Graphic Acceleration Mechanism for Multiple Desktop System Based on Virtualization Technology

He Li, Hai Jin, Xiaofei Liao
Services Computing Technology and System Lab
Cluster and Grid Computing Lab
School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, 430074, China
hjin@hust.edu.cn

***Abstract*—Multiple-desktop environment system (MPDS) based on virtualization technology focuses on improving user experiences of desktop applications. However, the current graphic performance of MPDS cannot satisfy the various desktop applications. Therefore, enhancing the graphic performance of MPDS is a key issue in virtual desktop environment. Leverage existing technical conditions, we propose a graphic acceleration mechanism to solve the problem. This graphic acceleration mechanism of MPDS provides better graphic performance than existing virtual machine systems. Using the graphic acceleration mechanism, virtual machines without hardware rendering capability in MPDS could execute complex graphic applications and display these with good effect. In the analysis, we have the conclusion that the graphic performance with the graphic acceleration mechanism is much better than before.**

Keywords - virtual machine; device model; graphic acceleration

I. INTRODUCTION

With the rapid development of computer science, the virtualization technology utilized in the mainframe [1] is being extended to the area of the desktop computer.

Depending on the mode that operating multiple desktop environments simultaneously, *multiple desktop system* (MPDS) based on virtualization technology will change the traditional usage of desktop computers. Since each desktop environment is isolated from another, users could use several or more desktops at the same time to access different applications. Users can quickly switch the desktop environment systems seamlessly, rapid deploy and migrate [2] the desktop environment after users define its usage on their demand. Because of the adaptation of the virtualization, MPDS provides good user experience, which is different with the traditional desktop computer systems.

MPDS is designed for desktop applications, so it is necessary to improve the performance of execution of desktop program of MPDS to close possible to the physical desktop system. The computing performance of virtual machine is close to the original computing capability of physical hardware [3]. But the I/O performance [4][5], especially the graphic display performance [6], is too low to execute many kinds of desktop applications which need graphic acceleration support. Since the physical graphic card can not be used in virtual machine, desktop applications need to provide a nice graphic user interface for interaction with consumers based on enough graphic performance. Some multimedia or entertainment applications, such as 3D graphic games and the video playback, need higher performance of the graphic display. It is necessary that the performance of graphic in MPDS is enough to process the graphic desktop application.

Therefore, after improving the graphic processing performance, MPDS will be more closed to the original physical desktop environment. Depending on the existing graphic virtualization technology, it needs many works to get enough performance to satisfy most graphic demand in MPDS environment including playback high-definition video and executing 3-D applications.

Improving the graphic performance of MPDS based on existing virtualization technology is really a field worth to study. We propose a graphic acceleration mechanism. The scheme uses the graphic virtualization technology and the inter-domain communication technology. The main idea is as follows. The original graphic library in the *virtual desktop environment* (VDE) is replaced by OpenGL [7] data interceptor to intercept the OpenGL command. Then these data and the desktop interface are transmitted to the domain to access physical graphic device by a high speed inter-domain communication technology. The OpenGL command data are processed in the domain with the full hardware acceleration. At last, the result of graphic acceleration processing is displayed on the domainU with the desktop interface of VDEs.

The rest of this paper is organized as follows. Section II discusses the related work. The detail designation is described in section III. Section IV describes the performance evaluation. We conclude this work in section VI.

II. RELATED WORKS

There are two types of existing graphic process

978-0-7695-4477-9/11 $26.00 © 2011 IEEE
DOI 10.1109/CSE.2011.204

mechanisms in virtual machines: providing virtual graphic device or directly accessing physical graphic card [8]. Two mechanisms are quite different on the usage mode and the implementation. The virtual graphic device, a software simulated in virtual machines, could be used just like using a physical device while the graphic interface could be displayed through communication protocols. The physical card direct access [9] based on the hardware support could provide better performance than software simulation. Existing virtual machine systems, such as VMware products, kernel-based virtual machine (KVM) and Xen, all implement the virtual graphic device mechanism in virtual machine but exclude the hardware direct access. Existing graphic devices could be accessed by one domain using its only one function interface. In the following part of this section, we will introduce some related instances of both two types such as VNC [10], Xen virtual frame buffer [11], VMware's virtual GPU [12] and VMGL [6].

