GPU Consolidation for Cloud Games: Are We There Yet?

Hua-Jun Hong[1], Tao-Ya Fan-Chiang[1], Che-Run Lee[1], Kuan-Ta Chen[2], Chun-Ying Huang[3], and Cheng-Hsin Hsu[1]
[1]Department of Computer Science, National Tsing Hua University
[2]Institute of Information Science, Academia Sinica
[3]Department of Computer Science and Engineering, National Taiwan Ocean University

***Abstract*—Since the operating expense is crucial to the increasingly popular cloud gaming services, server consolidation is one of the key technologies for the success of these services. In this paper, we conduct extensive experiments using real GPUs and a complete cloud gaming platform to answer the following question: *Are modern GPUs ready for cloud gaming?* Our experiment results show that modern GPUs have low consolidation overhead, and are powerful enough to concurrently support multiple GPU-intensive cloud games. For example, when using our cloud gaming platform, the recent NVidia K2 GPU outperforms NVidia Quadro 6000 by up to 3.46 times in FPS (frame per second). Moreover, our experiments lead to two new findings that are counter to common beliefs. First, with the latest GPU virtualization technique, shared GPUs may run faster than dedicated GPUs. Second, more context switches not necessarily lead to lower FPS. Last, our experiments shed some light on further enhancements of cloud gaming platforms. For example, offloading the software video codec to the GPUs will result in better gaming experience, which is one of our future tasks.**

I. Introduction

The computer game industry generated five times higher revenues than the music industry, 15% more revenues than the consumer book sales, and roughly equal revenues as the movie industry in 2011 [10]. Moreover, *cloud gaming* has been recognized as the killer application of cloud computing [14], and attracted serious attentions from both the industry [6], [13] and the academia [11], [17]. Cloud gaming moves the games from potentially weak clients to powerful cloud servers, where the game scenes are captured, encoded, and streamed to the clients in real-time. The game scenes are decoded and rendered at clients for gamers, whose inputs are intercepted, coded, and sent back to cloud servers over the reverse channels. Cloud gaming enables gamers to play computer games anywhere, anytime on any devices, and is expected to be very popular in the near future [3].

The cloud gaming providers face a challenge for higher revenues: they want to minimize the operating expense yet achieving high gaming experience [8], [21]. One possible way to reduce the operating expense is to *consolidate* multiple virtual machines (VMs) onto a physical machine, so as to virtualize various resources, including CPUs, networks, storages, and GPUs for sharing. Server consolidation, however, has to be carefully performed, or it may result in degraded gaming experience and drive gamers away from the service. While virtualizing CPUs, network interfaces, and storages is rather mature, virtualizing GPUs is still considered experimental. In fact, several papers warn the potentially poor performance in terms of low frame rate, high response time, and low video quality when GPUs are shared among multiple VMs [4], [16]. For example, Shea and Liu [16] show that the frame rate of Doom 3 is lower than 40 FPS (frame-per-second) even if the hypervisors (Xen and KVM) are configured with one-to-one GPU pass-through, which indicates that sharing the GPU among multiple VMs is virtually impossible.

978-1-4799-6882-4/14/$31.00

Nonetheless, in the past couple of years, the GPU virtualization technology has been dramatically improved, which *may* have solved the performance issue of GPU consolidation. In this paper, we conduct detailed experiments using modern GPUs and a real cloud gaming platform called GamingAnywhere (GA) [9] to answer the following question: *Are modern GPUs ready for cloud gaming?* In particular, we perform two types of experiments: (i) *end-to-end experiments* using the complete cloud gaming platform to quantify the overall gaming performance, and (ii) *GPU-only* experiments using only GPUs to zoom into their detailed performance. Our end-to-end experiments demonstrate that modern GPUs, such as NVIDIA K2, significantly outperform the earlier generation GPUs, such as Quadro 6000, by up to 3.46 times in FPS. This can be attributed to both Moore's law and more advanced GPU virtualization technologies. Moreover, the cloud gaming platform is stable under different network conditions, such as various bandwidth, delay, and packet loss rate, and in the Internet. On the other hand, we also find that the cloud gaming server with the modern GPUs may become *CPU-bounded*. Hence, offloading the software video codec to the GPUs will further improve the gaming experience.

Our GPU-only experiments reveal several insights that have never been reported in the literature. For example, we observe that: (i) virtualized GPUs may *outperform* pass-through GPUs, (ii) more context switches not necessarily result in lower FPS, and (iii) the hypervisor is not a bottleneck for managing the virtual GPUs. Some of the observations are different from the previous studies [16], and can be attributed to the recent advances on GPU virtualization. The findings in the GPU-only experiments show the merits of modern virtualized GPUs and shed some light on how to optimize the configurations of cloud gaming platforms.

