Benchmarking the Performance of XenDesktop Virtual DeskTop Infrastructure (VDI) Platform

Shie-Yuan Wang

Department of Computer Science
National Chiao Tung University, Taiwan
Email: shieyuan@cs.nctu.edu.tw

Wen-Jhe Chang

Department of Computer Science
National Chiao Tung University, Taiwan
Email: ethan-shy@hotmail.com

***Abstract*—The recent advances in portable devices and the trends to move a user's desktop to cloud environments have changed how people use traditional computers today. Several companies have developed the "Virtual Desktop Infrastructure" (VDI) technology for this trend. By this technology, people need not use a traditional PC with a high clock-rate CPU and a large storage device to run heavy tasks. Instead, they can use a "thin-client" and a network connection to run these heavy tasks on a remote VDI server. A VDI user can perform these tasks in a virtual desktop run by a virtual machine (VM) on the VDI server. During operations, the screen image of the virtual desktop will be delivered to the screen of the thin-client. The VDI technology offers many advantages. However, it may increase the perceived delays when a VDI user operates a virtual desktop. These delays may be caused by bad network conditions or by overloading conditions on the VDI server. In this paper, we developed a VDI performance benchmarking tool and used it to measure the perceived delays when the XenDesktop VDI platform is used under various network conditions and overloading conditions on the VDI server. The EstiNet network simulator and emulator was used to create various network conditions for benchmarking measurements.**

Keywords–EstiNet; VDI; VM.

I. INTRODUCTION

Recently, the VDI technology has become more and more popular due to the availability of high-speed network accesses and advances in portable devices. In this architecture, users operate their virtualized desktops on a remote VDI server rather than operate a real desktop on their local machines. When they operate from their local machines, all of the programs, applications, processes, and data are run and kept on the VDI server. In this way, a user can run the same operating system and execute the same applications and access the same data from any machine via a network connection. Because this computing model has great potential to save cost and increase data security, many companies such as Citrix [1], Microsoft [2], Oracle [3], and VMware [4] have developed their own VDI technologies.

VDI offers many advantages but also comes with several challenges. Nowadays, many users still hesitate to adopt the VDI technology. One major concern is that using VDI may suffer a much higher delay than using a desktop computer and it is difficult for the VDI user to find out the causes. VDI is a client-server architecture. When VDI users operate their virtual desktops through a network, they must compete with other VDI users for the network bandwidth and the various resources on the VDI server. As a result, many factors can cause the VDI users to more easily experience long delays when operating their remote virtual desktops. Besides, because a virtual desktop is run by a VM on a remote VDI server, it is more difficult for the VDI users to find out what factor is causing the long operation delays. For example, either a high CPU usage on the VDI server or a long round-trip network delay between the VDI user and the VDI server can cause the user to experience large delays. However, the VDI user does not know which factor is causing this delay and thus does not know whether he should contact the VDI cloud service provider or the Internet service provider to report and complain the bad performance problems.

In this paper, we develop a performance benchmarking tool to measure the delay (i.e., the screen response time) of Citrix's XenDesktop VDI platform under five important conditions. The first two conditions: (1) large link delay and (2) high packet loss rate, are about the quality of the network. The other three conditions: (3) high CPU usage, (4) insufficient memory allocation per VM, and (5) high disk usage, are about the VDI server resource usage conditions. According to our measured results, each of these five factors can cause a large delay when a VDI user performs tasks on his virtual desktop. Our results reveal that these factors affect the delay differently. Due to the paper length limitation, in this paper we can only present the performance benchmarking results. In our future paper, we will present how we use the delay features of these factors to develop a VDI performance diagnostic tool that can accurately tell a VDI user which factor(s) is (are) causing the long perceived delay.

The rest of the paper is organized as follows. In Section II, we present related work on virtual desktop infrastructure. In Section III, we present the implementation of our performance benchmarking tool. Experimental setups are presented in Section IV and various experimental results are presented in Section V. Finally, we conclude the paper in Section VI.

