today for delivering 3D professional graphics over WAN bandwidths. Figure 1 shows the workflow of the Citrix XenDesktop. All service applications are installed and run at the server level. The virtual desktop agent of the server sends captured screens of applications to various client terminals, such as phones, desktop computers, laptops, and personal digital assistants (PDAs). The captured screen images are delivered using the Independent Computing Architecture (ICA) protocol and H.264./MPEG-4 AVC virtual channel. In addition, user input data from mouse clicks and keystrokes from the client terminal are delivered to the server using the ICA protocol.

The merits of Citrix XenDesktop are that it allows applications to be accessed from most devices, and it covers both wired and wireless networks. In particular, the High-Definition User Experience (HDX) technology of Citrix XenDesktop enables it to optimize the performance of graphics-intensive 2D and 3D and media-rich applications. This capability offers the ability to leverage both software- and hardware-based rendering to assist with compression, performance, and network efficiency. However, it is difficult to continuously create original screens with image data captured from rich applications including 3D media if the requested client's display resolution is high or its screens are often shifted. Furthermore, because Citrix XenDesktop exclusively uses one graphics processing unit (GPU) of the server for processing 3D rendering commands for one user, it is extremely limited in providing multiple users with a virtual desktop service including 3D rendering applications [4].

Another well-known solution, Microsoft RemoteFX, enables the delivery of full Windows user experiences to various client devices using a virtual desktop infrastructure. RemoteFX is integrated with the Remote Desktop Protocol, which enables

Manuscript received May 16, 2012; revised Oct. 19, 2012; accepted Oct. 30, 2012.

This work was supported by the Technology Innovation Program [10035185, Development of a Separated Software Execution Technology for Software Service] funded by the Ministry of Knowledge Economy (MKE, Korea).

Su Min Jang (+82 42 860 0724, jsm@etri.re.kr), Won Hyuk Choi (whchoi@etri.re.kr), and Won Young Kim (wykim@etri.re.kr) are with the Software Research Laboratory, ETRI, Daejeon, Rep. of Korea.

http://dx.doi.org/10.4218/etrij.13.0212.0213

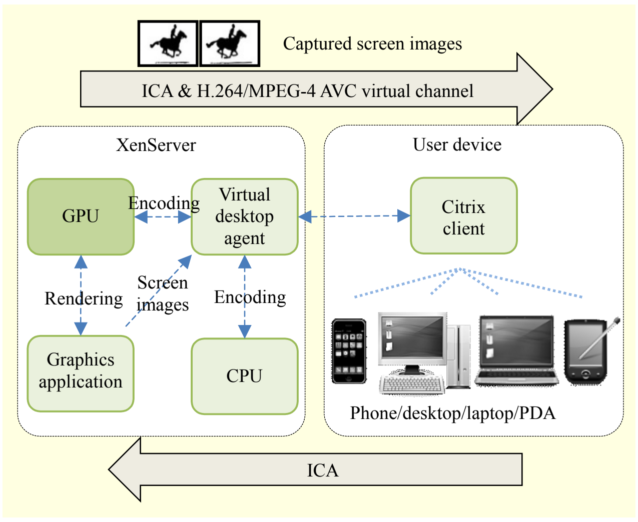

Fig. 1. Workflow of Citrix XenDesktop.

shared encryption, authentication, management, and device support. Host-side rendering of RemoteFX allows graphics to be rendered on the host device instead of on the client device. It enables support for all graphics types by sending highly compressed bitmap images to the endpoint device in an adaptive manner. This also allows the applications to run at full speed on the host computer by taking advantage of the GPU and central processing unit (CPU). GPU virtualization in RemoteFX allows multiple virtual desktops to share a single GPU on a Hyper-V server. GPU sharing leads to a lower performance as the number of users increases.

Another solution for application services is using a rich client. The server of a rich client sends executing source codes and all related data to the client, and the client then executes the streamed source codes with its own resources. There is no problem, as the rich client solution can continuously create original screens. However, this solution cannot be executed immediately and is very weak in terms of data security, as all related data and codes are transferred from the server to the client.

II. Client Rendering Method

We use a client rendering (CR) method to solve the problems addressed in existing solutions. In the CR method, the host rendering commands are delivered to the client through the network and are executed by the client's graphics device. The proposed method has two main goals: i) original screens of the hosted application can be continuously seen at the remote client, regardless of the client's display resolution, and ii) an efficient service for multiple users is provided without server overhead, such as an exclusive use of the GPU.