The rest of this paper is organized as follows. We survey the literature in Sec. II. Sec. III presents our testbed setup and measurement methodology. This is followed by the experiment results and discussions in Sec. IV. Sec. V concludes this paper.

II. Related Work

GPU architecture has been constantly changing, which renders virtualizing GPU for multiple VMs fairly challenging. Dowty and Sugerman [4] discuss several GPU virtualization techniques, which can be roughly classified into software-based and pass-through. The software-based GPU virtualization is compatible with more GPU hardware, while the pass-through approach achieves better performance. The software-based virtualization is more flexible, adopted by VMWare [4], and used in prototyping optimization algorithms for GPU scheduling [23]. The pass-through approach can further be classified into: (i) one-to-one fixed pass-through and (ii) one-to-many mediated pass-through [4].

Until very recently, commercial products did not support mediated pass-through and refer to fixed one-to-one pass-through as pass-through. In fact, the performance of one-to-one fixed pass-through GPUs for cloud gaming has not been studied until late 2013 [16]. More precisely, Shea and Liu [16] conduct extensive experiments to quantify the performance gap between native hardware (without virtualization) and pass-through GPU (with virtualization). They find that the native hardware significantly outperforms pass-through GPU, and conclude that sharing a GPU among multiple cloud gaming VMs will lead to unacceptable low frame rate. Their measurement results also reveal that the low frame rates are partially attributed to the excessive context switches. Our current paper extends Shea and Liu [16] in the sense that: (i) we consider modern GPUs that support more advanced (mediated) pass-through approach, and (ii) we quantify the performance of GPUs shared by multiple VMs. To our best knowledge, the performance of these modern GPUs has not been measured in the literature.

III. Experiment Methodology

We present our testbed implementation and measurement methodology in this section.

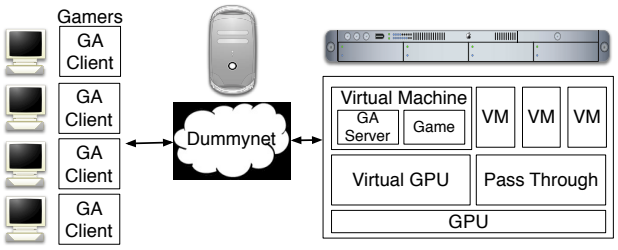

Fig. 1: Our testbed consists of a cloud gaming server running multiple GA servers and games, a dummynet router, and several GA clients.

TABLE I: Specifications of The Two Considered GPUs.

GPU	Year	Core	Memory	No. Inst.	vSGA	vGPU
Quadro 6000	2010	448	6 GB	1	Yes	No
K2	2013	3072	8 GB	2	Yes	Yes

A. Testbed Setup

Fig. 1 illustrates our testbed setup. We setup a XenServer 6.2 on a cloud gaming server with a Xeon 2.1 GHz 24-core CPU and 64 GB memory. We conduct experiments with two GPU cards: NVIDIA Quadro 6000 (released in 2010) and NVIDIA K2 (released in 2013). Their specifications are given in Table I. The K2 GPU has two physical GPU instances, and each instance can be independently configured to be in one of the following modes: (i) pass-through, (ii) vGPU with up to 2 VMs, (iii) vGPU with up to 4 VMs, and (iv) vGPU with up to 8 VMs. vGPU is the latest NVIDIA GPU virtualization technique realizing mediated pass-through [4]. We refer to these modes as: PassThrough, vGPU_2 (2 VM on each GPU instance), vGPU_4 (4 VMs on each instance), and vGPU_8 (8 VMs on each instance), respectively. Since each K2 GPU contains two instances, we can configure it for up to 16 VMs. Quadro 6000 GPU does not support vGPU and only works with vSGA, which is software-based GPU sharing. If not otherwise specified, we allocate 1 CPU core and 2GB memory to Dom0, which manages all VMs. The remaining CPU cores and memory are equally divided among the VMs running Windows 7. Next, we set up the GA server [9] (as of August 2014) in these Windows VMs, which are connected to 8 Windows machines running GA client. GA is an open-source cloud gaming platform, which is modularized, cross-platform, and efficient. Therefore, GA is suitable to the research community for testing innovative ideas, to the service providers for developing the systems, and to the gamers for setting up private cloud gaming servers. To emulate diverse and dynamic network conditions, we add a dummynet router [1] to impose additional bandwidth constraints, network delay, and packet loss rate, between GA clients and servers.