Virtual network computing (VNC) is a platform-independent remote desktop system with a graphic interface. It was originally developed by the Olivetti Research Laboratory in Cambridge. Its original source code and many derivatives are open source under the GNU General Public License. It is composed by a VNC viewer on one operation system and a VNC server on a different operation system. It shares the server's desktop to clients by using the RFB protocol. A remote computer could control another one. Multiple clients may connect to a VNC server at the same time. It transmits the keyboard and mouse events from the client to the server, depending on the graphic screen updates over a network. The technology is used in virtual machine system for the virtual machine to control and access files on virtual machines from monitor computer.

Xen's virtual frame buffer is a virtual graphic device implemented in Xen based virtualization systems. The frame buffer, implemented for *hardware virtual machine* (HVM), maintains a remote desktop server in Domain 0 with interfaces including communication with VNC protocol or accessing with *Simple DirectMedia Layer* (SDL). The guest writes the graphic data into the frame buffer, and notifies the Domain 0: which area has been updated. Both VNC protocol and SDL library will retransmit the parts of the image of which the change detected and the virtual frame buffer maps directly to these methods. After some work based on the source code from QEMU [13], this buffer is able to be accessed by para-virtualized guest virtual machines in latest version of Xen.

These solutions based on software simulation cannot allow virtual machines to access the graphic hardware. With these implementations, the graphic data are processed by the processor and memory, which is lack of any acceleration capacity, provided by the physical graphic device in general physical machines.

Besides these, two other methods, VMware's virtual GPU and VMGL, are implemented based on front-end virtualization [14]. In the mechanism, the graphic data are transported to a monitor machine with hardware graphic acceleration and displayed to users after processed in this domain. VMware's virtual *graphic process unit* (GPU) is also a virtual graphic device which supports VGA and basic high resolution 2D graphics modes in VMware products. On VMware's hosted products, this virtual GPU also haves a graphic acceleration capability using a front-end virtualization strategy. This strategy meets the good portability and performance requirement since a front-end virtualization provides integrating well with existing operating system driver models. The virtual GPU has some characteristics similar with remote API parsing but not only an emulation device. Just like the mechanism in VNC, the driver and the virtual GPU in guest machine communicate with VMware's *Mouse-Keyboard-Screen* (MKS) abstraction asynchronously. MKS, which runs as a separate thread, owns all access to the host GPU. The performance that virtual GPU process complex graphics application is not good enough but it is already enabling virtualized applications to run interactively that could never have been virtualized before. It is a foundation for virtualization of tomorrow's GPU-intensive software.

VMM-Independent Graphics Acceleration (VMGL) is another mechanism for applications using the GPU in virtual machine. It is both VMM and GPU independent solution for cross-platform OpenGL virtualization. After OpenGL virtualization is implemented, VMGL could provide accelerated rendering capability for applications. Many existing virtual machine systems could adopt VMGL such as VMware workstation, KVM and Xen. VMGL is running in the host domain such like the Domain 0 of Xen or VMM host of VMware Workstation. In this host domain, with the help of the driver and the OpenGL library provided by the graphic hardware venders, hardware graphic acceleration in the operation system is supported. Through network transportation of OpenGL commands, VMGL exports the OpenGL library to the virtual machine for running graphic applications. The OpenGL commands are transported to the host machine with the capability of hardware support graphic acceleration. It is not necessary to modify the source code or rewrite binary when using VMGL. The graphic process performance of VMGL is really close to the native machine, but under such workloads the total bandwidth between application virtual machine and the host machine will be a performance bottleneck.