The CR method separates all tasks for the 3D graphics

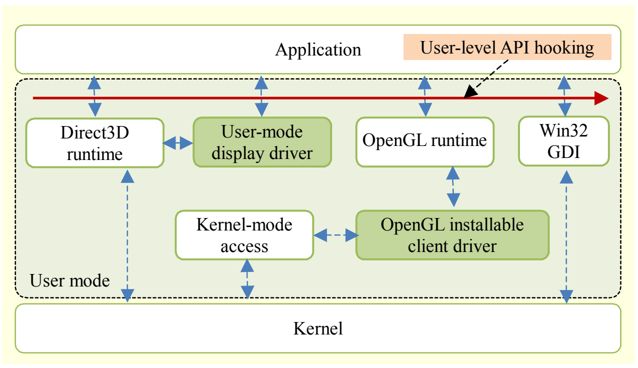

Fig. 2. User-level API hooking of CR method.

application execution into graphics and nongraphics tasks at the host server. The graphics tasks are delivered and processed at the client, and nongraphics tasks are processed at the server. In this sense, the idea of the CR method is quite similar to the remote GPU of GVirtuS, GVirm, and rCUDA [5]. The primary difference among them and the CR method is that the CR method is not CUDA. In the proposed system, the server does not send captured screen images to the client but delivers rendering commands to the client instead. This mechanism can dramatically reduce the amount of data transferred between the client and server. In addition, it can solve the problem in that the server exclusively uses its own GPU, as the graphics tasks are computed by the client's GPU. The limitation of our system is that it relies on network bandwidth among servers and clients and the GPU device performance of a client.

To intercept rendering commands from the host application execution, the proposed method uses user-level application programming interface (API) hooking, as shown in Fig. 2. The hook mechanism is implemented using wrapper libraries [6]. When the application of the server calls rendering commands, the wrapper libraries are executed instead of the original libraries.

Figure 3 shows the overall workflow of the client rendering used in our method. When rendering commands are generated from the host application, the rendering commands are delivered to the client by the wrapper libraries of the server, and then the CR client agent executes the delivered rendering commands with wrapper library handlers of the client. As a result, the GPU of the server is not used, and the rendering commands use the GPU of the client.

The virtual desktop agent of the CR server is composed of the rendering command hooker and the virtual object manager. The role of the rendering command hooker is to hook rendering commands or related messages from executing graphics applications and deliver the hooked commands to the client. The virtual object manager maintains such objects as handles and interface instances generated from the host

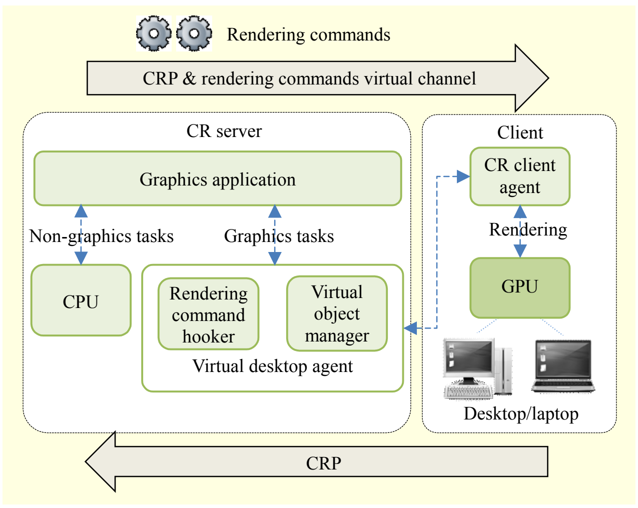

Fig. 3. Workflow of client 3D rendering.

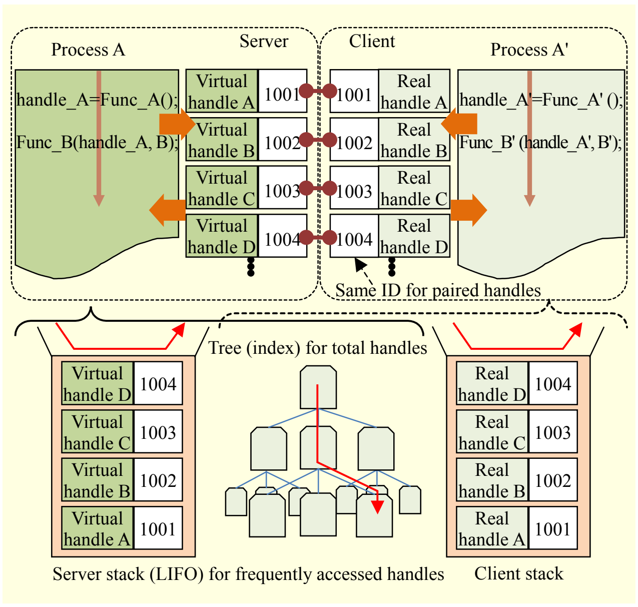

Fig. 4. Management structure and index of paired objects.