B. Workload Generators

We generate GPU workload using two kinds of applications on Windows 7. The details are given below.

- **Game.** Three games are chosen from different game genres: Fear2 is a first-person shooter game, LEGO Batman is an action game, and Limbo is a scroll-based puzzle game. In all these games, the graphics detail levels are kept as default.
- **Benchmark.** We use Sanctuary for overall GPU benchmark and Cadalyst for detailed (2D versus 3D) GPU benchmark. The Sanctuary benchmark gives an FPS number as the overall score. The Cadalyst benchmark gives four 2D scores on ortho lines, radial lines, texts/blocks, and erase/zoom, and we denote them as 2D_1, 2D_2, 2D_3, and 2D_4 in the figures. The Cadalyst benchmark also gives four 3D scores on rotate wireframe, rotate hidden, rotate conceptual, and rotate realistic, which are referred to as 3D_1, 3D_2, 3D_3, and 3D_4 in the figures. We set the resolution for benchmarking to 1920x1200.

For fair comparisons, we use `TinyTask` [18] to record the mouse and keyboard inputs of a 3 minutes gameplay session of each game. We then replay the same user inputs in each experiment with different system parameters.

C. Performance Metrics

In addition to the scores given by Sanctuary and Cadalyst benchmark, we also consider the following metrics.

- **FPS.** The number of rendered frames per second.
- **Context switch.** The number of context switches in Dom0.
- **CPU utilization.** The CPU load of Dom0 (CPU_{dom0}) and each VM (CPU_{vm}).
- **GPU utilization.** The load of GPUs.
- **Frame loss rate.** The fraction of frames that are not rendered at the GA client due to packet loss or late arrival.
- **PSNR (Peak Signal-to-Noise Ratio) [19]** and **SSIM (Structural Similarity) [20].** The video quality at the GA clients, compared against the videos captured at the GA servers.
- **Response delay** The time difference between a gamer generates an input and the GA client renders the first frame affected by that input.

When reporting the results, we give 95% confidence intervals whenever possible.

D. Measurement Utilities

XenServer is a patched CentOS Linux and thus may not support all Linux utilities. For example, the PAPI and Perf do not work on XenServer, and we adopt the following utilities:

- **Fraps [5].** To measure the FPS of the foreground window.
- **Sar [15].** To measure the number of context switches.
- **Xentop [22].** To measure the CPU utilization of Dom0 and VMs.
- **Nvidia-smi [12].** To measure the GPU utilization under vGPU.
- **GPU-Z [7].** To measure the GPU utilization of pass-through GPUs.

In addition, we write several tools to derive measured values. For example, the frame loss rates are calculated using color-coded frame numbers.

TABLE II: Achieved frame rates on two considered GPUs

# of VMs	Quadro 6000	K2	Speed-up (times)
2 VMs	22.3	25.1	1.13
4 VMs	13.1	32.3	2.47
8 VMs	7.0	24.2	3.46

IV. Experiment Results

We first use the complete GA platform to compare the modern vGPU technology with the previous generation one. We then isolate and zoom into the GPU performance. This is followed by comprehensive end-to-end experiments using the complete GA platform.

Performance edge and scalability of vGPU. We first compare the performance of Quadro 6000 (software-based vSGA) and K2 (mediated pass-through vGPU) using the GA testbed. We report the achieved average FPS from Limbo at 1.5 Mbps, which is the streaming rate of GA server. in Table II. This table shows that K2 outperforms Quadro 6000 by up to 3.46 times in terms of FPS. Furthermore, the FPS achieved by K2 does not drop too much even with 8 VMs, which shows its scalability. This experiment demonstrates the huge edge of vGPU (mediated pass-through) over vSGA (software-based virtualization). Hence, we no longer consider Quadro 6000 and vSGA in the rest of this paper.

TABLE III: Sanctuary Scores in FPS from Different GPU Configurations

GPU Configuration	PassThrough	vGPU
1 PassThrough on 1 Instance	150.4	x
2 PassThrough on 2 Instances	146.6	x
1 PassThrough + 4 $vGPU_4$	142.7	45.9
4 VMs with $vGPU_4$ on 1 Instance	x	42.7
8 VMs with $vGPU_4$ on 2 Instances	x	42.8

Independence of the two K2 GPU instances. We use the benchmark Sanctuary in the VMs on the GA server to measure the FPS of pass-through, vGPUs, and mixed configurations. We report the average results in Table III, which shows that the two GPU instances of K2 operate independently, as the FPS values are pretty stable. To reduce the experiment complexity, we only enable a K2 instance in the rest of this paper.