II. RELATED WORK

With the advances of desktop virtualization and thin-client devices, there have been some work on performance evaluation of VDI. The closest work to ours is DeskBench [5], which captures the screen and records and playbacks keyboard and mouse events on the client side. The other close work is VNCPlay [6], which is also based on matching screen and recording and playback of keyboard and mouse events. Another similar work is Slow Motion Benchmarking [7]. It captures the network traffic exchanged between the client and server and replays the network traffic later in slow motion. The

Copyright (c) IARIA, 2015. ISBN: 978-1-61208-398-8

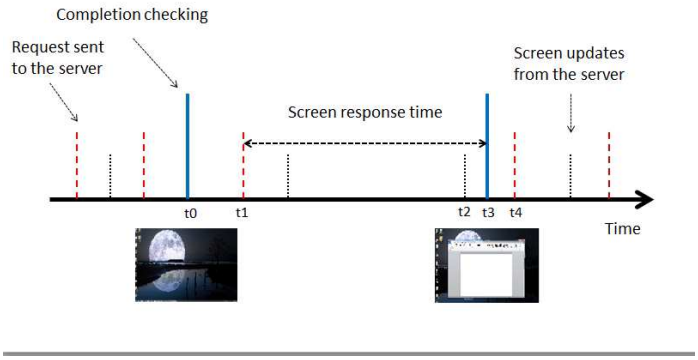


Figure 1. Interaction between the VDI client and server

authors in [8] proposed the VDBench toolkit to measure thin-client performances under server load and network conditions. In [9], the authors focused on benchmarking the audio transmission performance when virtual desktop platform is used, as this is the topic that a Telcom operator would concern.

In our paper, we use a similar approach to these work to measure the screen response time. However, we use the Citrix's XenDesktop VDI platform as the performance study target while most of these previous work studied the open-source VNC protocol and Microsoft's Remote Desktop Protocol (RDP). Compared with the work done in [8], the authors in [8] used VMware ESXi 4.0 as the VDI server while we used Citrix's XenDesktop VDI platform as the VDI server. XenDesktop uses a patented highly-efficient HDX technology as the VDI protocol between its VDI client and server. According to our measurements and comparisons, we found that HDX performs much better than VNC and RDP because the perceived VDI delay when HDX is used is much less than those when VNC and RDP are used. Beside the differences in the tested VDI technologies, in this paper we focus on the XenDesktop delay performances under various network conditions and overloading conditions on the VDI server. Most of these conditions are not studied in these previous work.

III. Implementation of the Performance Benchmarking Tool

The screen response time of a VDI user's action is the time between when the user clicks the mouse or does the keyboard input and the time when the corresponding screen shows up completely on the VDI user's device. The response time measuring process is depicted in Figure 1. The horizontal line represents the time axis. Vertical dashed lines represent the requests sent from the client to the server. Short vertical dotted lines represent screen updates arriving from the sever. High vertical solid lines represent when our tool compares the current screen image with the expected one. Assume that the user clicks the mouse left button to execute Microsoft Word application at time t1. Further assume that the server sends the new screen of executing Microsoft Word to the client at t2 and our tool detects that a match occurs at t3. For this example case, the time between when the user sends a request to the server and the time when the corresponding screen shows up is t3 - t1. This is the VDI delay performance measured and reported in this paper.

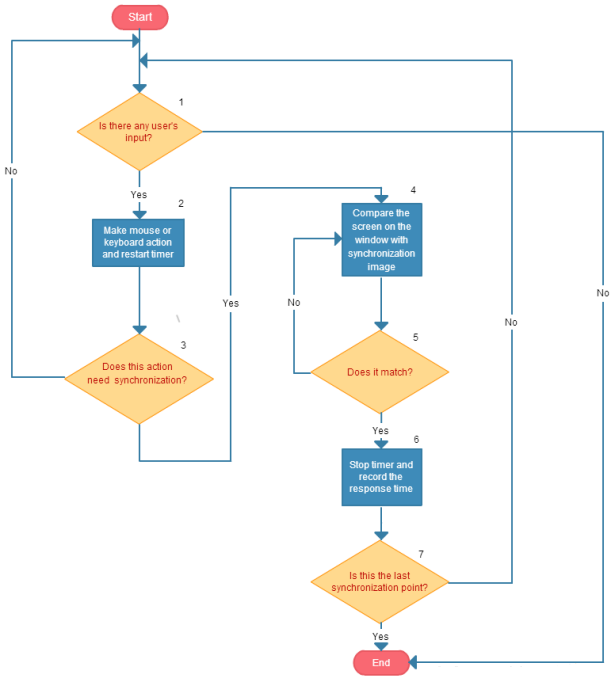


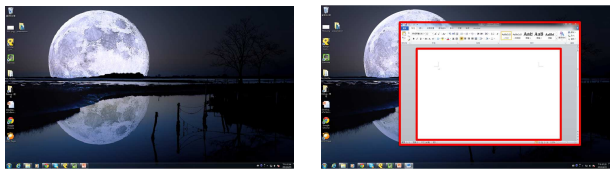
Figure 2. Flow chart of measuring screen response time

The flow chart illustrating how we automatically generate a user's action and measure its screen response time is shown in Figure 2.

