Cloud Environments with GPU Virtualization: Problems and Solutions

Vladimir A Smirnov, Evgeniy V Korolev and Olga I Poddaeva

Moscow State University of Civil Engineering

Abstract**:** *Remote computing is the current trend in information technologies. At present, productive remote access to the office workplaces composed with word processors, spreadsheets and similar tools is not something unusual. However, setting up multi user environment for efficient remote work with graphically intensive applications (CAD/CAM and GPU computing) is still a big challenge. Several vendors offer solutions for GPU virtualization in cloud environments; such solutions are quite expensive, designed for tight integration with specific hardware, host and guest operating systems and directory services. There are cases when commercial solutions can not be avoided; but there are also cases when they can. It the present work we briefly review several types of use cases and discuss feasible hardware and software solutions for GPU virtualization, paying primary attention to the software tools covered by open licenses. During analysis of our test systems we have cleanly demonstrated the possibility of GPU virtualization for multi user OpenCL development environment. While the conclusion about such possibility can also be made during analysis of publically available online resources, the information required for the implementation is still scattered. We have consolidated the relevant information, verified most interesting solutions and provided links to the appropriate configuration files.*

Keywords: *cloud computing, PaaS, virtualization, GPU pass-through, QEMU, KVM, GPGPU*

1. Introduction

Recent trends in IT in business and academic include numerous technologies for remote computing. Several terms are invented for such technologies – *cloud computing*, *software as a service*, *platform as a service* etc. While these technologies have their roots in remote access (*telnet* [1]) and network-transparent graphics (*X Window System* [2]) solutions that were developed more than 30 years ago, they are also in constant evolution due to current demands.

Productive remote work with office applications – word processors, spreadsheet applications and similar tools – can be performed even through web interface, especially if the client browser properly supports HTML5. However, many types of workloads are graphically intensive and require graphics processing units (GPU) either for interactive rendering or for GPU-based computing. Even simple estimation can reveal that Gigabit Ethernet link can carry uncompressed high color (16 bit per pixel) 1024x768 video stream with interactive frame rate. Required bandwidth can be reduced by orders of magnitude in case of advanced video compression methods. Thus, it can be assumed that setting up multi user interactive graphical environments is a simple task; unfortunately, this assumption is very far from true. The complexity is the main reason for this: configuration of interactive multi user graphical environment must encompass and interact with elements starting from hardware layer (GPU drivers) though windowing (and, sometimes, audio) system up to network layer; the input events from the client must be delivered undistorted (this is a real problem with Virtual Network Computing – VNC – protocol) and processed with minimal delays, and so forth. Interactive graphical environment is not as simple as video streaming.

At the moment, only three commercial implementations (offered by Microsoft, Citrix and VMware) can be adopted for the purpose; all of them require expensive GPU and proprietary operating systems (OS) with directory services. Tight integration with proprietary directory services is beneficial mostly for business environments; in academic institutions such integration may only unnecessarily complicate administration (though Linux and FreeBSD clients can nicely coexist with Microsoft Active Directory).

Complex nature of client-server solutions for the graphical remote desktop environments also causes a lot of confusion associated with terms "virtual workplace" and "remote workplace". For instance (as of July 2015), the former is "defined" in Wikipedia as a "workplace that is not located in any one physical space" [3]. By itself, such definition says little about distribution of workload along the networked environment. Henceforth, terms *virtual workplace* and *remote workplace* will be used for the environment in which *essential parts of applied tasks are performed on day to day basis on the system that does not associated with local console*. In particular, remote session (either text console or graphics terminal) to the network server for administration purposes can hardly be considered as a virtual workplace – just because for properly configured system such sessions should not be opened on a day to day basis. On a contrary, text console to the packet batch scheduler on the high performance computing (HPC) cluster is, by definition, a virtual workplace. To completely designate the proposed sense of the virtual workplace, the uncertainty that is tied to the word "essential" should also be eliminated.

2. Workloads in Cloud Environments

The "essential" part of work that is performed within the virtual workplace can be coupled with four main types of workloads (Table I). These workloads mostly correspond to *Platform as a service* (PaaS) service model of cloud computing, though in some cases virtual workplace can be composed of single interactive application (SaaS service model).