Shared GPUs may outperform dedicated GPUs. We next compare the performance of pass-through, $vGPU_2$, $vGPU_4$, and $vGPU_8$ by running Sanctuary, Fear2, Batman, and Limbo on the GA servers, where pass-through refers to one-to-one fixed pass-through. We plot the resulting FPS in Fig. 2(a). This figure reveals a surprising observation: vGPU results in higher FPS than pass-through when executing Limbo and Fear2. This is different from the common belief. We therefore take a step further by running Cadalyst in the VMs and reporting the 2D and 3D scores in Figs. 2(b) and 2(c). These two figures show that $vGPU_2$ outperforms pass-through in all 2D operations by up to 15%. Moreover, $vGPU_2$ also leads to better scores than pass-through when executing some 3D operations (two out of four scores). Similar observations are also true for $vGPU_4$ and $vGPU_8$. Such observations explain the inferior FPS of pass-through GPUs: Limbo and Fear2 heavily rely on 2D operations; in contrast, the 3D-intensive Batman performs better on pass-through GPUs. The take-away of this experiment is that state-of-the-art vGPU virtualization technique has been well optimized and works better than pass-through GPUs for some game genres. Therefore, sharing a GPU among multiple VMs running games is now a reality.

TABLE IV: Relation Between FPS and Number of Context Switches

Game	FPS			No. Context Switches		
	$vGPU_8$	$vGPU_4$	Ratio	$vGPU_8$	$vGPU_4$	Ratio
Fear2	45.8	64.9	0.7	9472	14149	0.67
Batman	43.3	41.6	1.04	4325	3991	1.08
Limbo	39.2	64.8	0.6	10700	13927	0.76

The performance of modern GPUs is no longer dominated by the overhead due to context switches. An earlier study [16] reports that more context switches incur higher overhead and reduce the rendered FPS. We try to reproduce

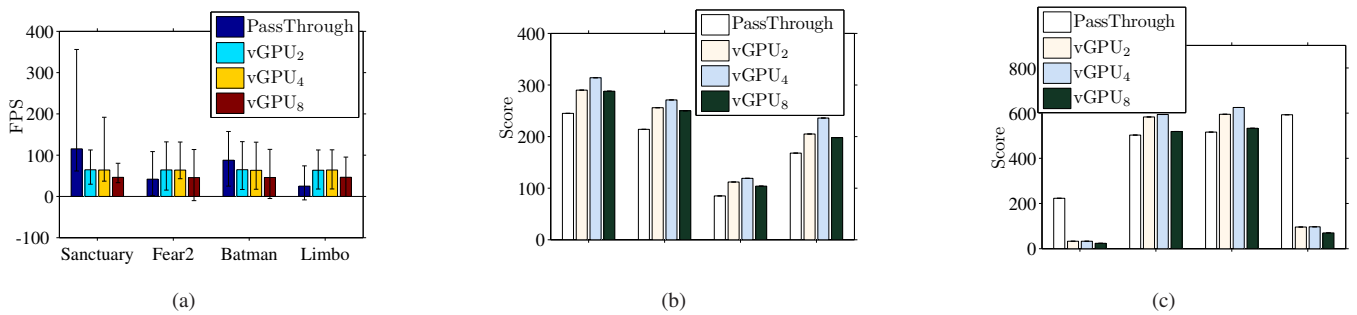

Fig. 2: Comparing the pass-through and vGPU: (a) resulting FPS, (b) 2D benchmark scores, and (c) 3D benchmark scores.