1 First, check if there is any user's input occurring.
2 If input occurring, our tool uses the WIN32 API to capture and replay the user's actions of mouse and keyboard.
3 If an action needs to synchronize with the screen (that is, after executing the action, a new screen must show up before the next action can be executed), our tool will start a timer to record the response time. (Note that some actions need not synchronize with the screen. In such a case, the next action in the replay list can be executed immediately without waiting for the new screen to show up.)
4 For an action that needs screen synchronization, after the timer is started, our tool will periodically check whether the expected screen image has arrived from the server by comparing the current screen image of an area with the new screen image of the same area.
5 Detect if there is any screen match occurring.
6 If a screen match is detected, our tool will stop the timer and record the response time of this action. Then, the tool will execute the next action.
7 If there are no more actions to send to the server, the tool will stop and show the response time of all executed actions.

To make the measured response time accurate, when the timer is turned on, the interval between two successive screen image comparisons must be small enough. This means that each image comparison must be finished as soon as possible. To do so, our tool intelligently compares only the new and old images of the area where its screen image is expected to change without comparing the new and old screen images of the whole screen. As an example, Figure 3(a) shows the whole

Copyright (c) IARIA, 2015. ISBN: 978-1-61208-398-8



(a) Screen before an action is sent (b) Screen after an action is completed

Figure 3. Screen before and after a user sends a request

screen before an action is sent to the server while Figure 3(b) shows that a window is popped up after the action is sent to the server and completed. We see that this action only changes an area of the screen. As a result, our tool only needs to compare an area of the whole screen to speed up the image comparison operation. With this improvement, our tool can finish the image comparison in 100 microseconds. In our implementation, when the measuring timer is turned on, our tool will perform the image comparison every 100 microseconds.

The three types of actions that most users will issue when performing tasks on a desktop are listed in Table I. Because these types of actions will be executed very frequently, the VDI delays of these actions are important performance metrics that can be used to judge the delay quality experienced by a VDI user.

TABLE I. TYPES OF ACTIONS ISSUED ON THE CLIENT TO THE VDI SERVER

Action Types	Explanation
Opening and closing Microsoft Word	The client sends the mouse click action to the server to open/close the Microsoft Word application. It then measures the screen response time for the application to completely open up/close down its window.
Keyboard input	The client sends the keyboard input action to type some few words in Microsoft Word. It then measures the screen response time of the words showing up on the screen.
Compressing files	The client sends the mouse click action to perform a compress files operation. It then measures the screen response time of the compressing application finishing its jobs completely.

IV. EXPERIMENTAL SETUP

In this paper, we use Citrix's XenDesktop as our VDI platform and use the EstiNet network simulator and emulator [10]–[12] to create various network conditions between the VDI client and server. The setup of our experiments is presented in Figure 4. The lower part of Figure 4 shows that in the real world we use Asus RS500A-E6 for our VDI server, which is equipped with two AMD CPUs (each with 12 cores operating at 1.9 GHZ) and 48 Gigabyte memory. The VDI server runs XenServer (version 5.6) to host up to 32 VMs each running Windows 7 operating system. It also runs the XenDesktop controller to manage these Windows 7 virtual desktops. The VDI client runs XenDesktop receiver and our tool on a PC that is equipped with an Intel CPU (3.4 GHZ dual core). We run EstiNet on another computer as a network emulator between the VDI client and server. It connects to both the VDI client and server to intercept their exchanged packets to vary the delay and packet loss rates experienced by these packets.