application execution. These rendering commands are delivered through virtual channels, and additional information, such as mouse and keyboard inputs, is sent by the CR protocol (CRP).

To service our system, two processes should work closely, almost as one process. One process is executed for nongraphics tasks at the server, and another process is executed to render tasks at the client. The two processes individually generate objects, such as the windows handle, file handle, and data pointer, during execution. The generated objects should be managed as pairs. Figure 4 shows the management structure and index of the paired objects. The identification (ID) is used as a link for making a pair using a virtual handle of the server

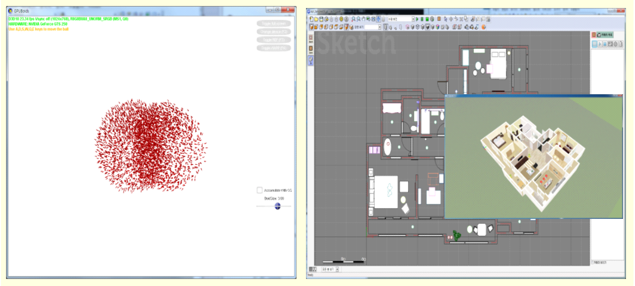

Fig. 5. Screenshots of test programs.

and the real handle of the client. To quickly access the paired object, we use a stack (last in, first out [LIFO]) and tree index. Frequently referenced objects are accessed by the stack, and the remaining objects are accessed by the tree index.

In our implementation, there are many side effects, including a priority change in the processes and the private usage of system resources among the processes, because two or more processes used in the proposed method are executed simultaneously at the server. To solve these problems, various virtualization modules for our service are developed. The core parts of the virtualizations are the Windows system resource, register, and graphics device.

III. Performance Evaluation

To show the superiority of the proposed method, we compare it with RemoteFX and HDX. To evaluate the performance of each server for a 3D graphics application service, we choose two 3D programs, GPUBoids and ArchiSpace LT, as shown in Fig. 5. GPUBoids is a sample program that creates the illusion of flocks of birds or bugs swarming around a light or a fallen comrade. ArchiSpace LT is a commercial application used for generating and displaying virtual buildings or architectural drawings, using 3D objects.

The clients and servers are connected via 100-Mbps Ethernet. Table 1 shows the specifications of the clients and servers. It shows that the CR server is set with the lowest-cost GPU ($450), unlike those of RemoteFX and HDX.

Figure 6 shows an evaluation of a single client (1:1) on each server (RemoteFX, HDX, CR), where the program used is GPUBoids. The evaluated factors are frames per second (FPS), CPU/GPU usage, and the amount of network transmission. CR shows a more stable FPS than HDX and RemoteFX, as illustrated in Fig. 6(a). On the other hand, RemoteFX shows a large gap between the maximum and minimum FPS, which causes an unstable display. Figures 6(b) and 6(c) show that the CPU and GPU usages of HDX and RemoteFX are very high, but the CR rarely uses them since the rendering commands are

Table 1. Specification of clients and servers.

	CPU / Memory	GPU
Client PC1 to 5	Intel Core i7 3.4 GHz, Memory 8 G	NVIDIA GeForce GTX 570
Client PC6	Intel Core i7 2.8 GHz, Memory 4 G	NVIDIA GeForce GTS 250
Citrix HDX server	Intel Core i7 3.4 GHz, Memory 16 G	NVIDIA Quadro FX3800 4 GB * 4 ($5,000)
RemoteFX server	Intel Xeon X5650 2.6 GHz Memory 20 G	NVIDIA Quadro 6000 8 GB ($5,500)
CR server	Intel Core i7 3.2 GHz, Memory 12 G	NVIDIA GeForce GTX 480 ($450)

