Baltic J. Modern Computing, Vol. 5 (2017), No. 4, 379-390
http://dx.doi.org/10.22364/bjmc.2017.5.4.04

Virtual Machine Based High-Resolution Display Wall: Experiments on Proof of Concept

Rudolfs BUNDULIS, Guntis ARNICANS

Faculty of Computing, University of Latvia, Raina bulv. 19, Riga, LV-1586, Riga, Latvia

rudolfs.bundulis@gmail.com, guntis.arnicans@lu.lv

Abstract. This paper provides scalability and use case analysis of a prototype for virtual machine based high-resolution display architecture. This architecture has been presented by the authors to overcome the reasons due to which other research results in the high-resolution display wall domain have still not achieved industrial success. Authors have provided use cases of this architecture with common operating systems like Linux and Windows and common software applications to demonstrate how a display wall solution can become seamless to the software layer while providing scalability, which is limited in the hardware-based display wall solutions that dominate the industry.

Keywords: large high-resolution display, tiled displays, large-scale visualization, virtual machine

1. Introduction

The need for high-resolution display walls is trending due to the growth of the imaging device resolutions. However, the current solutions that focus on maximizing the overall resolution of homogeneous display surfaces remain mostly academic and have not conquered commercial success. Bundulis and Arnicans (2013, 2014) have analyzed the leading research results in this domain and tried to tackle the downsides of the existing solutions by proposing a new display wall architecture that is based on GPU virtualization.

Even though the demand in the industry exists, several factors prevent such great systems as SAGE (Jeong et al., 2006) and Chromium (Humphreys et al., 2002) to be utilized from small and medium business throughout to large enterprises:

- A common misconception that a display wall is always more expensive and harder to deploy than a single high-resolution display (4K or 8K) due to the additional hardware. For example, Reality Deck (Papadopoulos et al., 2014) – a 1.5 gigapixel cluster based display surface cost the authors approximately $950 000 which is around $0.00063 per pixel.

- The inability to create a large homogenous surface. Solutions like SAGE or Reality Deck provide great overall resolutions. However, the display surface is still not homogenous – in both cases, it is constructed of independent nodes each driving a part of the wall. Thus, it still seems more attractive to simply use hardware solutions to obtain a reasonably sized homogenous display wall.

- As seen in the survey done by Chung et al. (2014) most of the solutions and frameworks are limited to certain graphics APIs like OpenGL, meaning that commodity software cannot be visualized without modifications. This again is a huge drawback since a display wall should work out-of-the-box and serve as a transparent media transport layer for the source of the visualization.

- All the cluster based approaches suffer from the cost and energy consumption factors for each new node added to the display wall. As seen in (Papadopoulos et al., 2014) the Reality Deck 1.5 gigapixel wall was driven by 18 high-end nodes with 4 GPUs in each meaning the idle power consumption of the whole wall to be around 14 kilowatts. Since the content on the display wall can change from static to very dynamic, the display wall architecture should be able to efficiently scale the power consumption instead of requiring a high idle power for static content. This is one of the factors that intuitively proposes virtualization as the solution since efficient use of available hardware resources was one of the main drivers in the birth of virtualization technologies.

- For static data, the GPU utilization in cluster-based display wall solutions is very poor. The output count of a GPU usually correlates with the overall processing power, meaning that GPUs with multiple outputs are usually mean for high-end applications that require good 3D performance. If such GPUs are used in a cluster for visualizing non-3D data like images videos and graphs the idle power consumption and the price of the cards are unattractive.

- Most of the solutions are scientifically oriented – they are mostly meant for distributed rendering of large 3D models and graphs. This leads to the fact that the architectures are often inefficient for small-scale implementations.

- The solutions that are oriented towards sending some kind of model not pixel data among the nodes (e.g., OpenGL commands) suffer from poor video playback performance, although that, of course, is not the main target in those cases. Still, this leads to the fact that none of the solutions is feasible for everyday use, where the content can change from static images to video and 3D applications.

- If the software applications that provide the actual visualization content are commercial (e.g., CAD tools) then installing them on each node of a clustered system will result in high software license expenses.

In summary, researchers have succeeded to create multiple solutions to target different domains where high-resolution display surfaces are a must have, but they work poorly among different sets of such domains.