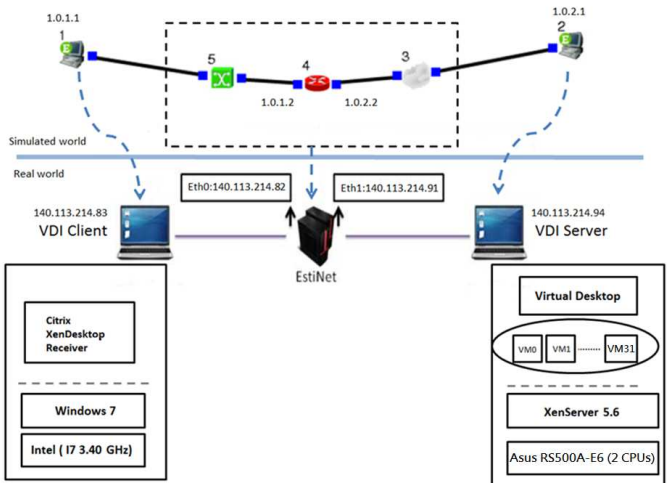


Figure 4. Experimental Environment

The upper part of Figure 4 shows how the tested network in the real world is represented and simulated in the EstiNet network emulator. In the simulated network, node 1 with the IP address 1.0.1.1 represents the VDI client while node 2 with the IP address IP 1.0.2.1 represents the VDI server. Because the packets exchanged between node 1 and node 2 will go through node 3, which is configured to add certain delays to these packets or drop these packets at a certain rate, EstiNet can easily vary the network conditions between the VDI client and server in the real world.

V. EXPERIMENTAL RESULTS

We measure the response time of opening and closing Microsoft Word, keyboard input, and compressing files under different server loadings, link delays, and packet loss rates. We vary the server's total CPU usage by running 0-32 VMs on the server and let each VM run a CPU-bound job, which consumes about one CPU core. We vary the size of the memory allocated to each VM from 4 GB down to 1 GB and execute a program on each VM to purposely consume about 1 GB memory. Doing so is to test how important the usable memory space is to the VDI delay of an action. We also vary the disk usage on the server by running 0-10 VMs on the server and let each VM compress files to generate about 6 MB/s disk read/write load per VM. We found that 10 VMs are enough to generate heavy loads on the disk. In the following figures, each data point is the average of 100 runs of the measurements of the same type of action performed under the same settings.

Figures 5 - 7 show the response time of opening and closing Microsoft Word under different server loading conditions. A first finding is that the delay of the "Word Close" action is much less than the delay of the "Word Open" action, and its delay remains low and stable under high server loading conditions. These results suggest that the "Word Close" action is a light-weight operation. In contrast, from Figure 5 we see that when more and more CPU-bound VMs are competing for the shared CPU resource, which results in insufficient CPU resource allocation for the VM executing the "Word Open" action, the delay of this action goes up quickly. From Figure 6, we see that when the allocated memory to a VM is less than 1.5 GB due to the competition among more and more VMs, the

Copyright (c) IARIA, 2015. ISBN: 978-1-61208-398-8

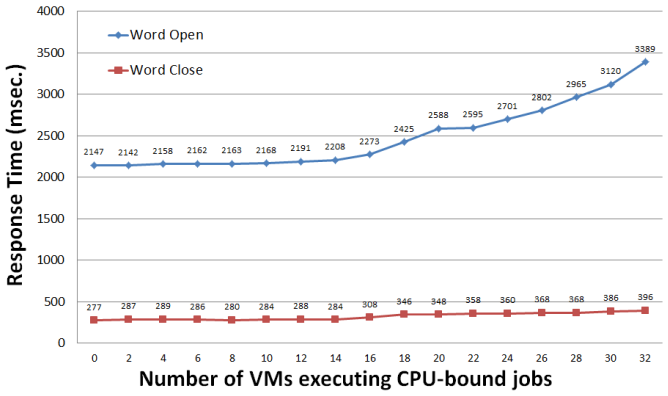


Figure 5. Screen response time of opening and closing Microsoft Word under different numbers of VMs each executing CPU-bound jobs

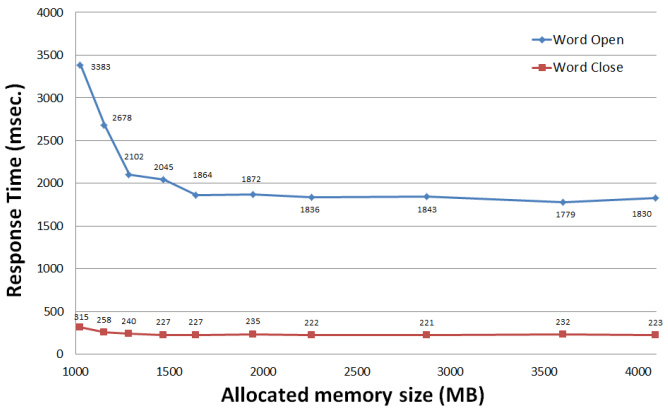


Figure 6. Screen response time of opening and closing Microsoft Word under different allocated memory sizes per VM

delay of the "Word Open" action goes up very quickly. This phenomenon is due to the "thrashing" effect of the operating system, which refers to the situation when the hard disk space is used as memory space due to insufficient memory space allocated for executing an application.) From this phenomenon, we see that the "Word Open" action not only requires much CPU resource, it also requires much memory resource for quick response. Figure 7 shows that even with high disk usages (which is caused by compressing applications executing many disk I/O operations), the delay of "Word Open" action does not increase much. This suggests that the "Word Open" action does not require much disk throughput resource. In summary, our results show that the VDI delay of "Word Open" action increases under high CPU usage or high memory usage but remains about the same under high disk usage.