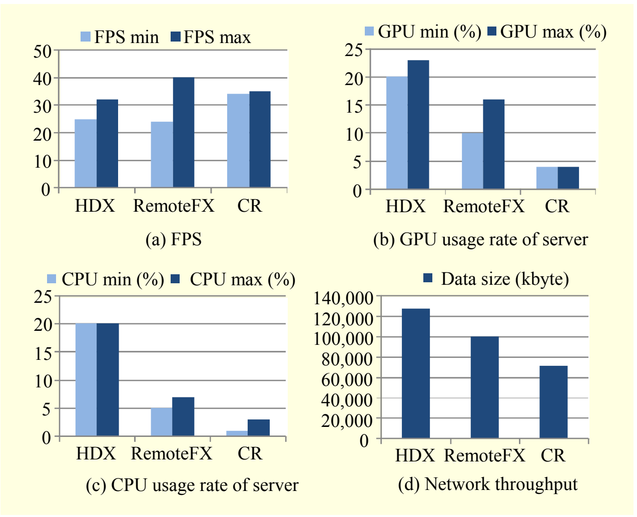

Fig. 6. Evaluations on single client of GPUBoids.

executed not at the server but at the client. Figure 6(d) shows that the CR significantly declines transferred packets at the network compared with other solutions and that, moreover, the data size is smaller.

Figure 7 shows an evaluation of multiple clients (1:*N*) on RemoteFX and CR servers with ArchiSpace LT. Whenever the number of clients is increased, all clients of RemoteFX show the same FPS, which decreases significantly, as shown in Fig. 7(a), as the RemoteFX server executes all of the rendering and nongraphics tasks. However, the FPS of a CR client is independent of the client count, as shown in Fig. 7(b), since each client uses its own GPU.

IV. Conclusion

Among existing cloud computing issues, desktop virtualization has emerged as a more convenient and economical way of using software. To improve the quality of 3D graphics application services, we devised the CR method, which is more effective for services with multiple users. In a performance experiment, the proposed method showed desirable scalability and obtained a more stable frame rate than the existing desktop virtualization solutions. In the near future, the proposed service system will be applied to various real applications and will be further optimized.

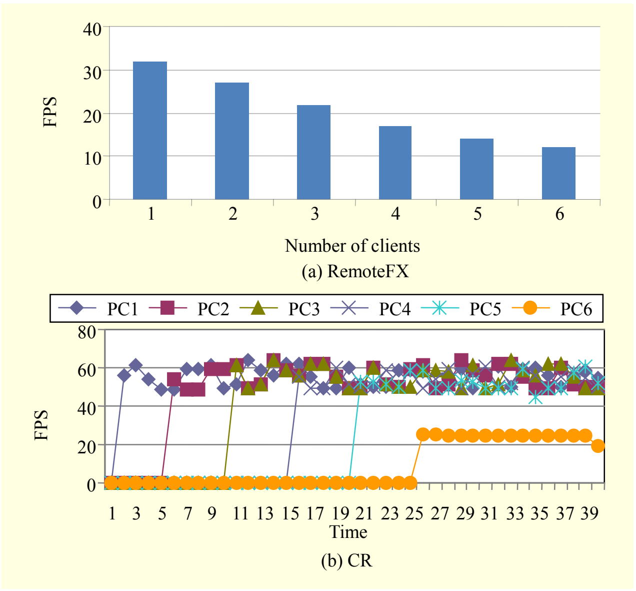

Fig. 7. Evaluation of multiple clients of ArchiSpace LT.

References

[1] C.W. Yoon et al., "Dynamic Collaborative Cloud Service Platform: Opportunities and Challenges," *ETRI J.*, vol. 32, no. 4, Aug. 2010, pp. 634-637.
[2] HDX of Citrix XenDesktop. http://hdx.citrix.com/ hdx
[3] RemoteFX of Microsoft. http://technet.microsoft.com/ en-us/library/ ff817578%28WS.10%29.aspx
[4] L. Shi, H. Chen, and J. Sun, "vCUDA: GPU Accelerated High Performance Computing in Virtual Machines," *IEEE Int. Symp. Parallel Distrib. Process.*, 2009, pp. 1-11.
[5] J. Duato et al., Performance of CUDA Virtualized Remote GPUs in High Performance Clusters" *IEEE Int. Conf. Parallel Process.*, 2011, pp. 365-374.
[6] M. Dowty and J. Sugerman, "GPU Virtualization on VMware's Hosted I/O Architecture," *ACM SIGOPS Operating Syst. Rev.*, vol. 43, no. 3, 2009, pp. 73-82.