The paper is organized into five chapters. The first chapter is an introduction that provides a brief description of the major problems with currently available large-scale high-resolution display wall solutions. The second one is a description of a proposed display wall architecture and the first actual prototype built by the authors. The third chapter contains current results on the scalability of the prototype in a number of displays and resolutions. The fourth chapter is a summary of use cases tried out on the prototype, and the fifth chapter contains conclusions and a roadmap for future work.

2. Virtual machine-based high-resolution display wall

Bundulis and Arnicans (2014) have proposed to solve the downsides of the existing display wall solutions with an approach that uses virtualization as an abstraction layer between the number of displays and resolutions in the display wall and the GPUs available in the computer system that provides the visual content.

GPU virtualization has become a trending technique in the virtualization technologies. The leading solution providers like VMware (Dowty, 2009) have implemented a way to provide the features of the GPU directly to the hosted operating system. Similar technology is provided for Citrix XenServer by NVIDIA vGPU (WEB, a) and RemoteFX (WEB, b) for Microsoft Hyper-V. The GPU virtualization technology has successfully helped to utilize a single GPU in a multi-operating system environment.

This approach can be somewhat applied to the field of display walls too. The problem with GPU virtualization is the same as with an actual GPU that the operating system is limited regarding the actual number of video outputs. However, if the virtualization technology itself presents a purely simulated GPU to the hosted operating system and implements a partial acceleration (for 3D APIs like OpenGL and Direct 3D) by using the physically available GPUs, this can solve both problems – remove the dependencies on the physically available video outputs and increase hardware utilization.

The authors have proposed a general architecture for a virtualized display wall concept (Fig. 1). It consists of a tiled monitor wall with each monitor being backed by a *display node* (DN #) and a host computer system, which provides the actual content for the monitor wall.

The host system runs a software stack currently denoted as *Framebuffer Manager* and some kind of virtualization platform which in turn hosts the guest operating system that is running the content that needs to be visualized on the display wall.

The virtualization platform simulates a virtual GPU that can be freely configured regarding virtual monitors and resolutions to exactly match each desired use case depending on the amount of data that needs to be visualized.

The virtualization platform interacts with the Framebuffer Manager software stack by providing notifications about drawing operations on the guest operating system and access to the video memory contents of the virtual GPU.

The Framebuffer Manager itself performs event-driven management of the framebuffers and handles (crops/scales) the mapping of image data from the virtual monitors to the display nodes in the monitor wall. After the logical partitioning of the image, the Framebuffer Manager uses hardware-based video encoding capabilities in the host system to encode the image and provide an encoded video stream to each display node.

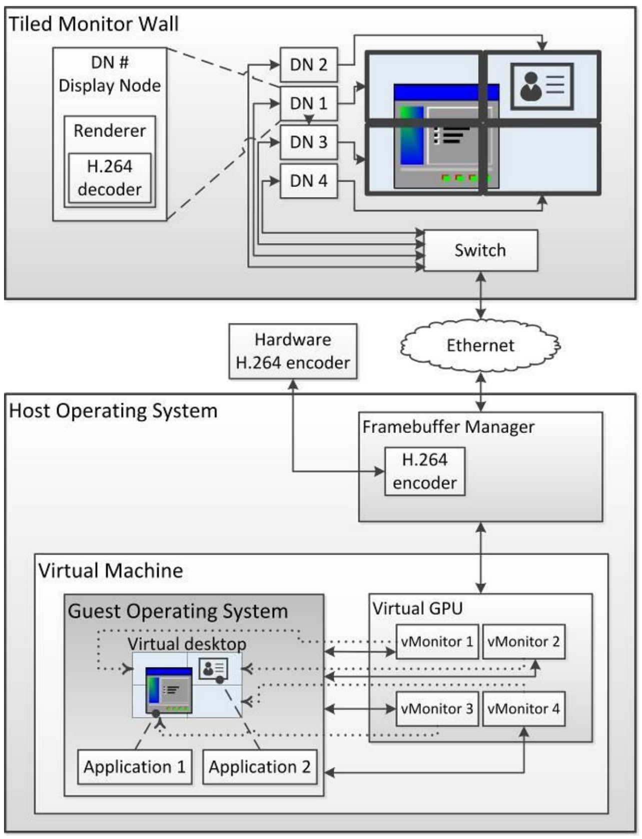

Fig. 1. A general schematic of the proposed display wall architecture

The proposed architecture removes direct dependencies between the needed monitor setup and presence of physical GPUs – all of this is taken care by the Framebuffer Manager and virtualization platform. The Framebuffer Manager takes care of using the present hardware for video encoding and 3D acceleration support for the virtualization platform, while the virtualization platform provides unconstrained display and resolution configuration options.