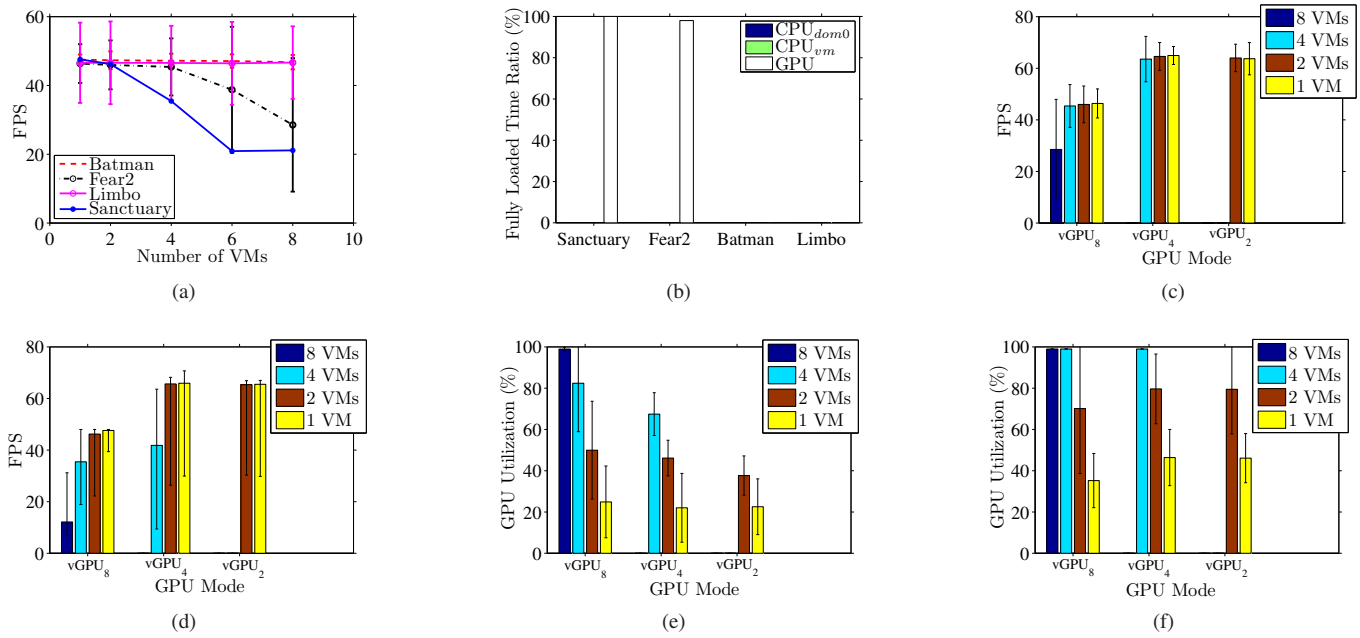

Fig. 3: GPU consolidation overhead: (a) resulting FPS with various numbers of VMs, (b) fully loaded time ratio from $vGPU_8$, (c), (d) resulting FPS, samples from Fear2 and Sanctuary, and (e), (f) resulting GPU utilization, samples from Fear2 and Sanctuary.

this by running three games on $vGPU_4$ and $vGPU_8$, and measure the resulting FPS and number of context switches in Dom0. We report the results in Table IV, which shows that the FPS is proportional to the number of context switches. That is, more context switches indicate that VMs are busier, leading to higher FPS, which is different from the earlier study [16].

Consolidation overhead and root cause analysis. Next, we quantify the consolidation overhead by configuring the K2 GPU into $vGPU_8$, and gradually adding more VMs (from 1 to 8). We then measure the FPS, CPU_{dom0}, CPU_{vm}, and GPU utilization when running Sanctuary and each game in the VMs. We plot the resulting FPS in Fig. 3(a). This figure shows that Limbo and Batman do not suffer from consolidation overhead, while Fear2 and Sanctuary do. We then compute the fraction of time each resource is fully loaded, and refer to it as *fully loaded time ratio* in %. We plot the sample ratio from $vGPU_8$ in Fig. 3(b), which shows that Sanctuary and Fear2 are bounded by GPU, while Limbo and Batman are not bounded by any resource. To take a deeper look at the consolidation overhead of Fear2 and Sanctuary, we repeat the same experiments with 1, 2, 4, and 8 VMs on $vGPU_8$, $vGPU_4$, and $vGPU_2$. We plot the achieved FPS and GPU utilization in Figs. 3(c)–3(d) and 3(e)–3(f), respectively. We make two observations out of these figures. First, under the same vGPU mode, fewer VMs lead to higher FPS and lower GPU utilization. For example, under $vGPU_8$, moving from 8 VMs to 4VMs, the FPS increases to 35 (Fear2) and 45 (Sanctuary), respectively. This indicates that K2 dynamically allocates GPU resources among all VMs, rather than statically distributing a fixed share to each VM. Second, we observe that even when the GPU utilization is not saturated, the FPS never exceeds 48 and 66 under $vGPU_8$ and other modes, respectively. This indicates that an FPS-aware GPU scheduling algorithm, similar to Zhang et al. [23], has been implemented, making vGPU even more suitable for sharing GPUs among VMs running cloud games.