III. DESIGN OF MPDS

A. System Architecture

As shown in Figure 1, the system is composed by a virtual machine system, a desktop environment controller, a quick change controller, a desktop environment monitor, and several desktop environments.

In MPDS, the virtual machine system is the basic of the whole system. Each VDE in MPDS is based on a virtual machine. We adopt Xen as the virtual machine system supporting the upper application. The desktop environment controller is used for VDEs management. In MPDS, VDE are deployed, controlled, and managed by the desktop environment controller. After users input some special hot keys, this controller adjusts the layout of the window of desktop environment. The desktop environment monitor

realized an interaction for users operating the VDE. This monitor displays its graphic interface after starting the VDE. We also implement many *multimedia desktop environments* (MDEs) for different usage. There is an environment designed for multimedia application. This environment brings a television like experience to users for its multimedia functions and lightweight realization.

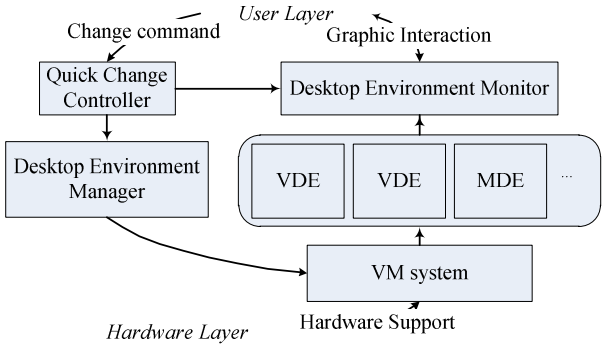

Figure 1. System architecture of MPDS

B. Desktop Environment Monitor Design

As shown in Figure 2, the desktop environment monitor is composed by a graphic acceleration module, a desktop display and control module, and a window manager module. The graphic accelerate module is the key module for providing a method to accelerate graphic process in MPDS.

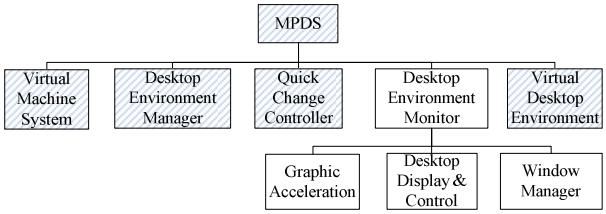

Figure 2. Desktop environment monitor in MPDS

The graphic acceleration module which accelerates the desktop graphic processing is used to satisfy the users for the capability of handling complex graphic application. This module intercepts the OpenGL command data after the application called the OpenGL API in VDE. The OpenGL command data will be transmitted to the virtual machine monitor through the high speed inter-domain channel. The module will display the graphic result at last after the accelerator supported by the hardware has processed the OpenGL command in virtual machine monitor. The desktop display and control module displays the interface of VDEs and sends the keyboard and mouse events to VDE. The window management module manages the display of desktop environment in users' domain. According the priority level, the window management module creates a layout of these windows for users in one screen. This module, furthermore, responds the command of the quick change controller and changes the layout of windows to finish the user's new usage mode.

C. Graphic Acceleration Mechanism Design

As shown in Figure 3, the graphic acceleration mechanism is composed of four parts, the graphic interceptor, high speed communication channel, OpenGL graphic display, and desktop display.

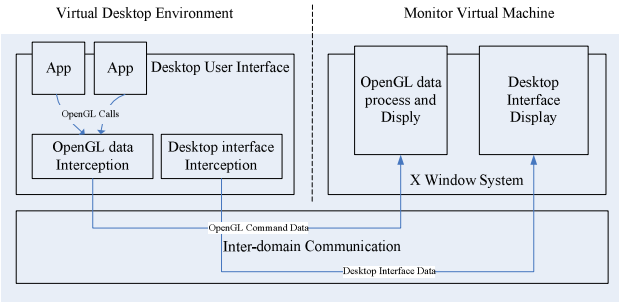

Figure 3. Graphic acceleration mechanism design

This mechanism replaces the OpenGL library in VDE by a graphic interceptor. This interceptor gets all OpenGL commands. After getting these commands, the OpenGL data interceptor packs these data and transmits them to the OpenGL graphic display through the channel high-speed communication implement. This interceptor gets desktop graphic interface data by invoking the *Xlib* function.