Bundulis and Arnicans have developed a working prototype of such system (Bundulis and Arnicans, 2014). The current prototype includes a 5x5 22” monitor wall and a host system with moderate hardware – Gigabyte Brix Pro mini PC (Fig. 2). The server has an Intel Core i7 4770R CPU (4 physical cores, 8 virtual cores at 3.2 GHz), Intel Iris 5200 Pro GPU, 12 GB of RAM and Windows 8.1. It runs VirtualBox as the virtualization platform with both Windows and Linux guests. VirtualBox simulates a virtual GPU with 25 display outputs each running at the resolution of 1920x1080. The arguments for choosing VirtualBox as the virtualization platform is explained further on.

The authors have developed a Framebuffer Manager implementation that runs alongside VirtualBox on the host operating system and collects the image data from the

Fig. 2. Display wall server - Gigabyte Brix Pro mini PC

framebuffers of the simulated GPU, encodes them into an H.264 stream using the Intel Iris 5200 Pro GPU and then sends the stream over a gigabit Ethernet to the monitor wall.

Currently, no cropping capabilities have been implemented in the Framebuffer Manager, so the configuration of the virtual GPU is limited to a maximum of 25 displays, and each virtual monitor is always run at the resolution of 1920x1080 since that is the native resolution of each node in the monitor wall.

The monitor wall itself consists of 25 22" DELL displays, each of which is driven by a Raspberry Pi model B (Fig. 3). The Raspberry Pi devices were chosen to implement the role of the display node because of the low cost, efficient power usage and ability to decode a 1920x1080 H.264 at acceptable frame rates for live streaming. Carlos and Garcia (2014) have also demonstrated a successful application of Raspberry Pi embedded systems in a small tiled video streaming solution proving the capabilities of the device in such domain. The Raspberry Pi units are poorest regarding scalability in this prototype since they support H.264 decoding only up to the resolution of 1920x1080 meaning using displays with higher resolution is not possible in the current prototype. However, since the embedded systems are developing vastly, the authors do not see this as a significant issue. NVIDIA has already released Jetson K1 embedded system that can drive a 4K monitor (WEB, d).

Further on in this paper, we are going to describe the current measurements of

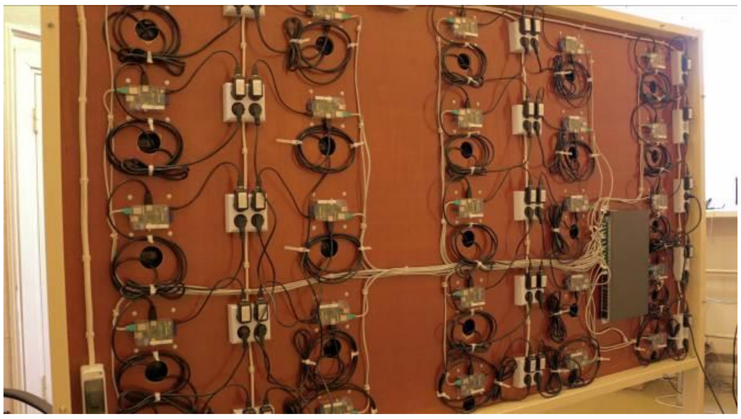

Fig. 3. LAN inter-connected Raspberry Pi devices at the back of the display wall (Bundulis and Arnicans, 2014)

scalability with this prototype and comparison regarding performance and cost against SAGE and the Reality Deck.

3. Virtualization platform

3.1. GPU virtualization

Most of the leading major commercial and noncommercial virtualization platforms support an implementation of GPU virtualization. However, they all have limitations on the maximum number of screens and resolutions. If we look at the maximum resolution for a homogeneous display surface:

- *NVIDIA vGPU* (WEB, a) that is used in Citrix XenServer allows 4 displays each at the resolution of 4096x2160 each on an NVIDIA TESLA M6card, which totals to a maximum of 35 megapixels.