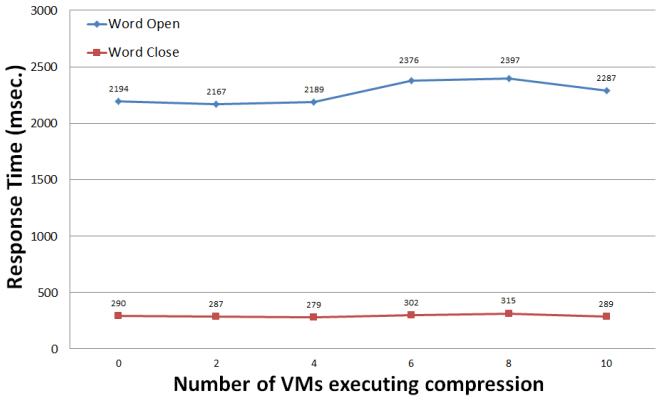


Figure 7. Screen response time of opening and closing Microsoft Word under different numbers of VMs each executing compression

The delays of the "Word Open" action under different link delays are shown in Figure 8. As expected, the link delay linearly affects the delay of the action because the VDI client must wait for the link delay to elapse before the screen update can arrive. Figure 9 shows the delay of the "Word Open" action under different packet loss rates. We see that the packet loss rate increases the delay non-linearly and when the packet loss rate exceeds 12%, the delay starts to increase dramatically. This phenomenon can be explained by the fact that the transport protocol used by XenDesktop's VDI technology is TCP and TCP throughput is very sensitive to the packet loss rate due to its conservative congestion control algorithm.

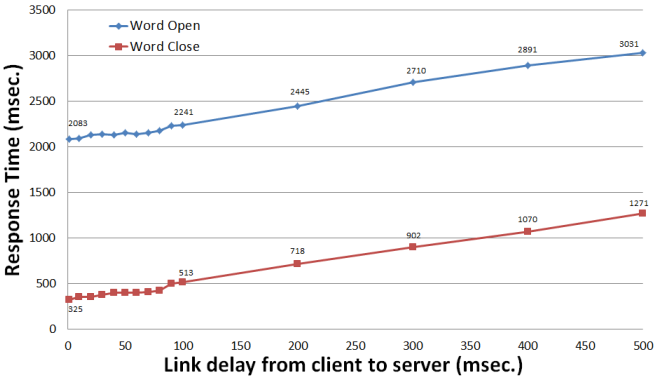


Figure 8. Screen response time of opening and closing Microsoft Word under different link delays

Figures 10 - 14 show the response time of keyboard input under different server loading and network conditions. Figure 10 shows that the "Keyboard input" action requires some CPU resource and its delay increases by a small amount of 30 milliseconds when more and more VMs are competing for the shared CPU resource. Figure 11 shows that the delay of the "Keyboard input" action remains about the same under different allocated memory sizes unless the size of the allocation drops to 1 GB, where the thrashing effect starts to begin. Figure 12 shows that the delay of the "Keyboard input" action does not increase as the disk usage increases. This phenomenon is expected as the processing of a keyboard input on the VDI server does not need to use any disk I/O operation. As a result, the delay of the "Keyboard input" action has no relationship with the current disk usage on the VDI server. From these three figures, we see that the maximum and minimum delays measured for the "Keyboard input" action under different server loading conditions only differ by about 30 milliseconds. This difference is quite small and the VDI user will not notice such a difference.

However, Figure 13 and Figure 14 show that the network conditions can dramatically increase the delay of keyboard input. As expected, the link delay between the VDI client

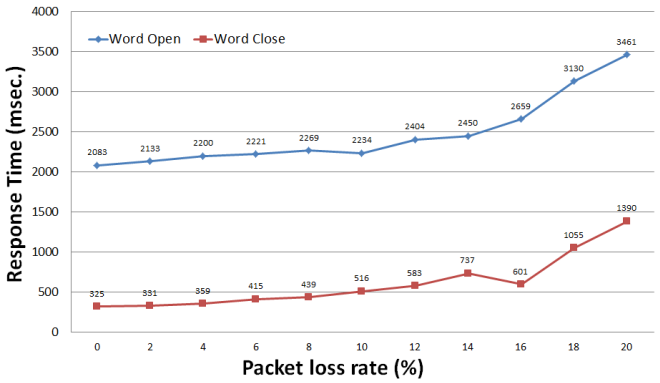


Figure 9. Screen response time of opening and closing Microsoft Word under different packet loss rates