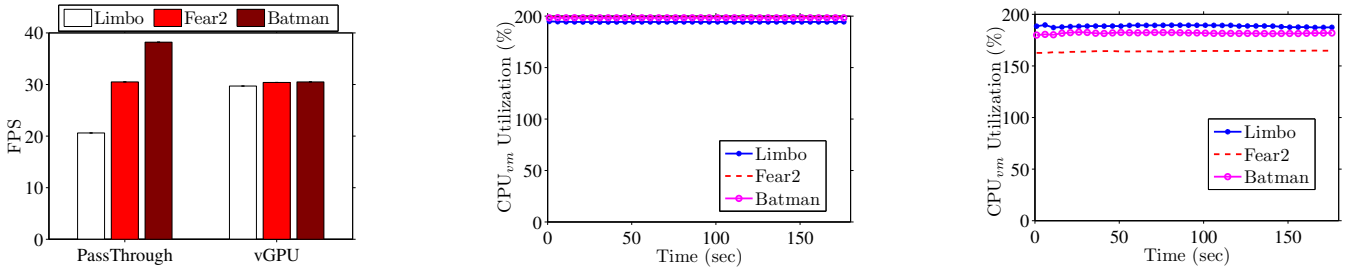

Fig. 4: End-to-end performance of a complete GA platform: (a) resulting FPS, (b) CPU_{vm} utilization with pass-through GPU, and (c) CPU_{vm} utilization with $vGPU_2$.

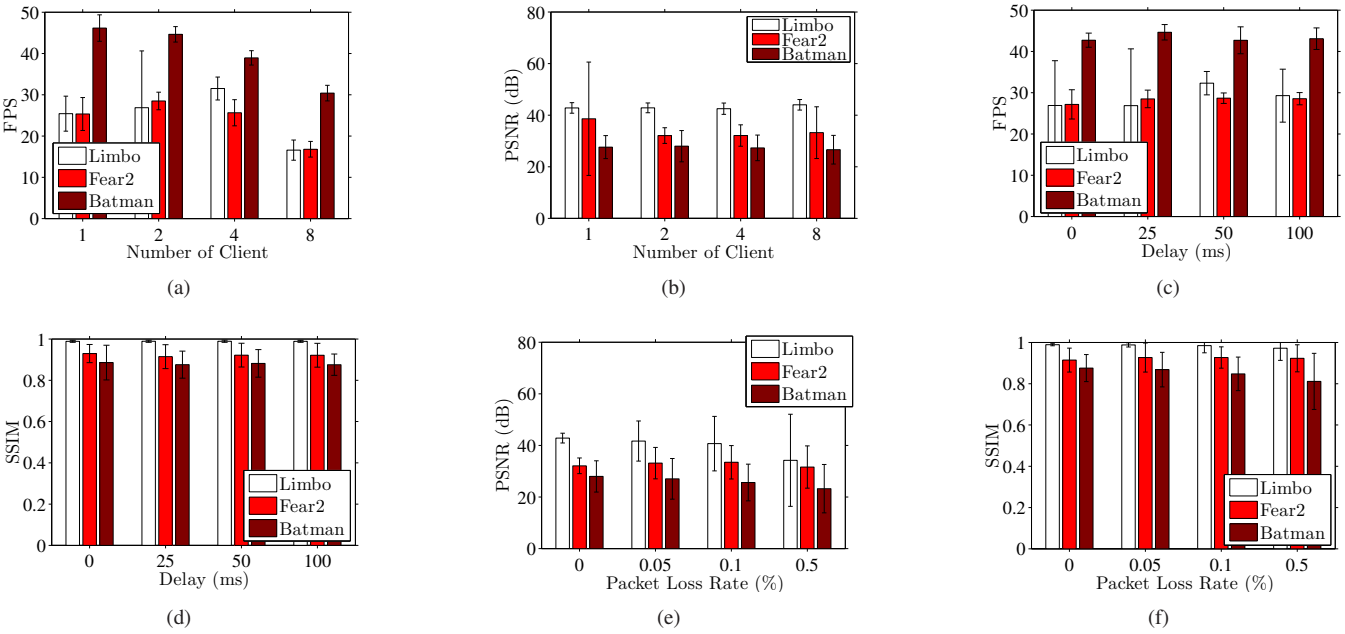

Fig. 5: End-to-end performance of a GA platform with dummynet: (a) FPS and (b) sample PSNR under different numbers of clients; (c) FPS and (d) sample SSIM under different transmission delays; (e) PSNR and (f) SSIM under different packet loss rate.