- *RemoteFX* (WEB, b) on Microsoft's Hyper-V allows 8 displays at the resolution of 1280x1024 each, which totals up to a maximum of 10 megapixels (RemoteFX allows only 4 displays at the resolution 1920x1080 each but that totals up to 9 megapixels).

- *VMware vSGA* (WEB, d) allows up to 2 displays at the resolution of 1920x1200 each, which totals up to a maximum of 4 megapixels.

- *Oracle VirtualBox* (WEB, e) allows configuration of a single virtual GPU with no limits on the number of displays and resolutions as long as the framebuffers can fit into the maximum allowed video memory.

The authors decided to use Oracle VirtualBox because it had the greatest overall resolution in megapixels among the compared virtualization platforms and no hard limitations on the configuration of output count and resolution as long as the needed memory for the framebuffers fit into 256MB (Table 1). The actual number of displays varies depending on the virtualized OS. According to VirtualBox development team, each framebuffer takes up *width* × *height* × *bytes per pixel* + *memory for maintenance data*. That gives 1920 × 1080 × 4 + (4096 + 1,048,576) = 9,347,072 bytes for one framebuffer.

For Linux/X11 based operating systems, this allows 28 displays at the resolution of 1920x1080 and 60 megapixels in total. For Windows the size of the framebuffer depends on whether XPDM or WDDM drivers are used - XPDM requires an extra off-screen framebuffer and WDDM requires two extra off-screen framebuffers. The details are given in subsections 3.2 and 3.3.

The next step for the authors was to find out if and how the maximum number of theoretically possible video outputs would be perceived by each corresponding type of the guest operating system.

Table 1. Summary of virtualization platforms

Vendor	Maximum resolution for a homogenous surface	Comments
NVIDIA vGPU	35 megapixels (4 displays at 4096x2160)	The mentioned results are based on the NVIDIA TESLA M6 card, other cards provide lower or equivalent capabilities
RemoteFX (Microsoft Hyper-V)	10 megapixels (8 displays at 1280x1024)	Windows Server 2012 R2 host operating system and Windows 8/8.1 guest operating system
VMWare vSGA	4 megapixels (2 displays at 1920x1200)	Only Windows guest operating systems
Oracle VirtualBox	256/(*bytes per pixel * framebuffers per screen + some small amount of memory for bookkeeping data*) This would give 64 megapixels with standard 32bit colour depth for guest OS that require only a single frame buffer per screen (Linux/X11, Windows with XDPM diver model). Windows with WDDM driver model requires an extra off-screen framebuffer.	Theoretically, the resolution should be capped by the total memory required for the display framebuffers, which cannot exceed the total memory of the virtualized GPU (256MB).

Another aspect that needed to be verified if the total display surface size can be increased by using smaller number of displays with resolutions exceeding 1920x1080 but are multiples of this resolution – for instance 3840x2160 (2x2 1920x1080), 3840x3240 (2x3 1920x1080), 5760x3240 (3x3 1920x1080), etc. In case of such resolutions, the framebuffer manager component of the architecture would perform split of a single virtual display among multiple tiles on the physical display wall. Since in the case of VirtualBox the formula includes some bookkeeping memory for reach framebuffer. It is possible that with such configuration the overall display resolution can be increased even more.

3.2. Virtualized displays in Linux/X11

The authors had calculated that running a Linux/X11 based guest operating system should support up to 28 displays at the resolution of 1920x1080. Since the actual display wall prototype only has 25 displays the limit was checked only as far as if X11 can enumerate and detect all the displays. Authors used *Xubuntu 14.04 Linux* distributive and the ARandR display configuration tool to verify the detected monitors and resolutions.

As the authors concluded VirtualBox was able to start Xubuntu 14.04 with up to 30 displays at the resolution of 1920x1080 that theoretically exceeds the calculated limit, with greater values the guest operating system froze during the load process, but the authors could not verify if it was due to lack of the amount of video memory in VirtualBox (however it is the most probable cause).