TABLE I: Classification of PaaS Workloads

Applied task:	Content creation	Computation
GPU-agnostic:	Type I: interactive office workloads: word processing, spreadsheet applications, database clients, many types of interactive image processing (e.g. vector graphics editing with Dia, Inkscape and similar tools).	Type II: CPU-intensive computation, usually in HPC environments.
GPU-intensive:	Type III: interactive graphics workloads: 3D content creation (Blender, 3ds Max, etc.), CAD, GPU-intensive image processing (albeit video games are irrelevant to content creation, they also correspond to this category).	Type IV: General-purpose GPU computing (GPGPU): using GPU as coprocessors for numerical simulation packages (ANSYS, ABAQUS etc.), GPU rendering (Blender Cycles, V-Ray, LuxRender) and other applied GPGPU tasks, including hybrid HPC clusters.

Historically, remote workplaces were first used for the Type II workloads: these are well-known text mode remote sessions to telnet and secure shell (SSH) servers. Type I workloads nowadays can be handled by different client-server technologies: Microsoft Remote Desktop Protocol (RDP), VNC, proprietary remote access software packages and WEB technologies (Google Documents etc.) – either as complete virtual workplaces or WEB clients for other remote access protocols.

Type III and IV workloads are of primary interest in the present work; it will become clear from further discussion that setting up virtual workplace for Type III workloads is most problematic. Handing of Type III and IV workloads requires:

- GPU sharing (either on *one-to-many* or *one-to-one* basis) either between virtual machines (VM) within same host (*hardware* GPU sharing, hereafter *GPU virtualization* or *GPU pass-through*) or between software graphics servers within the same OS. This requirement is necessary and sufficient for the Type IV remote workload. The problem of exposing the graphics hardware to guest OS is relatively new; it is described in detail in e.g. [3], albeit the proposed solution for GPU sharing does not rely on the most appropriate hardware features ([4] and, later, [5]).
- Specific server software (hereafter *virtual graphics server*, VGS) on each virtual machine (mostly for Type III workload) and specific client software (hereafter *virtual graphics client*, VGC); the latter is only required for Type III workload.

If both requirements are satisfied, then infrastructure is ready for Type III and IV remote workloads; if neither one is satisfied, then resulting system is a single user graphics workstation. But there can also be cases when exactly one requirement is satisfied (Table II).

TABLE II: Requirements for Virtual Graphical Workplace

Requirements	GPU sharing: not implemented	GPU sharing: implemented
Virtual graphics server and client: not implemented	*Single user local workstation*	*Multi user local workstation*
Virtual graphics server and client: implemented	*Single user remote workstation*	*Multi user remote workstation*

Implementation of GPU sharing without VGS and VGC leads to local multi user graphics infrastructure. Several users can independently do their tasks on a single workstation equipped with multiple GPU directly connected to local consoles. There are cases when such configuration is profitable (in terms of hardware price). What is more important is that implementation of GPU sharing is enough for GPGPU (Type IV) virtual workplace, because remote access can be organized either as plain text (SSH) console or as an X tunnel (more in section 5.1) between server with high-end GPU(s) dedicated for computing and client with low-end GPU dedicated for rendering.

Conversely, implementation of VGS and VGC (*GPU remote access*) without GPU sharing leads to single user remote graphics workstation: Type III workload can be handled only for single remote user. Such configuration can be adequate in case of CPU- and GPGPU- intensive tasks, when sharing the host between several users is not beneficial al all.

There were also reports of VGS and VGC implementation from scratch. In particular, [6] describes solution for realtime 3D virtual appliances. The proposed solution includes graphics rendering network and a media streaming network for transcoding rendered frames into H.264/MPEG-4 media streams. It was noted [6] that a prototype implementation of the proposed solution was made, though it is hard to find additional information about it. The solutions similar to [6] should be considered as predecessors for commercial (e.g. [7]) GPU virtualization, sharing and remote access solutions.