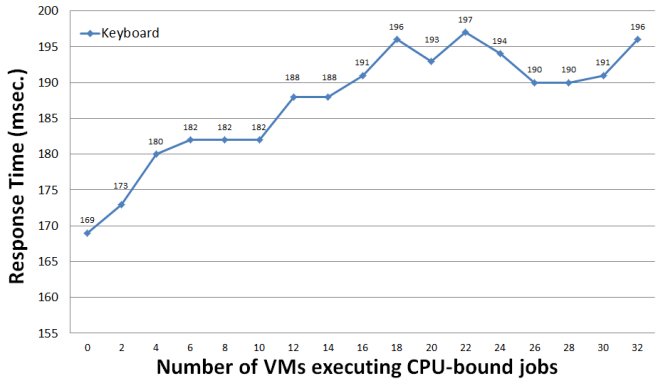


Figure 10. Screen response time of keyboard input under different numbers of VMs each executing CPU-bound jobs

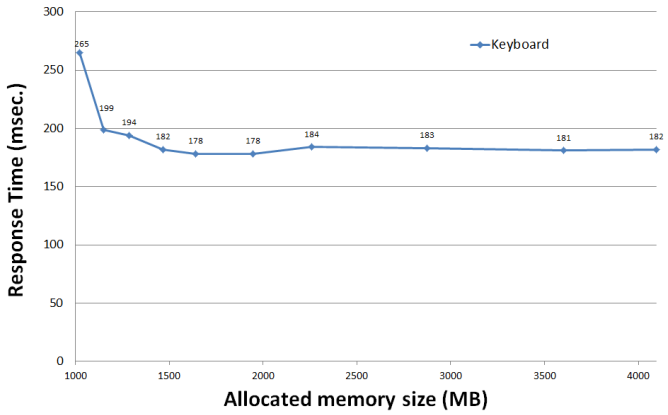


Figure 11. Screen response time of keyboard input under different allocated memory sizes per VM

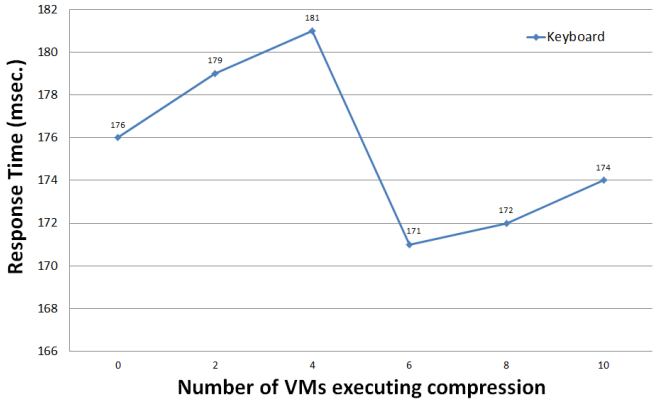


Figure 12. Screen response time of keyboard input under different numbers of VMs each executing compression

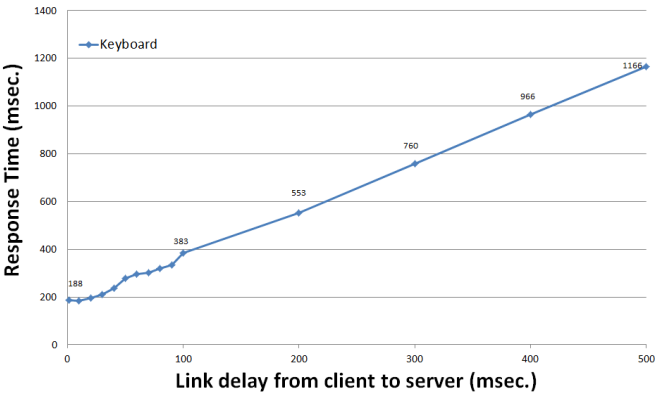


Figure 13. Screen response time of keyboard input under different link delays

and server linearly increases the delay of keyboard input. This is reasonable as the data amount that needs to be exchanged between the VDI client and server for a keyboard input is only a few bytes. Thus, the processing time required for the exchanged data on the VDI server is little and constitutes only a very tiny portion of the delay. That is, most of the delay comes from the link delay between the VDI client and server. Regarding the packet loss rate, we see that it also affects the delay of keyboard input significantly. We see that when the packet loss rate exceeds 18%, the delay abruptly jumps to 1.078 seconds, which is very noticable and annoying for the VDI user.