In terms of resolutions on a single display Linux/X11 was able to resize each display to a resolution up to 5760x2160 (3x2 1920x1080) or 3840x3240 (2x3 1920x1080) thus giving 6 1920x1080 tiles per virtual display. The next logical value of 5760x3240 (3x3 1920x1080) did not work.

The authors discovered that they were able to force up to 29 virtual displays in such mode totaling up to 360 megapixels of total resolution (this conflicts with the memory limits, but X11 could perform some internal memory swapping algorithms to actually require less memory or some of the displays could have been mirrored, since the prototype wall has only 25 displays and does not support scaling at this point the authors were unable to verify layout/mirror problems with this configuration). This result will be checked by validating the actual video streams in the future.

3.3. Virtualized displays in Windows 7

The authors were able to launch, perform layout and validate the video streams on the actual display wall prototype with up to 16 displays using XPDM drivers.

With 17 displays the default Windows display manager failed to allow the layout of the 17th display, it was placed over the existing ones, and thus the displayed video was a mirrored part of the existing screen (this again could be due to the insufficient amount of video memory in VirtualBox).

The authors did not continue increasing the display count since all added displays resulted in a mirrored video output.

The results were better with using higher resolutions on a smaller amount of displays. Windows allowed one display at the resolutions of 11520x5400 (6x5 1920x1080), 9600x6480 (5x6 1920x1080) and 9600x5400 (5x5 1920x1080) that would allow using the whole physical 5x5 display wall when splitting support is added to the framebuffer manager.

Authors also verified that they can run two displays each at the resolution of 7680x4320 (4x4 1920x1080) and 5760x5400 (3x5 1920x1080). Adding additional displays seemed to limit the available resolutions to such that the summary resolution did not increase. Thus by using the splitting approach with two virtual displays each at 7680x4320 would give a total of 66 megapixels divided among 32 physical displays each at the resolution of 1920x1080.

Thus Windows 7 validated the assumption that using a single virtualized display with a higher resolution that is being split in the Framebuffer Manager allows achieving a higher overall resolution than using multiple virtual displays with lower resolutions.

4. Example use cases

The authors of the prototype have determined the possible scalability limits regarding monitors and total display resolution. The next step is to measure the actual performance in different use cases – static content, dynamic content, and 3D accelerated content. The prototype in its current state does not have a centralized statistics mechanism for measuring FPS and bitrate on each monitor, but the authors did perform a subjective evaluation of some use cases that demonstrate the capabilities of the prototype.

Authors successfully launched Xubuntu 14.04 with 25 monitors at 1920x1080 each, thus creating a homogenous surface of 52 megapixels. In comparison, the Reality Deck uses a high-end Exxact Corporation GPU server station with four high-end ATI Radeon V9800 GPUs to drive an 88-megapixel surface. Although smaller in total resolution due to the lack of available video memory in VirtualBox, the developed virtualized display wall prototype has the benefits of lower cost and power consumption.

Fig. 4 demonstrates Xubuntu 14.04 running common applications with static content like Google Maps and SVG based graphs in Firefox and a PDF viewer. All these applications seemed to work without issues regarding the high display area.

With more dynamic content, like video playback with VLC, authors noticed frame rate decrease because of the encoding speed limits on the Intel Iris Pro 5200. However, the authors plan to overcome this issue by stacking up multiple GPUs that provide hardware accelerated H.264 encoding in the next prototype.

Since Windows 7 does not provide a way to lay out more than 16 displays with the built-in tools authors had to use a subset of the monitor wall to evaluate Windows 7