Importance of hardware codecs. Next, we measure the end-to-end cloud gaming performance using the complete GA platform. We configure the server for pass-through and $vGPU_2$, assign 8 CPU cores to each VM, and measure the rendered FPS at clients and the CPU utilization at the server, which is configured to stream at 1.5 Mbps. We report the resulting FPS values in Fig. 4(a), which are between 20 and 42. The FPS results are less ideal to high-quality cloud gaming, and a closer look indicates that this is because of limitations of XenServer and Windows 7. In particular, because the GA server [9] relies on the CPUs for real-time video encoding, more CPU cores mean higher encoding speeds. However, the free version of XenServer only supports exposing multiple virtual CPUs, rather than CPU cores, to each guest OS, and Windows 7 supports up to 2 CPUs. Hence, the guest Windows 7 only schedules the tasks to 2 CPUs while leaving the other 6 CPUs idle (see Figs. 4(b) and 4(c)), which renders lower FPS values. We note that K2 GPUs come with hardware H.264 codecs, and how to leverage these codecs for higher cloud gaming FPS is an interesting future task.

Performance under diverse network conditions. We set the network latency to be in $\{0, 25, 50, 100, 200\}$ ms, the bandwidth to be in $\{10, 15, 20, 40\}$ Mbps, the packet loss rate to be in $\{0\%, 0.05\%, 0.10\%, 0.5\%\}$, and the number of clients to be in $\{1, 2, 4, 8\}$. If not otherwise specified, we set 0 ms delay, 15 Mbps bandwidth, 0% packet loss rate, and launch 2 GA clients in the following experiments. The network constraints are applied to both uplink and downlink using dummynet [1]. The XenServer is configured with the GPU modes that match the number of clients, and the GA clients are configured to stream at 2 Mbps. We vary the number of clients and give the FPS and sample PSNR results in Figs. 5(a) and 5(b). Fig. 5(a) shows that the FPS decreases once the number of consolidated clients is increased, and the FPS is at least 18. Fig. 5(b) shows that the PSNR values are always higher than 21, 32, and 42 dB in Batman, Fear2, and Limbo, respectively.

We vary the network latency, and report the FPS and sample SSIM results in Figs. 5(c) and 5(d), which reveal that the FPS and SSIM values are always higher than 25 and 0.9, even when the latency is long. In the part where we vary the server bandwidth, we notice that once the bandwidth exceeds 8 Mbps, each game's performance is fairly consistent, e.g.,: (i) in terms of FPS, Limbo, Fear2, and Batman achieve at least 26.86, 28.49, and 42.14, (ii) in terms of sample PSNR, Limbo, Fear2, and Batman achieve at least 42.61, 32.08, and 27.67 dB, (iii) in terms of sample SSIM, Limbo, Fear2, and Batman achieve at least 0.9887, 0.9038, and 0.8737. In all the above experiments, the frame loss rates are always less than 2.38%. We vary the packet loss rate and report the resulting PSNR and SSIM in Figs. 5(e) and 5(f). These two figures show that the video quality slightly drops as the packet loss rate increases. The frame loss rates of Limbo, Fear2, and Batman are 4.97%, 5.21%, and 6.88% under 0.5% packet loss rate. In summary, Fig. 5 demonstrates that the GA platform performs well on modern K2 GPUs under diverse network conditions.

Response delay in real networks. Response delay is critical to gamers, and we set up a real testbed to measure the response delay over a Trans-Pacific fiber between Taiwan and California. We configure the K2 server in Taiwan into vGPU_4 mode, and launching 3 GA clients in Davis, CA playing Limbo, Fear2, and Batman streaming at 2 Mbps. The experiments are conducted at 12 p.m. (GMT+8) on September 1, 2014. We measure the response delay by pressing the ESC button and capturing in-game video at the GA client to find the first frame with the pop-up menu. Their time difference is the response delay [2]. We measure the response delay 5 times. We then move the games and GA servers out of the XenServer, and repeat the same experiments. Table V gives the response delay with and without XenServer. We make two observations from this table. First, given the round-trip-time of about 140 ms between Taiwan and California, the response delay of our GA platform is short. Second, the virtualization overhead is negligible in terms of response delay. This confirms that the modern virtualization techniques are ready for cloud gaming.