In summary, we found that the delay of the "Keyboard input" action generally is not affected by the server loading conditions but will be affected by large link delays and high packet loss rates in the network.

Figure 15 - 17 show the response time of compressing files under different server loading conditions. We do not measure the response time of compressing files under network conditions. This is because the time required to finish compressing files is too large (e.g., above 100,000 ms) compared to the tested link delays, whose maximum value is 500 ms. For such a situation, the link delay affects the delay of compressing files very minimally. In addition, because the "compressing files" action compresses the files on the VDI server without the need to transfer any file from the VDI client to the server, the packet loss rate does not affect the delay of the "compressing files" action at all. As a result, we do not study its effects on the delay of the "compressing files" action.

As shown in Figure 15, the "compressing files" action requires much CPU resource. This is evident because the figure shows that when more and more CPU-bound VMs are competing for the CPU resource, the delay of the "compressing files" action increases rapidly. Figure 16 shows that the "compressing

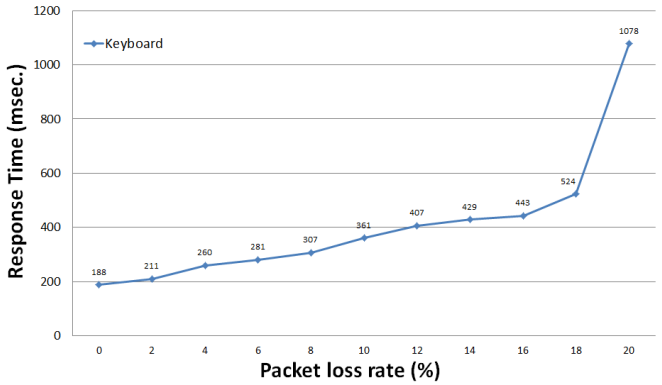


Figure 14. Screen response time of keyboard input under different packet loss rates

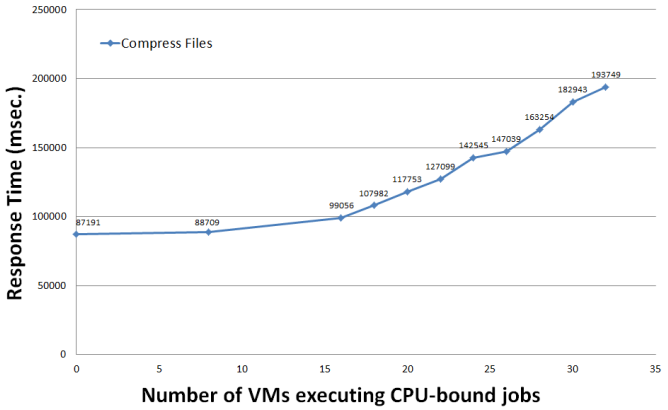


Figure 15. Screen response time of compressing files under different numbers of VMs each executing CPU-bound jobs

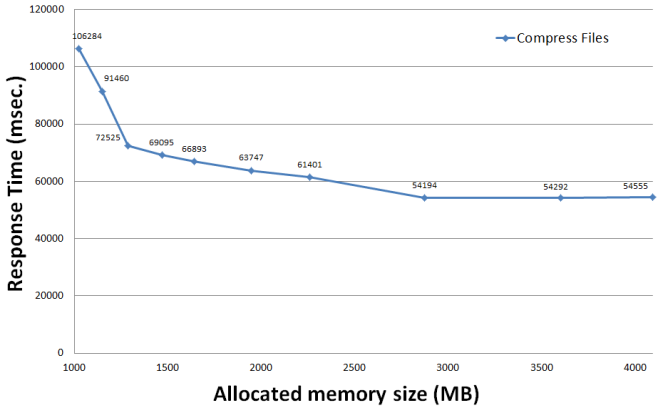


Figure 16. Screen response time of compressing files under different allocated memory sizes per VM