3. Commercial Solutions for GPU Virtualization, Sharing and Remote Access

3.1. Microsoft

Microsoft RDP server performs very well on sending Graphics Device Interface (GDI) draw calls along the network, both for vector and bitmap graphics. Excellent interactive performance can be achieved for Type I workloads, especially if the Microsoft RDP server is combined with some open source RDP clients (as a rule, configuration and tuning of such clients is required). When it comes to GPU draw calls, the situation changes dramatically.

Even if Microsoft RDP server is operating on the system equipped with GPU, RDP by itself (non-extended RDP) does not properly handle Type III workloads. This is due to several reasons. RDP server switches all 3D application programming interfaces (API) – OpenGL and Direct3D – to CPU-based (software) implementations; GPGPU APIs (OpenCL and CUDA) becomes unavailable. In particular, Microsoft software OpenGL implementation that was written in early 1990s is still used in RDP session, even on modern Windows OS. Thus, non-extended RDP protocol can not be used for GPU remote access. There is, however, a trick which allows to use non-extended RDP for GPU remote access: if we start program from local console (or VNC) session and reconnect to the same session by RDP, then running program will continue to use hardware-accelerated APIs (in general, starting remote "console" RDP session is not enough, and many open source RDP clients does not properly support such sessions). This trick works at least with Windows 7 OS with NVIDIA desktop-grade GPU. No doubt, relying on such tricks for GPU remote access is a bad practice.

Efficient GPU virtualization has to be based on hardware I/O virtualization technologies similar to [4,5]. Hyper-V hypervisor is offered as a part of Windows Server and some versions of desktop Microsoft operating systems. As an employee of Microsoft had explicitly stated, VT-d technology is in the list of "hardware features that Hyper-V does not utilize and enabling them will prevent Hyper-V from loading" [8]. Also, neither VT-d nor AMD-Vi is mentioned on [9]. At the same time, Hyper-V supports [10] Single-Root I/O Virtualization (SR-IOV), but the hardware prerequisites for SR-IOV are I/O Memory Management Unit (IOMMU) and Direct Memory Access Remapping (DMAR) [11], which are, in turn, parts of VT-d specification. Thus, search for the answer on a simple question "Does Hyper-V support PCI (Peripheral Component Interconnect) pass-through?" is complicated by contradictions and inconsistencies. This has stopped us from further exploration; we assume that Hyper-V is incapable to pass-through PCI devices (in particular, physical GPU that are installed in PCI Express slots) to the guest VM.

Nevertheless, starting from Windows Server 2008 R2 SP1, Microsoft offers the solution to handle Type III workloads remotely. RemoteFX is the "set of user experience technologies" that "delivers a full-fidelity user experience for Virtual Desktop Infrastructure by providing a 3D virtual adapter, intelligent CODECs" and "is integrated with the Remote Desktop Protocol" [12]. Thus, RemoteFX is the solution for GPU remote access; conventional RDP client (with support of RDP version 7.1 and above) can be used as VGC. It has nothing to do with GPU pass-through; in terms of [7], RemoteFX is *API interception*. Only *virtual GPU* is visible in the quest OS; all processing is performed in the address space of virtualization host (*control domain*). Contrary to the physical GPU, virtual counterpart has to be controlled by drivers that are specific for hypervisor, and not to the physical hardware.

There are both advantages and disadvantages of the API interception. In particular:

- API interception can be implemented independently from virtualization technologies, on per-application basis within the single OS – either control or guest domain.
- Without software emulation, virtual GPU can represent only subset of the real GPU functionality; available functionality is determined by the hypervisor-specific driver which is also requires updates, for every type of the guest operating system.
- API interception can cause noticeable performance loss (about tens of percents) if compared with GPU virtualization (about tenth of percent).
- Usually there is no implementation for GPGPU API interception.

Despite the only mentioned advantage, for some obscure reasons Microsoft had implemented RemoteFX as a part of "Virtualization host" (Hyper-V) server role.