Transmitting the graphic data needs high bandwidth communication channel. The graphic acceleration mechanism constructs high speed channels based on inter-domain communication technology. The high speed communication channel is the key method for the graphic acceleration. After the communication channel has been created, this mechanism provides APIs. The OpenGL graphic data and the desktop graphic data will be transmitted in these communication channels but the control message is still transmitted by the network.

In the virtual machine monitor, the graphic acceleration mechanism creates several daemon applications called OpenGL graphic processor. These processors receive the OpenGL command data and perform these commands with the OpenGL library in the virtual machine monitor. There is a desktop interface window in the virtual machine monitor mapped to a VDE. After receiving the data from the VDE, this window displays the desktop environment's graphic interface with the result image of those executed OpenGL command. Another purpose of this window is to transmit the keyboard and mouse events in the area of the window to the VDE to response the users' operation.

D. Inter-domain Communication Interface Design

We implement an inter-domain communication in MPDS for transmitting the graphic data from applications in VDE to display processes in virtual machine monitor. For a better usage of this high speed technology, we design a set of interfaces to control the inter-domain communication. There are four goals when designing these interfaces.

First, the inter-domain communication uses the device mode in Linux and controls the data transmission through the device driver. The data input, transmission and output

needs a complex work procedure. As a result, the interfaces hide the technical details to the upper layer for a transparent usage.

Second, all of the graphic data will be reflected in users' display. Any error in transmission will give influence on the graphics quality at last. So, there is a set of interface used for dealing with the error and exception in the data transmission.

Third, these interfaces are simple and easy to use. The usage of the inter-domain communication is only the graphic data transmission. It is unnecessary to design complex interfaces for a single purpose.

Last, compared with some existing inter-domain communication instances such like XWay [15] and Xenloop [16] which override the network communication, these interfaces do not give influence on other communications. To design this set of interfaces, we make a reference of XenSocket [17]. This set of interfaces designed for transmitting graphic data will not bring any performance penalty on parsing the network protocol stack.

This set of interfaces is composed by five catalogs of interfaces, including transmission initialize, transmission channel open and close, data send and receive and exception, error handle interfaces. These interfaces include all operations of the inter-domain communication such as data chains manage, open and close data chains, exception and error handling, write or read data chains.

IV. PERFORMANCE EVALUATIONS

A. Hardware Environment

As the development of MPDS is based on Xen, the hardware environment for running MPDS graphic acceleration mechanism should meet the requirements of Xen. In addition, MPDS system employs Windows based VDE supported by Xen's HVM, so CPUs used in the test should also have the hardware virtualization like Intel-VT or AMD-V. Furthermore, at least one physical graphic card is necessary to support the image processing of proposed graphic acceleration mechanism.

Smooth running of multiple VDEs requires sufficient physical memory. In our experiments the hardware environment setting is as follow: Intel dual core CPU with single core frequency as 2.00GHz; 160GB hard disk; a 2GB memory and a 512MB one in two slots consist 2.5GB memory totally; network bandwidth is 100Mbps.

B. Software Environment

Software suites used in the experiments are as follows.

- Quake 3: this is a FPS (*First Person Shooting*) game. It uses an extensive selection of OpenGL graphic techniques including shadow scripts, texture, fogs and illumination, Gorade painting and rendering, sampling based curved surface. All the above features make Quake 3 a basic application for testing 3D graphic performance.
- Mplayer: this is a widely used open-source media player. Mplayer could run on most mainstream operating systems and supports various video codec, output modes including texture-driven OpenGL output.
- Glxgears: this is commonly used to demonstrate the GLX extension of OpenGL on X Window system, which could paint three rotating gears and show the related frame rate information.