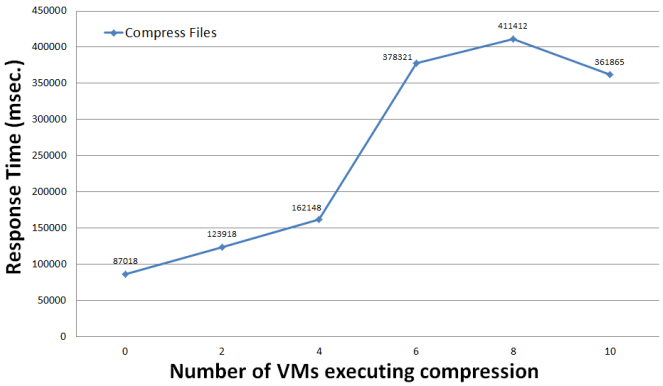


Figure 17. Screen response time of compressing files under different numbers of VMs each executing compression

files" action also requires much memory resource to finish the job quickly. Comparing Figure 16 with Figure 6 and Figure 11, we see that the "compressing files" action requires much more memory than the "Word Open" action or the "Keyboard input" action because its delay starts to go up when the allocated memory size is less than 3 GB while the delays of the other two actions go up only when the allocated memory sizes are less than 1.7 GB and 1.3 GB, respectively. Figure 17 shows that the "compressing files" action also requires much disk throughput resource because when more and more VMs are compressing files to compete for the disk throughput resource, the delay of the "compressing files" action goes up rapidly. From these figures, we see that the delay of the "compressing files" action is affected by high CPU usage, insufficient memory allocation, and high disk usage.

VI. Conclusion

In this paper, we developed a VDI performance benchmarking tool and used it to study the delay performance of the XenDesktop VDI platform under various network conditions and server loading conditions. Using the EstiNet network emulator to create various network conditions, our study shows that the following factors can increase the experienced delay of XenDesktop VDI significantly: the link delay and the network packet loss rate between the VDI client and server, the CPU utilization of the VDI server, the disk read/write load of the VDI server, and the size of the memory allocated to a VM running the virtual desktop. Our measurement results reveal that these factors affect the experienced delay of XenDesktop VDI differently. The delay features of these factors can be used to judge what factor(s) is (are) causing the delay when a user operates a virtual desktop. Based on these unique delay features, a diagnostic tool can be developed to help network service providers and cloud service providers to jointly identify the real causes for large experienced VDI delays. In the future, we will extend our work to study the perceived VDI delays when the VM that runs a virtual desktop migrates from one physical server to another. This topic is important as a VM may frequently migrate in a cloud for load balancing purposes.

References

[1] "XenDesktop Product Information," URL: http://www.citrix.com [accessed: 2015-03-04].

[2] "Virtual Desktop Infrastructure in Windows Server 2008," URL: http://www.microsoft.com [accessed: 2015-03-04].

[3] "Oracle Virtual Desktop Infrastructure Product Information," URL: http://www.oracle.com [accessed: 2015-03-04].

[4] "VMWare EMC." URL: http://www.vmware.com [accessed: 2015-03-04].

[5] J. Rhee, A. Kochut, and K. Beaty, "Deskbench: flexible virtual desktop benchmarking toolkit," in Integrated Network Management, 2009. IM'09. IFIP/IEEE International Symposium on. IEEE, 2009, pp. 622–629.

Copyright (c) IARIA, 2015. ISBN: 978-1-61208-398-8

[6] N. Zeldovich and R. Chandra, “Interactive performance measurement with vncplay.” in USENIX Annual Technical Conference, FREENIX Track, 2005, pp. 189–198.

[7] J. Nieh, S. J. Yang, and N. Novik, “Measuring thin-client performance using slow-motion benchmarking,” ACM Transactions on Computer Systems (TOCS), vol. 21, no. 1, 2003, pp. 87–115.

[8] A. Berryman, P. Calyam, M. Honigford, and A. M. Lai, “Vdbench: A benchmarking toolkit for thin-client based virtual desktop environments,” in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. IEEE, 2010, pp. 480–487.

[9] F. Wang, Y. Liu, B. Lei, and J. Li, “Benchmark driven virtual desktop planning: A case study from telecom operator,” in Proceedings of the 2012 International Conference on Cloud and Service Computing. IEEE Computer Society, 2012, pp. 204–211.

[10] S.-Y. Wang, C.-L. Chou, and C.-M. Yang, “EstiNet OpenFlow network simulator and emulator,” Communications Magazine, IEEE, vol. 51, no. 9, 2013, pp. 110–117.

[11] S.-Y. Wang, “Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet,” in Proceedings of the 2014 IEEE International Symposium on Computers and Communication (ISCC). IEEE, 2014, pp. 1–6.

[12] “EstiNet Network Simulator and Emulator,,” URL: http://www.estinet.com [accessed: 2015-03-04].

Copyright (c) IARIA, 2015. ISBN: 978-1-61208-398-8