3.2. Other Vendors

The first report of successful GPU (NVIDIA Quadro) pass-through with help of IOMMU/DMAR virtualization technologies was from Parallels (original video by J. Stanley was mentioned in [5], currently this video is unavailable). Parallels have implemented pass-through in "Parallels Workstation Extreme" workstation virtualization product [13,14]; in terms of Table II, such solution allows to set up multiuser local workstation and also handle Type IV workloads (local and/or remote).

Citrix [15] and VMware [16] offer complete solutions for handling Type III remote workloads. Citrix XenServer (based on XEN [17]) and VMware Hypervisor support GPU pass-through for workstation-grade NVIDIA (Quadro, GRID) and ATI/AMD (FirePro) GPU. In Citrix virtualization suite, XenDesktop, XenApp and Citrix Receiver can serve as VGS and VGC. Similar functionality is also implemented within VMware GPU virtualization suite (some options are based on API interception). Recent versions of Citrix Receiver are also implemented in HTML5 [18]; this makes them extremely portable. Deploying GPU virtualization suites requires proper configuration of Microsoft Active Directory services. While open source Samba suite [19] can serve as Active Directory domain controller, some features that are necessary for Citrix and VMware suites are not implemented yet (as of version 4.2.1). Thus, Microsoft Windows is always required as a part of environment.

4. GPU Remote Access with Open Source

Brief review of the GPU remote access technologies should be made prior to focusing on the primary aim of the present work (GPU virtualization).

4.1. Intrinsic Functionality of the X Window System

X Window system [2] is designed to be network-transparent: X draw calls issued by application are encapsulated into stream of events (X protocol) that can be processed either by local or remote X server. GLX [20] is the extension of the X protocol; it allows encapsulation of OpenGL draw calls. Does this solve the problem of GPU remote access? The answer is "by no means".

First of all, solution for remote Type III and IV workloads assumes that draw and GPGPU calls issued by application are processed by server-side consolidated GPU hardware; only results (rendered frames and outcomes of GPU-based calculations) should be transferred through the network. X operates in somewhat reversed manner: draw calls are processed by X server that is installed on the local system (i.e. local X server acts as VGC software, and Xlib on the remote system acts as VGS). There can be use cases suitable for such model of operation but, definitely, this is not what is under examination in the article.

Even worse, the only open source X server [21] is now broken – recent version of X.Org does not properly handle GLX protocol. For example, passing the OpenGL calls from recent (1.17.2) Xlib with software OpenGL (Mesa [22]) libraries to the old (1.7.7) X server with NVIDIA OpenGL (hosted on FreeBSD) is no problem:

```
[client]$ Xorg -version 2>&1 | head -n 2 | tail -n 1
X.Org X Server 1.7.7
[client]$ ssh -Y server
[server]$ Xorg -version 2>&1 | head -n 2 | tail -n 1
X.Org X Server 1.17.2
[server]$ glxinfo | grep 'OpenGL vendor'
OpenGL vendor string: NVIDIA Corporation
```

But it is impossible to pass draw calls from recent Xlib (Mesa OpenGL) to recent X server (NVIDIA):

```
[client]$ sudo Xorg -version 2>&1 | head -n 2 | tail -n 1
X.Org X Server 1.17.1
[client]$ ssh -Y server
[server]# glxinfo | grep --after-context=2 GLXBadContext | head -n 3
X Error of failed request:  GLXBadContext
  Major opcode of failed request:  154 (GLX)
  Minor opcode of failed request:  6 (X_GLXIsDirect)
```

Since the NVIDIA OpenGL is the only OpenGL implementation that is suitable for open source desktop environments (more than fifteen years of experience has convinced us), the latter result can only be considered as a software bug.

4.2. VNC, SPICE and open source clients for Microsoft RDP

VNC is platform-agnostic graphical desktop sharing system that uses the Remote Frame Buffer (RFB) protocol [23]. RFB and VNC were developed at Olivetti Research Laboratory in 1990s. The project was opened in 2002. Since VNC is platform-agnostic and open, VNC servers and clients exist for many different graphics subsystems (including Microsoft Windows and X); but not relying on the platform means that we should not expect suitable interactive performance. There is a very promising project that is aimed to overcome poor performance of VNC in Type III remote workloads – VirtualGL [24]. VirtualGL server uses OpenGL API interception which was already discussed above. VirtualGL does not intercept X draw calls; because of this, for better performance the entire graphical user interface (GUI) should be implemented in OpenGL (as it is done in, e.g., Blender [25]). Associated problems (input and audio) still have to be resolved.