C. Evaluation Metrics

The objective of graphic acceleration mechanism in VT-based multiple desktop environments is for the implementation of a complete solution which supports proper display of images and graphic acceleration. To evaluate the performance of our mechanism, following metrics are proposed.

Performance of 2-D video playback and 3-D graphic processing is measured by the FPS (*Frames per second*) value after running these test scripts on Fedora Core 7 VDE. Higher FPS value indicates better performance, which consequently demonstrates that the graphic acceleration mechanism has stronger capacity in both handling 2-D video playback and 3-D graphic applications. This capacity, supported by graphic acceleration mechanism, is not only needed by MPDS for its multimedia applications, but also an important function for processing general graphic applications. Better 2-D video playback and 3-D processing capability is good indicator that MPDS system could support both multimedia playback and basic 3-D applications, which could improve user experience of using MPDS system.

D. Evaluation Results and Analysis

In this section, we present the evaluation results on the three metrics previously described, including 2-D video playback and 3-D graphic processing. The performance of 2-D video playback indicates whether MPDS could smoothly supports video playback desktop applications. The performance of 3-D playback indicates whether MPDS has the capability to run 3-D applications.

1). Evaluation of 2-D video playback

Video playback is one of the most popular applications on common physical desktop. It is running mainly supported by performance of system's 2-D video playback. In our test, the performance of 2-D video playback in MPDS's graphic acceleration mechanism is measured by the FPS value. In this part, the influence of video resolution is also taken into consideration. To assure the accuracy of the evaluation, each experiment is conducted 5 times and the average metric value is calculated based on all the 5 test results. Besides, we test the 2-D video playback performance when multiple VDEs coexist including Windows XP and Fedora Core 7, which could demonstrate the applicability of MPDS's graphic acceleration mechanism.

Furthermore, we use the same test method in MPDS system to compare the performance of video playback run directly in virtual machine with our proposed VDE. Since the resolution of our computer LCD is 1024x768, we download two lower resolution videos from Internet of 1024x576 and 640x360 for Mplayer playback in performance test mode. They are both widescreen with the popular 16:9 ratio. As illustrated in Figure 4, video playback

performance of three different testing environments are listed, including play videos with different resolution, and play videos under multiple VDE.

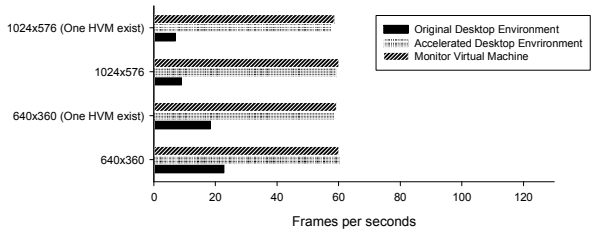

Figure 4. Mplayer performance of the original desktop enviroment, the accelerated desktop enviroment and monitor virtual machine

Based on testing results of Figure 4, the comparative results of video processing performance under different conditions (with/without graphic acceleration mechanism, directly using hardware acceleration) are depicted. It could be obviously identified from the figure that the performance increases with graphic acceleration mechanism on, and nearly reaches the performance of directly using hardware acceleration. However, under the situation of lower video resolution, the performance gap decreases. In addition, because the Windows VDE is based on the hardware virtual machine, which would consume more system resource, so the video playback performance of all the three conditions falls under the existence of Windows VDE.

2). Evaluation of 3-D graphic processing

Similar to the testing method used for 2-D video playback performance, we test the FPS value of running 3-D applications with/without graphic acceleration mechanism. All experiments are conducted repeatedly 5 times. To evaluate the influence of various applications on the performance of 3-D graphic processing, we test the FPS values of running both simple application Glxgears and complex Quake 3 with the same resolution.

a) Glxgears: The FPS values of running Glxgears under 3 different conditions are illustrated in Figure 5. Based on the testing results, the comparative results of 3-D processing performance under different conditions (with/without graphic acceleration mechanism, directly using hardware acceleration) are depicted.