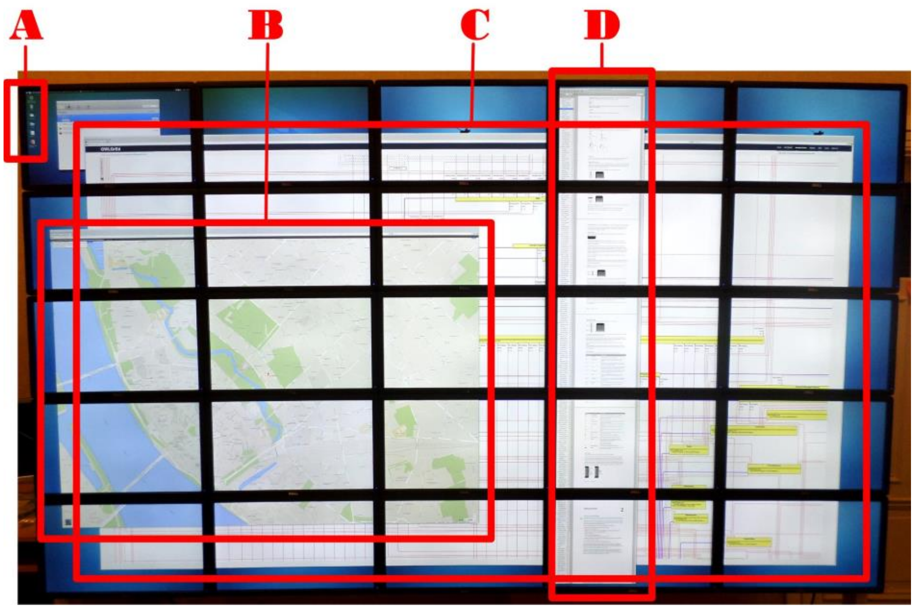

Fig. 4. Xubuntu 14.04 running common applications: A) Desktop icons; B) Google Maps in Firefox; C) SVG based graphs (Graphical Ontology Editor · OWLGrEd) in Firefox; D) PDF document in PDF viewer

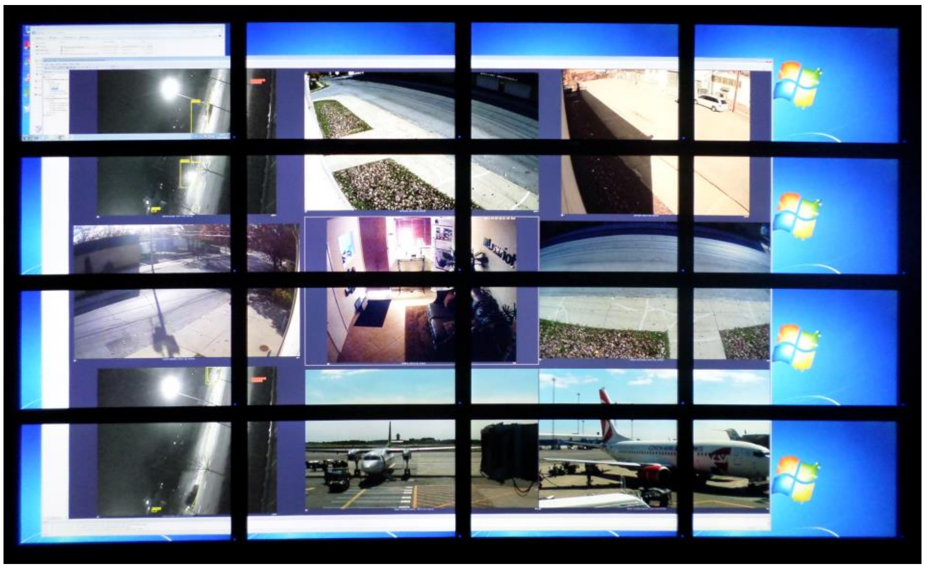

Fig. 5. Windows 7 running video surveillance software

performance. Windows 7 with 16 monitors at 1920x1080 each (total display area of 33 megapixels) was able to run video surveillance software that could be one of the possible use cases for this architecture (Fig. 5).

5. Conclusions

This paper aimed to prove the efficiency/scalability of the virtualized display wall architecture presented by Bundulis and Arnicans (2014) and address the cost/complexity/power usage and software compatibility issues present in other display wall systems.

The authors have constructed a prototype system that is driven by middle range hardware but is still able to create homogenous display surfaces up to 52 megapixels with a single GPU in comparison to 88 megapixels achieved by Reality Deck which uses expensive high-end hardware and drives the displays through physical digital video outputs. Thus for a reasonably sized display wall, the virtualization approach is very beneficial since it reduces the cost/power consumption and complexity.

Authors successfully used common (Google Chrome, PDF viewer) and domain-specific (video surveillance) software applications thus demonstrating that this approach does not force software developers to write display wall aware software – everything that works on a desktop PC works the same way in the virtualized display wall. This makes the virtualized display architecture transparent in contrast to some of the other display wall/distributed rendering solutions (e.g., Chromium) that force using graphics APIs like OpenGL.