Simple Protocol for Independent Computing Environments (SPICE) [26] is another graphical desktop sharing system. It was developed as proprietary product and opened in 2009. The most notable differences from VNC are advanced authentication features, compression algorithms and support for audio streaming. Currently, SPICE server is implemented only as a part of QEMU [27] and can not be used for GPU remote access.

There are also exist open source implementations of Microsoft RDP server and client. Probably, the most advanced open source RDP client is FreeRDP [28]. Its features (as of version 1.2):

- Documentation is scattered, command line syntax is changing over time.
- It may operate twice as fast as Microsoft RDP client, and it also may operate an order of magnitude slower – this depends on configuration.
- Stability of remote connection is significantly reduced in case of AMD/ATI open source graphics drivers. And we do not bother trying proprietary AMD/ATI driver because: a) it is for Linux only, and b) it is prohibited in many up-to-date Linux distributions. FreeRDP operates relatively stable only with NVIDIA proprietary drivers (we are using it for day to day work under Linux and FreeBSD in such configuration).

5. GPU Virtualization with Open Source

The possibility of choice between platforms, policies and implementations is the inherent fundamental feature of the Free and Open Source Software (FOSS). Depending on circumstances, this feature can be either drawback or advantage. The possible options for open source virtualization are summarized in Table III.

In contrast with previous section, GPU virtualization can be successfully implemented with open source hypervisors (Table IV).

TABLE III: Platform Support for Open Source Hypervisors

Hypervisor	Control domain	Guest domain
QEMU+XEN [17]	Linux, NetBSD and FreeBSD[1]	Any x86 or AMD64 OS[2]
QEMU+KVM [29]	Linux, FreeBSD[3], Illumos[4]	Any x86 or AMD64 OS[5]
BHyve [30]	FreeBSD	Only Linux and *BSD (x86 or AMD64)[6]
VirtualBox [31] open source edition (OSE)	Windows, Linux, FreeBSD	Any x86 or AMD64 OS[7]

Notes: 1 – in 11-CURRENT branch; 2 – uses modified BIOS code from bochs project; 3 – uses obsolete code branch; 4 – only Intel CPUs are supported; 5 – uses SeaBIOS; 6 – BIOS must be implemented for booting other OS; 7 – uses own BIOS code.

TABLE IV: GPU Virtualization Support for Open Source Hypervisors

Hypervisor	Support for IOMMU-based pass-through	Support for GPU pass-through
QEMU+XEN	Only for Linux control domain	ATI/AMD and several[1] NVIDIA GPU
QEMU+KVM	Only for Linux control domain	Any GPU[2,3]
BHyve	Yes	No
VirtualBox OSE	No[4]	No

Notes: 1 – Because this hypervisor is used in Citrix XenServer, only NVIDIA GRID and some NVIDIA Quadro [32] are guaranteed to be supported (it is known [33] that most desktop-grade NVIDIA GPU products can not be passed to the VM); 2 – in case of Virtual Function I/O (VFIO) interface that is introduced in Linux 3.9 and can currently be used only with QEMU+KVM; 3 – virtualization of integrated Intel GPU is complicated; 4 – experimental IOMMU-based pass-through is available for Linux in proprietary edition of VirtualBox.

It can be decided from Table III that FreeBSD is the best OS for virtualization – this OS can serve as host for any open source hypervisor. Unfortunately, as it follows from Table IV, the most promising GPU virtualization option is available only if recent Linux kernel is used in control domain.

6. Test Systems, Results and Discussion

Our purpose was to explore the possibility of desktop-grade NVIDIA GPU pass-through and to estimate GPGPU performance of the virtualized GPU.