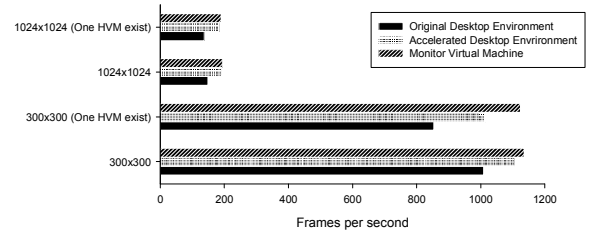

Figure 5. Glxgears performance of the orignal desktop environment, the accelerated desktop environmet and the monitor virtual machine

From the result, the 3-D graphic processing performance, improved after graphic acceleration mechanism on, is very nearly the same as the direct hardware acceleration. However, same as in video playback, lower image resolution would weaken the performance gap and existence of Windows VDE would also negatively reduce the 3-D processing capability.

b) Quake 3: The FPS values of running Quake 3 under different conditions are illustrated in Figure 6. Differently from glxgears test, from the graph we can see that the performance gap. The explanation for above fact is that, without graphic acceleration mechanism, 3-D graphic processing is executed on CPU. Even 3-D graphs of lower resolution have less image processing demand for each frame, which could not be handled well with the low software graphic capacity of CPU in processing complex applications like Quake 3. When the resolution grows higher, so does the image processing demand for each frame which grows in an exponential manner. Thus, the situation became worse that more time is needed by CPU to render each frame. But the graphic acceleration mechanism could hand most of the capacity of render units which are specifically designed for 3-D processing and could significantly improve the performance.

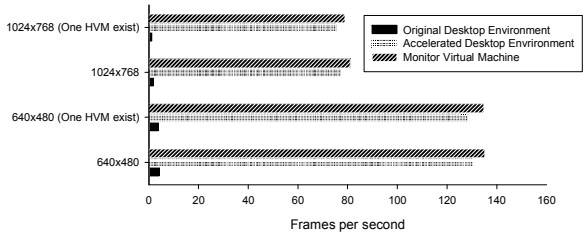

Figure 6. Quake 3 performance of the original desktop enviroment, the accelerated desktop enviroment and monitor virtual machine

From the comparison of above testing results, when enabling the graphic acceleration mechanism, there is much greater improvement in performance for running complex application Quake 3 than simple Glxgears. The reason is that Quake 3 needs more 3-D processing calculation than Glxgears, which could take more advantage of our proposed mechanism. The performance is increased 40 times for Quake 3. In general, both applications have experienced better 3-D processing performance with graphic acceleration mechanism on. Existence of other VDE (especially the one based on HVM) would impair the effectiveness of our acceleration mechanism on both applications.

E. Summary

This section compares the effectiveness of MPDS graphic acceleration mechanism for 2-D video playback and 3-D image processing. First, we analyze the performance of MPDS graphic acceleration mechanism for processing 2-D video playback with both high and low video resolutions. Experiment results show that the video playback capability of MPDS has been greatly improved. Higher resolution video playback experiences much more increased performance. Second, experiments of 3-D image processing show that the higher demand of processing calculation from

complex 3-D application makes it benefit more from our mechanism and experience much more increased performance. For Quake 3, the performance has risen 40 times and 30 times separately for high resolution and low resolution image. From the above results and analysis, it can be identified that employment of graphic acceleration mechanism greatly improves the image processing capability for MPDS, which nearly reaches the same level of performance under physical desktop environment.

V. CONCLUSION

To improve the graphic performance in MPDS, we propose the graphic acceleration mechanism and implement this mechanism in the MPDS desktop environment monitor.

After using this mechanism in MPDS, the graphic performance is greatly improved and the effect of executing some graphic applications is close to the physical machine. The experimental data illustrates the performance of the graphic acceleration system good.