TABLE V: Response Delay Between Taiwan and California

Game	Platform	Delay (ms)				
Limbo	XenServer	250	260	265	275	310
	Native	250	255	265	265	270
Fear2	XenServer	250	290	314	350	355
	Native	250	254	280	300	305
Batman	XenServer	215	220	230	250	260
	Native	225	235	240	250	260

V. Conclusion

In this paper, we designed and carried out detailed measurement studies to understand whether the state-of-the-art GPU virtualization techniques are ready for cloud gaming. We have found that the mediated pass-through GPU virtualization implemented in the latest GPUs enables efficient GPU sharing among multiple VMs. In fact, shared GPUs: (i) may outperform dedicated GPUs and (ii) are rather scalable to the number of VMs. Therefore, modern GPUs can be shared by VMs running GPU-intensive computer games. We also observed that CPUs may become the bottleneck on the VMs in the data center if the video coding is done in software, because a guest Windows 7 OS can only utilize up to 2 CPU cores. Hence, leveraging the hardware video codecs on GPUs is an attractive option to cloud gaming platforms. Furthermore, we evaluate the end-to-end performance of GA using an open-source cloud gaming platform [9] in both a dummynet testbed and in the live Internet. The experiment results shows that the cloud gaming platform achieves stable performance under diverse network conditions and in the Internet.

References

[1] M. Carbone and L. Rizzo. Dummynet revisited. *ACM SIGCOMM Computer Communication Review*, 40(2):12–20, April 2010.

[2] K.-T. Chen, Y.-C. Chang, H.-J. Hsu, D.-Y. Chen, C.-Y. Huang, and C.-H. Hsu. On the quality of service of cloud gaming systems. *IEEE Transactions on Multimedia*, 16(2):480–495, February 2014.

[3] Distribution and monetization strategies to increase revenues from cloud gaming. http://www.cgconfusa.com/report/documents/Content-5minCloudGamingReportHighlights.pdf.

[4] M. Dowty and J. Sugerman. GPU virtualization on VMware's hosted I/O architecture. *ACM SIGOPS Operating Systems Review*, 43(3):73–82, July 2009.

[5] FRAPS game capture video recorder fps viewer. http://www.fraps.com/.

[6] Gaikai web page. http://www.gaikai.com/.

[7] GPU-Z video card gpu information utility. http://www.techpowerup.com/gpuz/.

[8] H. Hong, D. Chen, C. Huang, K. Chen, and C. Hsu. Placing virtual machines to optimize cloud gaming experience. *IEEE Transactions on Cloud Computing*, May 2014. Accepted to Appear.

[9] C. Huang, K. Chen, D. Chen, H. Hsu, and C. Hsu. GamingAnywhere: The first open source cloud gaming system. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 10(1s):10:1–10:25, January 2014.

[10] A. Marchand and T. Hennig-Thurau. Value creation in the video game industry: Industry economics, consumer benefits, and research opportunities. *Journal of Interactive Marketing*, 27(3):141–157, August 2013.

[11] D. Mishra, M. E. Zarki, A. Erbad, C. Hsu, and N. Venkatasubramanian. Clouds + games: A multifaceted approach. *IEEE Internet Computing*, 18(3):20–27, May-June 2014.

[12] NVIDIA System Management Interfacei. https://developer.nvidia.com/nvidia-system-management-interface.

[13] Onlive web page. http://www.onlive.com/.

[14] P. Ross. Cloud computing's killer app: Gaming. *IEEE Spectrum*, 46(3):14, March 2009.

[15] Sar man page. http://www.linuxcommand.org/man_pages/sar1.html.

[16] R. Shea and J. Liu. On GPU pass-through performance for cloud gaming: Experiments and analysis. In *Proc. of ACM Annual Workshop on Network and Systems Support for Games (NetGames'13)*, Denver, CO, December 2013.

[17] R. Shea, J. Liu, E. Ngai, and Y. Cui. Cloud gaming: Architecture and performance. *IEEE Network*, 27(4):16–21, July-August 2013.

[18] TinyTask. http://www.vtaskstudio.com/support.php#tools.

[19] Y. Wang, J. Ostermann, and Y. Zhang. *Video Processing and Communications*. Prentice Hall, 2001.

[20] Z. Wang, L. Lu, and A. Bovik. Video quality assessment based on structural distortion measurement. *Signal Processing: Image Communication*, 19(2):121–132, February 2004.

[21] D. Wu, Z. Xue, and J. He. iCloudAccess: Cost-effective streaming of video games from the cloud with low latency. *IEEE Transactions on Circuits and Systems for Video Technology*, December 2013. Accepted to appear.

[22] Xentop man page. http://wiki.xen.org/wiki/Xentop(1).

[23] C. Zhang, Z. Qi, J. Yao, M. Yu, and H. Guan. vGASA: Adaptive scheduling algorithm of virtualized GPU resource in cloud gaming. *IEEE Transactions on Parallel and Distributed Systems*, November 2013. Accepted to appear.