As the control domain we have used modified Arch Linux – mostly due to ease of customization. Stock init system was cut out from the OS and replaced with OpenRC (https://wiki.gentoo.org/wiki/OpenRC) init system. Configuration for control domain is quite complicated; we have summarized it in [34]. As guest domains we have tested Microsoft Windows XP, Windows 7 and unmodified Arch Linux. GPGPU performance was estimated with Luxmark (it can be downloaded from http://www.luxrender.net/wiki/LuxMark).

Several hardware configurations, both Intel and AMD-based, were tested (Table V).

TABLE V: GPU Pass-through Test Results

System #	Motherboard	Chipset	CPU	Display adapter/GPU	Overall test result
1	ASUS P6X58D PREMIUM	Intel X58	Intel Core i7-970	NVIDIA Titan/GK110	Failed
2	Supermicro X8ST3-F	Intel X58	Intel Xeon E5645	NVIDIA Titan/GK110	Failed
3	Supermicro X8DTL-3F	Intel 5500	2 x Intel Xeon X5650	NVIDIA Titan/GK110	Passed
4	Supermicro X10SL7-F	Intel C222	Intel Xeon E3-1240 v3	NVIDIA GeForce GTX 750/GM107 + NVIDIA GeForce 730/GK208	Passed
5	Supermicro X10SLM+-LN4F	Intel C224	Intel Xeon E3-1220 v3	NVIDIA GeForce GTX 750/GM107	Passed
6	ASUS M5A99X EVO R2.0	AMD 990X	AMD Phenom II X6 1090T	NVIDIA Titan/GK110	Passed

Both X58-based systems are failed to properly support VFIO-based GPU pass-through. All other systems are capable to pass-through GPU to the guest VM (system #4 was tested as a host for two GPU-intensive guests).

Estimation of the GPGPU efficiency does not reveal any noticeable performance loss. For example, Luxmark score for the host system #4 is 3714±1; Luxmark score for the guest VM with exactly the same software configuration (Linux 4.07, NVIDIA 340.76) is 3697±1. The difference is about 0.5%.

Several important moments must be kept in mind while setting up GPU virtualization with QEMU+KVM and VFIO [34]. In many cases platform with AMD CPU is preferable because of relatively low cost and availability of components (CPUs and motherboards) with IOMMU/DMAR support. For Intel platform it is better to use latest server-grade components, paying special attention to the absence of integrated Intel GPU.

7. Summary and Conclusion

Two essential features must be implemented in multi user environment designed to handle GPU-intensive remote workloads: GPU virtualization and GPU remote access. Currently there is no adoptable open source solution for the latter, but the former can be implemented. Moreover, there are features unique for the open source software; for example, with help of QEMU+KVM it possible to pass NVIDIA desktop-grade hardware to the guest VM. Such virtualization causes no noticeable performance loss of CPGPU calculations.

8. Acknowledgements

This work is supported by the Ministry of Science and Education of Russian Federation, Project #7.11.2014/K “Theoretical and experimental study of the dynamics of constructions”.

9. References

[1] (1969). RFC 15. Network Subsystem for Time Sharing Hosts. *[Online]*. Available: https://tools.ietf.org/html/rfc15

[2] R. W. Scheifler and J. Gettys, *X Window System, Third Edition: The Complete Reference to Xlib, X Protocol, ICCM, XLFD, X Version 11, Release 5*, Digital Press, 1992.

[3] M. Dowty and J. Sugerman, “GPU Virtualization on VMware’s Hosted I/O Architecture,” in *Proc. First conference on I/O virtualization (WIOV'08). USENIX Association, Berkeley, CA, USA*, pp. 1-8, 2008.

[4] D. Abramson et al., “Intel Virtualization Technology for Directed I/O,” *Intel Technology Journal*, vol. 10 (03), pp. 179-192, August 2006.

[5] (February, 2009). AMD I/O Virtualization Technology (IOMMU) Specification. AMD Publication #34434. *[Online]*. Available: http://developer.amd.com/wordpress/media/2012/10/34434-IOMMU-Rev_1.26_2-11-09.pdf

[6] W. Shi, Y. Lu, Z. Li and J. Engelsma, "SHARC: A scalable 3D graphics virtual appliance delivery framework in cloud," *Journal of Network and Computer Applications*, vol. 34 (4), pp. 1078-1087, July, 2011.