In the future, based on our work, we will try to optimize the mechanism from following points: first, providing the support for new OpenGL graphic application; second, providing the support for Windows operating system with the graphic application.

ACKNOWLEDGMENT

This work is supported by National 973 Basic Research Program of China under grant No.2007CB310900, the MoE-Intel Information Technology Special Research Foundation under grant No.MOE-INTEL-10-05 and the National Science and Technology Major Project of the Ministry of Science and Technology of China under grant No.2010ZX-03004-001-03.

REFERENCES

[1] M. Rosenblum and T. Garfinkel, "Virtual Machine Monitors: Current Technology and Future Trends," *IEEE Computer*, pp.39-47, May 2005.

[2] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, "Live Migration of Virtual Machine Based on Full System Trace and Replay", In *Proceedings of the 18th International Symposium on High Performance Distributed Computing*, 2009.

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the Art of Virtualization," In *Proeedings of 19th ACM Symposium on Operating Systems Principles*, Lake George, NY, Oct 2003.

[4] J. Sugerman, G. Venkitachalam, and B. Lim, "Virtualizing I/O devices on VMware workstation's hosted virtual machine monitor," In *Proceedings of USENIX Annual Techical Conference*, 2001.

[5] L. Cherkasova and R. Gardner, "Measuring CPU overhead for I/O processing in the Xen virtual machine monitor," In *Proceedings of USENIX Annual Technical Conference*, Anaheim, CA, April 2005.

[6] H. A. Lagar-Cavilla, N. Tolia, M. Satyanarayanan, and E. de Lara, "VMM-independent graphics acceleration," In *Proceedings of Third International ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments*, 2007.

[7] M. Woo, J. Neider, T. Davis, and D. Shreiner, *OpenGL Programming Guide*, Addison Wesley Longman, Inc., Third edition, 1999.

[8] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper, "Virtual network computing," *IEEE Internet Computing*, 2(1):33-38, Jan./Feb. 1998.

[9] D. Chisnall, *The Definitive Guide to the Xen Hypervisor*, Prentice Hall International, 2007.

[10] M. Dowty and J. Sugerman, "GPU virtualization on VMware's hosted I/O architecture," In *Proceedings of First Workshop on I/O Virtualisation*, San Diego, CA, Dec 2008.

[11] R. Hiremane, "Intel Virtualization Technology for Directed I/O (Intel VT-d)," *Technology@Intel Magazine*, 4(10), May 2007.

[12] J. Duato and F. Igual, "An efficient implementation of GPU virtualization in high performance clusters," In *Proceedings of Euro-Par 2009–Parallel Processing Workshop*, Delft, Netherlands, Aug. 2010.

[13] F. Bellard, "Qemu, a fast and portable dynamic translator," In *Proceedings of the USENIX Annual Technical Conference*, CA, Apr 2005.

[14] K. Fraser, S. Hand, R. Neugebauer, I. Pratt, A. Warfield, and M. Williamson, "Safe hardware access with the Xen virtual machine monitor," In *Proceedings of first Workshop on Operating System and Architectural Support for the On-Demand IT Infrastructure*, Boston, MA, Oct 2004.

[15] K. Kim, C. Kim, S.-I. Jung, H. Shin, and J.-S. Kim, "Inter-domain Socket Communications Supporting High Performance and Full Binary Compatibility on Xen," In *Proceedings of Third International ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments*, Seattle, WA, March 2008.

[16] J. Wang, K. Wright, and K. Gopalan, "XenLoop : A Transparent High Performance Inter-VM Network Loopback," In *Proceedings of International Symposium on High Performance Distributed Computing*, Boston, MA, June 2008.

[17] X. Zhang, S. McIntosh, P. Rohatgi, and J. L. Griffin, "Xensocket: A high-throughput interdomain transport for virtual machines," In *Proceedings of International Middleware Conference*, Newport Beach, CA, Nov 2007.