The current prototype still lacks full 3D acceleration virtualization and centralized statistics measurement system, but these are the next steps in the roadmap of further development of the prototype.

The architecture has also opened an interesting opportunity for video games – most of the 3D accelerated video games run on a single monitor, and the newest trend of using multiple screens for gaming are achieved by different technologies from the GPU vendors in their drivers. In this article, the authors demonstrated that the virtualized display wall architecture can simulate a single large monitor with ultra-high resolution (9600x5400 pixels with Windows 7). Since the proposed architecture should be able to split a single virtual screen among physical tiles on the actual monitor wall, together with the 3D acceleration support this feature could provide a seamless way for ultra-high resolution gaming.

Although the current prototype was able to handle static data fine authors concluded that the current prototype is not able to handle dynamic content like video playback so in future authors plan to build a second prototype using more powerful hardware to see how that increases the performance of dynamic content. Authors plan to stack multiple GPUs with hardware accelerated H.264 encoding support instead of using a single GPU as in the current prototype.

Acknowledgements

The research was developed under the University of Latvia contract no. AAP2016/B032 “Innovative information technologies”.

References

Bundulis, R., Arnicans, G. (2014). Concept of virtual machine based high resolution display wall. In Information, Electronic and Electrical Engineering (AIEEE), 2014 IEEE 2nd Workshop on Advances in, IEEE, 1-6.

Bundulis, R., Arnicans, G. (2013). Architectural and technological issues in the field of multiple monitor display technologies. In: Caplinskas, A., Dzemyda, G., Lupeikiene, A., Vasilecas, O. (Eds.), *Databases and Information Systems VII: Selected Papers from the Tenth International Baltic Conference, DB and IS 2012,* Vol. 249, IOS Press, 317-329.

Carlos, V. R., Garcia, R. G. (2014). Implementation of a low cost video wall using Raspberry Pi devices. Master's thesis, Universitat Politècnica de Catalunya.

Chung, H., Andrews, C., North, C. (2014). A survey of software frameworks for cluster-based large high-resolution displays. IEEE transactions on visualization and computer graphics, 20(8), 1158-1177.

Dowty, M., Sugerman, J. (2009). GPU virtualization on VMware's hosted I/O architecture. ACM SIGOPS Operating Systems Review, 43(3), 73-82.

Humphreys, G., Houston, M., Ng, R., Frank, R., Ahern, S., Kirchner, P. D., Klosowski, J. T. (2002). Chromium: a stream-processing framework for interactive rendering on clusters. ACM transactions on graphics (TOG), 21(3), 693-702.

Jeong, B., Renambot, L., Jagodic, R., Singh, R., Aguilera, J., Johnson, A., Leigh, J. (2006). High-performance dynamic graphics streaming for scalable adaptive graphics environment. In SC 2006 Conference, Proceedings of the ACM/IEEE, IEEE.

Papadopoulos, C., Petkov, K., Kaufman, A. E., Mueller, K. (2015). The Reality Deck--an Immersive Gigapixel Display. IEEE computer graphics and applications, 35(1), 33-45.

WEB (a). NVIDIA. NVIDIA GRID Virtual GPU Technology. http://images.nvidia.com/content/grid/pdf/GRID-vGPU-User-Guide.pdf.

WEB (b). Microsoft. RemoteFX vGPU Improvements in Windows Server 2012 R2. https://cloudblogs.microsoft.com/enterprisemobility/2013/12/04/remotefx-vgpu-improvements-in-windows-server-2012-r2/.

WEB (c). VMware. Deploying hardware-accelerated graphics with VMware Horizon 7. technical white paper, `http://www.vmware.com/files/pdf/techpaper/vmware-horizon-view-graphics-acceleration-deployment.pdf`. 2017.

WEB (d). NVIDIA. NVIDIA Jetson TK1 Development Kit: Bringing GPU-accelerated computing to Embedded Systems. Technical Brief. `http://developer.download.nvidia.com/embedded/jetson/TK1/docs/Jetson_platform_brief_May2014.pdf`. 2014.

WEB (e). Oracle, VirtualBox. `https://www.virtualbox.org/` .

Received December 15, 2017, accepted December 15, 2017