[7] T. Mackey. (2013). True hardware GPU sharing with XenDesktop and NVIDIA GRID arrives. *[Online]*. Available: http://blogs.citrix.com/2013/12/11/true-hardware-gpu-sharing-arrives

[8] T. Soper. (2011). Hyper-V: How to Fix BIOS Errors Enabling Hyper-V. *[Online]*. Available: http://social.technet.microsoft.com/wiki/contents/articles/3190.hyper-v-how-to-fix-bios-errors-enabling-hyper-v.aspx

[9] Hyper-V Architecture. *[Online]*. Available: https://msdn.microsoft.com/en-us/library/cc768520(v=bts.10).aspx

[10] Overview of Single Root I/O Virtualization (SR-IOV). *[Online]*. Available: https://msdn.microsoft.com/en-us/library/windows/hardware/hh440148(v=vs.85).aspx

[11] A BIOS update may be required for some computers to install the Hyper-V Role and/or start Hyper-V virtual machines. *[Online]*. Available: https://support.microsoft.com/en-us/kb/2762297

[12] (January 2011). Microsoft RemoteFX for Virtual Desktop Infrastructure: Architectural Overview. *[Online]*. Available: http://www.microsoft.com/en-pk/download/confirmation.aspx?id=13864

[13] (April 2009). Parallels Workstation Extreme – The first virtualization with dedicated graphics. *[Online]*. Available: http://www.youtube.com/watch?v=AMDYgi4W1Ag

[14] (2009). VGA pass-through. Parallels Forums. *[Online]*. Available: https://forum.parallels.com/threads/vga-pass-through.96722

[15] Citrix Enterprise Business Products. *[Online]*. Available: http://www.citrix.com/products

[16] VMware Virtualization Products for Virtual Servers, Virtual Desktops, and Data Center. *[Online]*. Available: http://www.vmware.com/products

[17] The XEN Project. *[Online]*. Available: http://xenproject.org

[18] S. Sampath. (July, 2015). Receiver Internals: How Receiver for HTML5 & Chrome Connections Work. *[Online]*. Available: http://blogs.citrix.com/2015/07/08/receiver-internals-how-receiver-for-html5-chrome-connections-work

[19] Samba – Opening Windows to a Wider World. *[Online]*. Available: https://www.samba.org

[20] (December 2005). OpenGL Graphics with the X Window System (Version 1.4). *[Online]*. Available: https://www.opengl.org/registry/doc/glx1.4.pdf

[21] X.Org. *[Online]*. Available: http://x.org

[22] The Mesa 3D Graphics Library. *[Online]*. Available: http://www.mesa3d.org

[23] (March 2011). RFC 6143. The Remote Framebuffer Protocol. *[Online]*. Available: https://tools.ietf.org/html/rfc6143

[24] (December 2014). The VirtualGL Project. *[Online]*. Available: http://virtualgl.org

[25] (May 2015). Introduction – Blender Reference Manual. *[Online]*. Available: http://www.blender.org/manual/getting_started/about_blender/introduction.html

[26] SPICE. *[Online]*. Available: http://www.spice-space.org

[27] QEMU. *[Online]*. Available: http://qemu.org

[28] FreeRDP. *[Online]*. Available: https://github.com/FreeRDP/FreeRDP/wiki

[29] Kernel Virtual Machine. *[Online]*. Available: http://www.linux-kvm.org

[30] BHyve. *[Online]*. Available: http://bhyve.org

[31] VirtualBox. *[Online]*. Available: https://www.virtualbox.org

[32] HCL: GPU Pass-through Devices. *[Online]*. Available: http://hcl.xensource.com/GPUPass-throughDeviceList.aspx

[33] Xen VGA Passthrough Tested Adapters. *[Online]*. Available: http://wiki.xen.org/wiki/Xen_VGA_Passthrough_Tested_Adapters

[34] (August 2015). GPU Virtualization and Remote Access. *[Online]*. Available: http://vgpu.libv.org